# Evolutive system architecture design : design, optimization, and implementation of adaptable, regenerative, and reactive complex system architectures in the era of climate scarcity

Raul Polit Casillas

**UNIVERSITÉ DE STRASBOURG**

***ÉCOLE DOCTORALE MSII***

***ICUBE***


# THÈSE


présentée par :

**Raul POLIT CASILLAS**


soutenue le : **17 mars 2021**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/ Spécialité : Ingénierie Système

# Evolutive Systems Architecture Design

## Design, optimization, and implementation of adaptable, regenerative, and reactive complex system architectures in the era of climate scarcity


**THÈSE dirigée par :**

**M. CAILLAUD Emmanuel** — Professeur des Universités, université de Strasbourg

**RAPPORTEURS :**

**Mme BARON Claude** — Professeur des Universités, INSA Toulouse

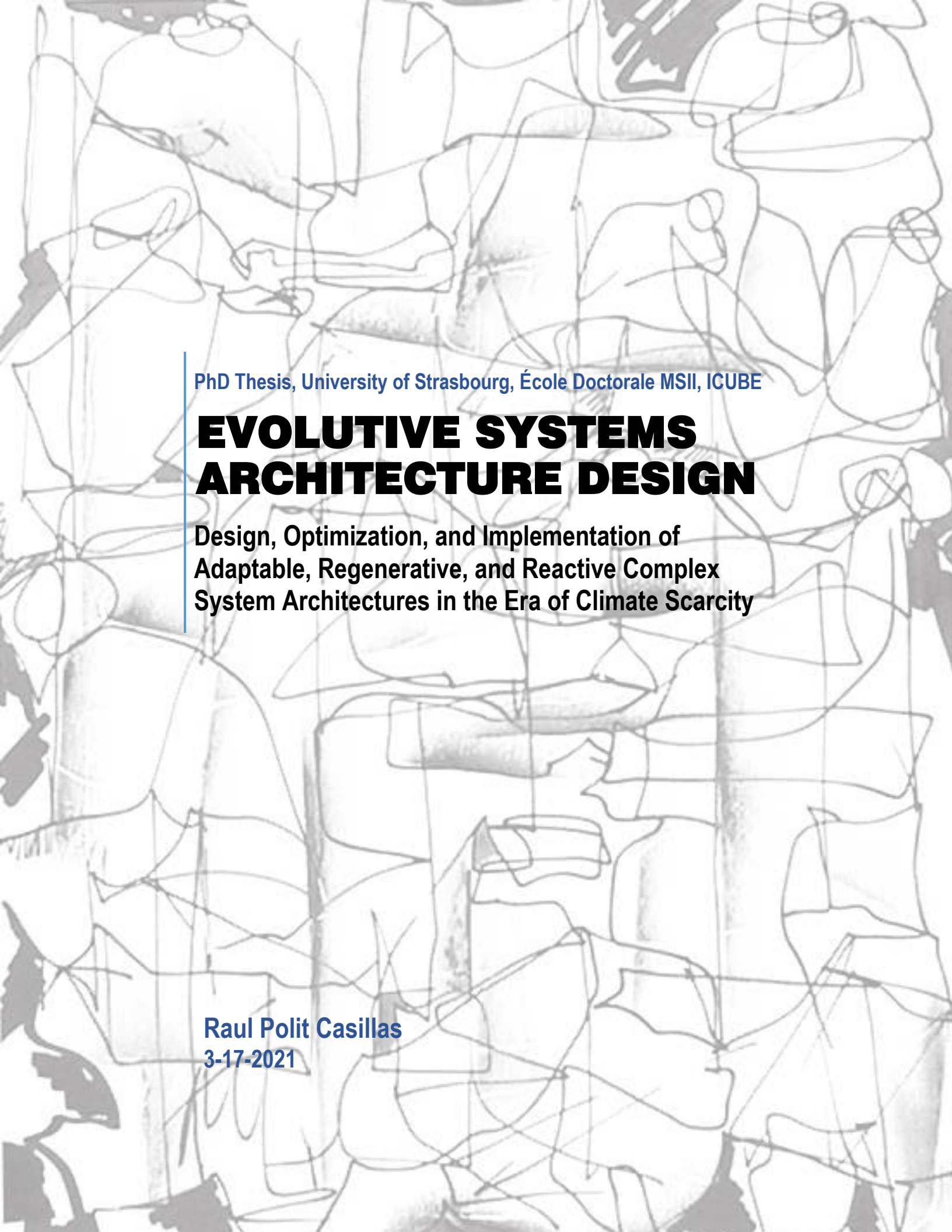
**M. BONJOUR Eric** — Professeur des Universités, ENSGSI-université de Lorraine

---

**AUTRES MEMBRES DU JURY :**

**Mme BERGER-WOLF Tanya** — Professeur, Ohio State University / TDAI

**M. Naderi Firouz** — Retired, JPL Caltech



PhD Thesis, University of Strasbourg, École Doctorale MSII, ICUBE

# EVOLUTIVE SYSTEMS ARCHITECTURE DESIGN

## Design, Optimization, and Implementation of Adaptable, Regenerative, and Reactive Complex System Architectures in the Era of Climate Scarcity

Raul Polit Casillas

3-17-2021

# Evolutive Systems Architecture Design

## Design, Optimization, and Implementation of Adaptable, Regenerative, and Reactive Complex System Architectures in the Era of Climate Scarcity

by
**Raul Polit Casillas**

Submitted to the
**École Doctorale « Mathématiques, Sciences de l'Information et de l'Ingénieur », ICUBE**
**on March 17th, 2021**
for the degree of
**Doctor of Philosophy in Systems Engineering Design**
at the
**Université de Strasbourg**

Advisor/Director: Professor Emmanuel Caillaud, University of Strasbourg
Thesis study case: Evolutive Portable Habitat (EPH) © 2020 Raul Polit Casillas

*This work was done as a private venture and not in the author's capacity as an employee of the Jet Propulsion Laboratory, California Institute of Technology. This is a fundamental research activity presenting only theoretical results.*

*Cover image by Ramon Polit Alabau – "Intenciones" Limited Series*

# Abstract

Human activity on Earth has been driven by the need to innovate towards the next level. Survival needs, competitive advantage, and intellectual curiosity, among others, have incentivized us to go beyond, and often against the everlasting hassle of finding resources or support. Around the second decade of the century, complexity, heritage, and resource scarcity among others, are increasingly influencing hardware-based complex architectures in terms of design, optimization, and implementation. Based upon critical synergies among domains such as systems engineering, architecture, and engineering design this thesis presents results, approaches, and contributions towards a novel system design methodology.

Nowadays, complex hardware-based systems such as cars, computers, robotic systems, virtual platforms, smart buildings, and other electro-mechanical devices show a growing need for quantum-leaps in terms of system performance, which are often beyond the limits of any existing heritage. For instance, consumer products have become more sophisticated by the day, requiring a better integration of hardware and software, as well as other social and cultural requirements to be competitive. Purely mechanical systems a few decades ago, like an automobile, today include hundreds of thousands of lines of code and showcase other disruptive manufacturing techniques (e.g., additive manufacturing) to deliver better quality, cheaper complexity, and easier customization at a lower price. However, beyond the competitiveness of a product within markets worldwide, the demand for better system performance (e.g., sustainable houses consuming less energy), and system adaptability (e.g., modular systems) is a growing trend partially based on the infusion of data-driven capabilities such as processes, product characteristics, or operational schemes. Namely, telecommunication businesses nowadays are no longer just about transmitting data over large distances, since they need to involve social trends, subsystems connectivity, and user experiences as well. In essence, such inherent new complexity is assumed in this research as a multidisciplinary networked reality rather than a unidimensional challenge, because our world is getting much more complex very fast, so our design methods must evolve in parallel as well.

At the same time, we are entering a whole new phase in terms of resource availability due to climate uncertainty and population growth, as well as other socioeconomic factors. Therefore, the balance between the need for complexity and the availability of resources (e.g., energy, workforce, building materials, etc.) is entering a new paradigm, which is the starting point for this research. Regardless the field of work (e.g., architecture, car-making, finance, product design, medicine, aerospace, etc.) the need to go beyond in terms of system performance, novelty, efficiency, uniqueness, and adaptability is becoming a major force in the design of any complex technical endeavor. Markets, customers, and requirements will keep demanding more of any system architecture, affecting 'what' they are as a system (artifact), and 'how' they are being developed (method) across all multiple development phases such as: design, optimization, prototyping, implementation, management, and sustainability. Thus, considering heritage constraints and resource scarcity, **how could we achieve much better levels of system performance and capability when developing new complex systems?** Furthermore, **how could we design those systems better with less resources while using faster and more efficient means?**

This dissertation presents theoretical bases, literature reviews, practice gaps, methodologies, and study cases of a **novel, fast-paced, and synergetic technical approach towards design systems engineering of complex system architectures**. Inspired by evolutionary principles, adaptive principles, and proven state-of-the-art techniques, this **evolutive architecture approach** tackles **design, optimization, and implementation** of complex systems under such tight constraints while it is focused on multiple connections across disciplines. The overarching goal of this approach is to **quickly overcome design barriers that are driven by heritage, performance, and uniqueness** in the same way that nature does, as a **continuous and efficient process building upon synergies rather than disciplinary and subsystem divisions**. Within this method those three key areas are linked as nodes of a networked approach**.**

However, this thesis is structured and centered mainly around the design node of the methodology while highlighting other phases such as implementation and operations. To exemplify this methodology, **a smart portable habitat system** is being used as a study case to introduce and elaborate critical methodology tools and principles, due to the complexity of the topic and the importance of system heritage in the field of architectural design and construction.

After an introductory chapter, a second part presents the context and rationale for this synergetic method tackling stressors, barriers, enablers, and gaps. The need to design better and more efficiently implies also to do so faster and cheaper. This is founded on **upcoming climate, socio-economic, and technical uncertainties** driving new balances

among system needs, resources, and heritage. Technical, science, market-driven needs compete for better and faster performance, leading to increasingly more complex systems. In the long term, this not only stresses current design capabilities, but also it makes more difficult to embrace new solutions, especially for heritage-rich and risk-averse sectors and organizations. Furthermore, such evolutionary-based method should provide **adaptability, scalability, and efficiency** towards any dramatic improvement enabled by current state-of-the-art solutions. Thus, a third chapter presents an extensive literature review tackling design methods, theories, and systems engineering approaches. Frugal, social, and low-tech design trends, among many others propose multiple options toward doing 'more with less', however **this evolutive approach tackles doing 'better with less'** in the context of 'high-tech' nature-inspired designs and system design engineering domains. Evolutive methods are driven towards systematic, radical, and disruptive change instead of incremental innovation.

Within such broad context, the fourth chapter makes an emphasis in a series of key characteristics within hardware-based system architectures that are increasingly becoming more critical due to multiple sources of resource scarcity, as well as the need to handle much more system complexity both as a product and within the development process. **Evolutive architectures** are defined by a **regenerative** approach toward the use of resources, a high-level system **adaptability**, and a **reactivity-driven** operational mode.

Under the evolutive perspective, any complex system could be described by its **geometry** (descriptive principles), its **behavior** (functional principles), and its **substance** (component nature). Thus, the next chapter lays out the evolutive system design methodology upon such context. Chapter six presents a study case that exemplifies evolutive principles, steps, tools, and criteria used to efficiently obtain feasible and ultra-performance design solutions fast, while being tool agnostic. This example provides the baseline to answer all research questions as well as to obtain conclusions for the multiple contributions developed and presented in this doctoral dissertation.

The foundation of this research is based on almost **20 years of professional experience designing complex** systems. This thesis is complemented by other fundamental research examples of public domain that have also published by the author during his activity at the Jet Propulsion Laboratory of NASA-Caltech for almost a decade. Hence, the guidelines and findings presented in this thesis develop a **theoretical foundation, applicable to the design, optimization, and implementation of any complex system architecture design** (evolutive or not) across multiple technical fields. Doing "better with less" is critical due to tackle resource scarcity, address the need for design agility, and increase the adaptability to more complex system requirements beyond any heritage solution. Furthermore, it is also essential to address such objective with a holistic perspective, just like nature does, while making the most of current design and manufacturing techniques. Thus, this approach creates a foundation towards the infusion of upcoming automation methods, and other disruptive techniques regarding both design and implementation. In a **world that is challenged by increasing and changing stressors such as climate change, population growth, system complexity, and everlasting market pressures**, we deserve a more efficient way of **getting better and more holistic solutions, so we can keep daring mighty new challenges**.

**Keywords**: architecture, evolutive, evolution, evolutionary, systems engineering, hardware-based systems, systems architecture, system design, hardware-based systems, optimization, evolutive systems architecture, systems design

# Acknowledgments

*I dedicate this research, which has slowly matured over the last 25 years of personal and professional experience, to my father Ramon Polit Alabau and my mother Charo Casillas Martinez who always believed in me. 'Estoy siguiendo Papa… y sé que esto te hace sonreír desde allí.'*

*I dedicate the hard work behind this dissertation to my thesis advisor Emmanuel Caillaud, who guided me with a very smart smile along some of the darkest times I have lived through.*

*Many thanks to the University of Strasbourg for allowing this thesis to see the light during such complicated times. Thank you as well to Lew Soloway, Sean Jenkins, Bahman Chavoshan, and Nathan Strange for their comments and reviews.*

*The future is made by those who dare to dream 'impossibles' and never back out in their pursuit, making the present always brighter.*

*Feet on the ground and the head in the stars...*

*Raul Polit Casillas*
*Los Angeles, Malibu, Valencia, July 2020*

# Table of Content

# List of figures

# List of tables

# Nomenclature

| | |
|---|---|
| 3DP | Three-dimensional printing |
| AM | Additive manufacturing |
| AML | Algebraic Modeling Languages |
| ANSI | American National Standards Institute |
| AI | Artificial Intelligence |
| ABM | Agent-based modeling |
| ARR | Adaptability Reactivity & Regeneration |
| BIM | Building Information Modeling |
| BPM | Business Process Mapping |
| CAS | Complex Adaptive System |
| CHS | Complex Hardware-based Systems |
| CAD | Computer Aided Design |
| CAM | Computer Aided Manufacturing |
| CE | Concurrent Engineering |
| CES | Complex Engineered System |
| CI | Configuration Item |
| COSYSMO | Constructive Systems Engineering Cost Model |
| CMMI | Capability Maturity Model Integration |
| CML | Concept Maturation Level |
| CSA | Complex System Architecture |
| CCA | Cross-Consistency Assessments |
| DAU | Design-Artifact-User |
| DE | Differential Evolution |
| DE | Design Engineering |
| DIO | Design Implementation & Operations |
| DSE | Design System Engineering |
| eADQN | Evolutive Architecture Dynamic Questioning Network |
| eAMG | Evolutive Architecture Maturity Gaps |
| eAML | Evolutive Architecture Maturity Level |
| eGBSEL | Evolutive Geometry Behavior Substance Equipment List |
| eSAR | Evolutive System Architecture |
| eSARD | Evolutive System Architecture Design |
| eASG | Evolutive Architecture Seed Geometry |
| eASM | Evolutive Architecture Seed Model |
| EC | Evolutionary Computation |
| ED | Evolutionary Development |
| EO | Evolutionary Optimization |
| EP | Evolutionary Programming |
| EPH | Evolutive Portable Habitat |
| EIS | Enhanced Imaging System |
| ES | Evolutionary Strategy |
| ESD | Evolutive System Diagram |
| ESS | Evolutive Seed Sketch |
| ESM | Evolutionary System Model |
| ESE | Evolutionary Systems Engineering |
| EVP | Evolutionary Principle |

| | |
|---|---|
| FEA | Finite Element Analysis |
| FDD | Feature-driven Development |
| FoS | Family of Systems |
| FPDS | Family of Point-Design Solution |
| GA | Genetic Algorithms |
| GBS | Geometry Behavior Substance |
| GE | Grammatical Evolution |
| GHG | Green House Gas |
| HD | Hardware Design |
| HMS | Holonic Manufacturing System |
| IBM | International Business Machine |
| ICAS | Intelligent Complex Adaptive Systems |
| ICSM | Incremental Commitment Spiral Model. |
| ID | Intelligent Design |
| IDP | Integrated Product Development |
| IDPT | Integrated Product Development Team |
| IID | Incremental and Iterative Development |
| IMF | International Monetary Fund |
| INCOSE | International Council con Systems Engineering |
| $IP^2D^2$ | Integrated product and process design and development |
| ISO | International Organization for Standardization |
| JPL | Jet Propulsion Laboratory |
| MA | Memetic algorithm |
| MBSE | Model-based system engineering |
| MPM | The Munich Procedural Model |
| MIL-STD | Military Standard |
| NASA | National Aeronautics and Space Administration |
| NoP | Network of Problems |
| OMG | Object Management Group™ |
| PDS | Point-Design Solution |
| PIT | Product Integration Team |
| PSO | Particle Swarm Optimization |
| RFP | Request for Proposal |
| RIBA | Royal Institute of British Architects |
| SOA | State-of-the-art |
| SoS | System of Systems |
| SoSE | System of Systems Engineering |
| SDLC | System Development Life Cycle |
| SE | System Engineering |
| SEDS | System Engineering Detailed Schedule |
| SEFT | System Engineering Competency Framework |
| SEIT | System Engineering and Integration Team |
| SEMP | System Engineering Management Plan |
| SEMS | System Engineering Master Schedule |
| SIT | Systematic Inventive Thinking |
| TRL | Technology Readiness Level |
| TRIZ | Teoriya Resheniya Izobretatelskikh Zadatch (“theory of the resolution of invention-related tasks") |
| WBS | Work Breakdown Structure |

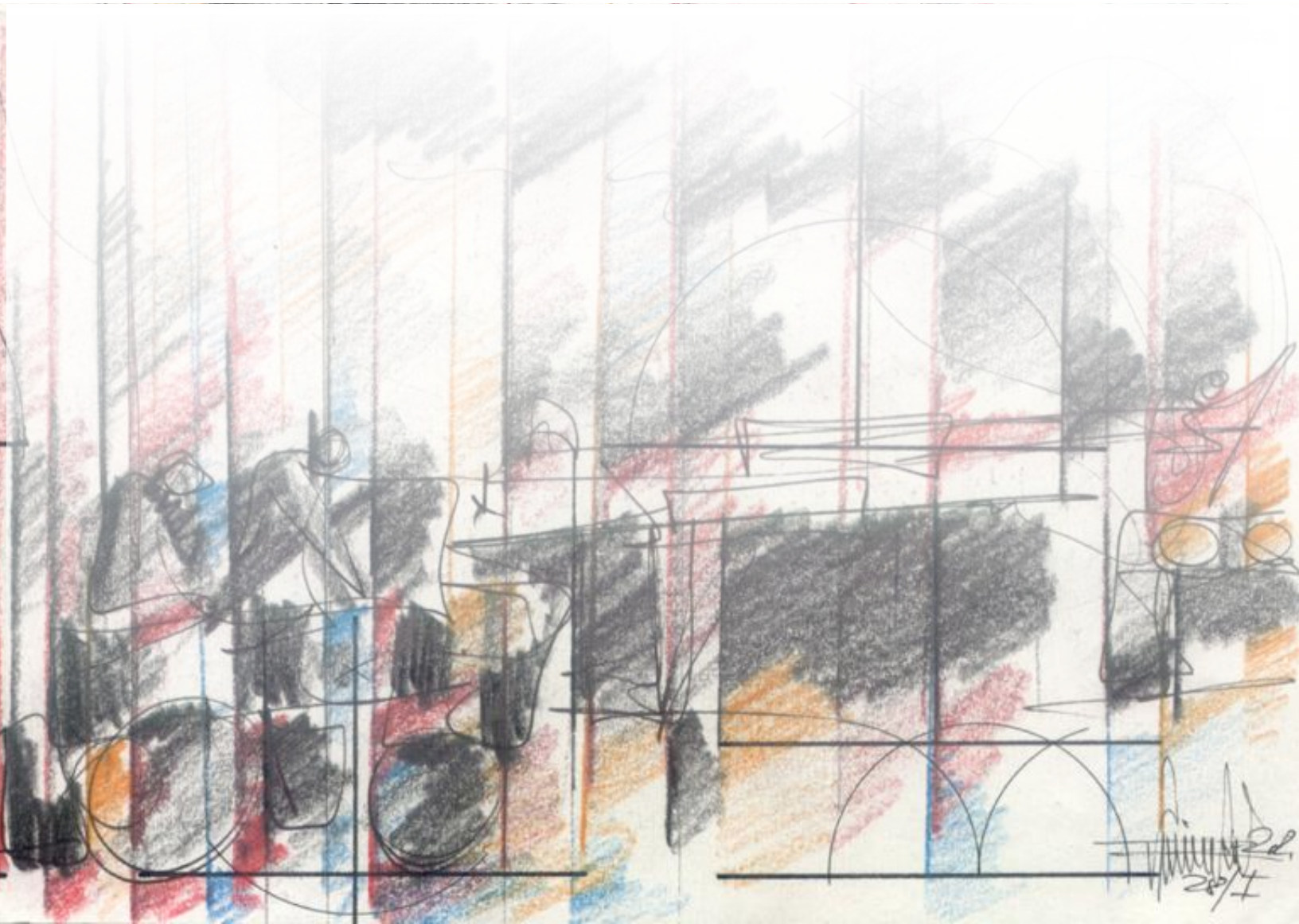
# EVOLUTIVE SYSTEMS ARCHITECURE

## Introduction and contributions

## CHAPTER 1

*“Un voyage de mille lieues commence toujours par un premier pas”.*

***Lao-Tzu***



# 1. Introduction

## 1.1. Motivation, Context, and Problem Statement

Nowadays the practice of multidisciplinary system design across technical fields is increasingly handling more complexity due to a growing number of global stressors such as resource scarcity, crosspollination drivers, workforce availability, and the influence of cultural and technical heritage, among many others.

This accelerating situation is especially relevant among hardware-based complex system architectures, since not only they are becoming a blend of hardware, software, data, and user interaction, but they are also demanding a many more new assets and capabilities in a world where the data-driven revolution is reaching our physical reality. Often these new systems do not have much relevant heritage, yet they aim towards challenges demanding much higher performance levels.

The overarching goal of this thesis is to provide a foundational design methodology to enable these fast-changing complex hardware-based systems (CHS). More specifically, the objective is to structure how to efficiently evolve from an [A] unadaptable, passive, and resource depleting system solution, to a [B] highly adaptable, reactive, and regenerative system architecture (Figure 1). This problem statement also implies the assessment of new contexts, practices, and system characteristics as key elements towards elaborating a novel, adaptable, and resilient system design approach. Based upon state-the-art techniques across multiple domains and inspired by nature, this system design research is also associated towards operational, implementation, and optimization workflows required in a system development process like this.

## 1.2. Research Questions

Upon such complex context, this thesis is organized around several intertwined research questions:

1. What new characteristics and complementary design needs do these ultra-complex systems present within resource-scarce environments?
2. What principles could enhance more traditional design and system engineering workflows to achieve faster, better, and more efficiently such multidisciplinary complex systems?
3. How could a design method that considers previous questions be used to develop more efficiently complex systems within such environment, when there is no direct heritage and ultra-system performance is a must?

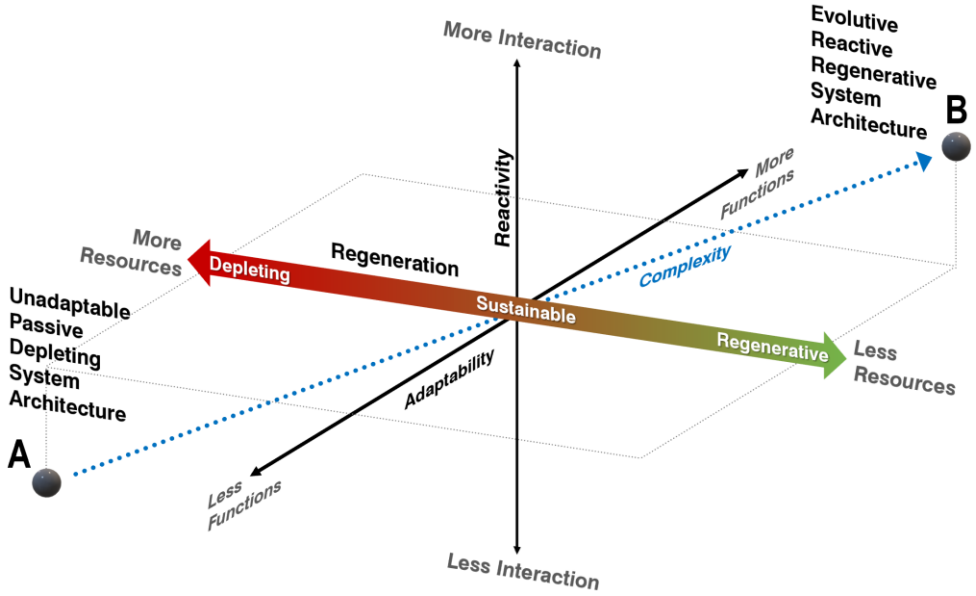



*Figure 1. Three-dimensional representation of evolutive design coordinates (adaptability, regeneration, and reactivity).*

### 1.2.1. General Research Approach

The associated research activity to tackle such questions is based on several key points such as:

- A thorough study of state-of-the-art techniques and gaps towards multidisciplinary design approaches.
- A comparative analysis complemented by an integrative approach, so it builds upon the gaps across techniques.
- An applied research and practice over two decades designing, leading, and managing complex systems design.

## 1.3. Delimitations and foundation

While the nature of this thesis is to address complex systems in general, and evolutive system architectures specifically as the following chapters elaborate, there are bounding conditions regarding scope, context, and applications:

- **Hardware-based systems** are the main study subject, although they can be enhanced by software, data, etc.
- **System architectures** range from top-level systems and assemblies to subsystems, components, and other assets.
- **Full cycle development** is the objective of this approach tackling design, implementation, and system operations.
- **Systems and design engineering** topics, workflows, and domains are studied and combined within this approach.

### 1.3.1. Literature Review Approach

Therefore, the strategy towards the selection of literature review topics includes the following areas.

- **Design engineering** techniques tackling geometry-driven design processes, from antiquity to today.
- **Systems engineering** workflows and methods handling the definition and development of large complex systems.
- **Evolutionary principles** coming from both biology studies, as well as pioneering software-driven applications.

## 1.4. Contributions

Research contributions of this doctoral dissertation are presented in chapter 7. They could be summarized as:

- A **thorough literature review and joint gap analysis** among design engineering (DE), systems engineering (SE), and evolutionary driven techniques.
- A **novel classification** for a complex system architectures subset driven by adaptability, regeneration, and reactivity.
- A new **evolutive system design and development method** tackling key aspects of such hardware-based systems.

## 1.5. Significance

The design of highly adaptable complex systems that can improve resource utilization and environmental interaction is at the core of many technical and creative areas in today's world. From shoe designs, to building improvements and data-driven consumer products, smart hardware-based systems become more complex and all associated design workflows need to cope with more complexity, and better performance at much faster speeds. In a world facing a growing scarcity of resources, such evolutive approach presents a novel and adaptable foundation from both academic and practice standpoints. It is a method based on disciplinary synergies that also integrates traditional and discrete DE/SE approaches.

## 1.6. Thesis outline

The layout of this doctoral dissertation follows the conducted research process through the following chapters.

1. **Introduction**: motivation, research questions, limitations, contributions, and significance.
2. **Context**: scarcity, complexity, performance, multidisciplinarity, agility, network, heritage, innovation, and culture.
3. **Literature review**: engineering design, systems engineering, evolutionary principles, gaps, and conclusions.
4. **Evolutive system architectures (eSAR)**: approach, keystones, drivers, and interconnections.
5. **Evolutive system architecture design method (eSARD)**: approach, characteristics, objectives, principles, helix design model, workflow, eADQNs, eAMGs, eASGs, eASMs, eAMLs, as well as metrics and conclusions.
6. **Study case**: evolutive portable habitat and deployment subsystem.
7. **Discussion, conclusion, and future steps.**

## 1.7. Domains and Perspectives

In this research several interconnected domains and perspectives are addressed from a multidisciplinary approach. These include both design engineering and systems engineering, as well as concepts coming from architecture, computer science, and biological evolution studies. This multidisciplinary and multifaceted approach highlights the standpoint of the global analysis in chapter 2, as well as the detailed literature review in chapter 3.

### 1.7.1. Design System Engineering

This thesis is developed within the intersection between design engineering (DE) and system engineering (SE), since the goal is to address the full design of complex systems from a holistic perspective. This intersection is defined as design system engineering or DSE (Faisander and Adcock, 2020) and it combines key aspects of both domains as follows:

- **Engineering design** (a.k.a. design engineering). This group of disciplines focusses on the conceptual, organizational, and configuration description of a geometry-driven system in general, and hardware-based systems in particular (Cross, 2008). This includes the exploration of the design space, as well as all required activities to define, describe, and design multiple aspects of the system such as logic, idiosyncrasy, configuration, organization, shape, volume, interfaces, mechanisms, assemblies, states, etc. Among some of the most relevant techniques, tools, and workflows within this disciplinary domain are the following: hand sketching, technical drawing, three-dimensional models, movement analysis, installation diagrams, flowcharts, CAD models, CAM schemes, etc.
- **Systems engineering**. Within this context, this domain of disciplines relates to the analytical definition, integration, and management of complex hardware-based systems (Buede, 2009) with emphasis on non-geometrical aspects throughout the system lifecycle. Among the many activities included in this category are the management of requirements, system description, parametric studies, risk assessment, project coordination, project management, process engineering, system optimization, project management, and many more (Badiru, 2019; Braha et al., 2006; Haberfellner et al., 2019; INCOSE, 2015; Liu, 2015; Long and Scott, 2011).

Both areas and their associated techniques will be studied in detail in chapter 3, however the approach used in this thesis integrates both sides from full system design, implementation, and operations standpoints. From a more detailed perspective some application domains include complex electro-mechanical systems, complex machines, product designs, process designs, robotic systems, cybernetic systems, architectural designs, and architectural buildings, among many more highly complex hardware-based systems.

### 1.7.2. Architectural Multidisciplinary Mindset

*Figure 2. Poulnaborne Dolmen, Ireland*

While architectural design is certainly not the only application domain of this research it provides an important mindset towards this approach. Architecture is one of the oldest disciplines of humankind (Benevolo, 1977). Starting with the first megalithic testimonies (Figure 2) the design, technology, and construction of dwellings, houses, and other representative buildings, has been the keystones of this field over millennia (Moffett et al., 2004). The ancient Greek etymology of the word architect comes from *ἀρχιτέκτων* (*arkhitéktōn*, "master builder"), which has its origins in *ἀρχός* (*arkhós*, "leader") or *ἀρχι-* (archi-, "chief"), and *τέκτων* (*téktōn*, "builder, mason") (Dejtiar, 2018; Retamosa, 2020). Thus, architecture is the discipline of the 'leading' worker, so it could be understood as the process of making a vision, which was often expressed graphically through a drawing, to come true. That also implies the facilitation of the project by connecting people, knowledge, resources, and technologies. Therefore, architecture since its beginning has been about creating synergies, which are multidisciplinary in nature. Such is the power of that subtle concept, that even in today's world of information technology and advanced computing, the term architecture (*architecting*) is used to describe a meta-level organizational approach and overarching design principles for any logical and digital activity beyond its physical origins (Maier, 2009).

In its more traditional form, the practice of architecture is about managing both complexity (e.g., cultural needs) and scarcity forces (e.g., gravity or materials), which has shaped the architecture standpoint over centuries (Roth, 1994) beyond the action of building.

However, as a mindset, its practice also provides a unique perspective leveraging and managing big-picture perspectives, while understanding enough of the details involved in its implementation (Figure 3, Polit-Casillas, 2008) so such vision could be buildable ('*firmitas*'), feasible ('*utilitas*'), and desirable ('*venustas*') beyond its heritage (Pollio, 2018). Hence, such mindset is at its best when creating and facilitating synergetic connections, as well as guiding its implementation. Such dual perspective is key towards this research. The practice of architecture historically has been about dealing with **complexity and entropy**, as its objective is to build human environments that address technical and cultural requirements. Thus, this discipline also presents a consequent mindset towards managing both **quantifiable and qualifiable** parameters.

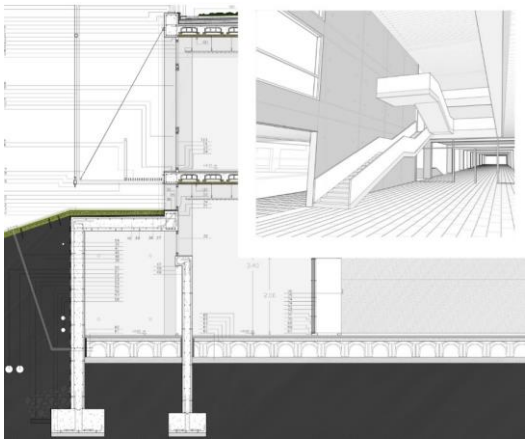


*Figure 3. Construction details and architecture vision on a sustainable building. (Polit-Casillas, 2008)*

Therefore, from a domain standpoint this design, management, and implementation perspective not only includes architectural constructions of any kind, but also complex hardware system assemblies requiring overarching facilitation.

### 1.7.3. Biological Natural Evolution

Finally, as section 3.3 will elaborate in detail, this thesis includes principles and perspectives coming from the domain of natural evolution and biological studies. The study of species, and the development of organisms, among other topics enabled over the years an understanding of multiple natural mechanism with application towards complex systems design. Techniques such as genetic algorithms and evolutionary programming among many more, are examples of this approach. As next chapters will elaborate in detail, such perspectives can influence new perspectives and they are an integrated part of the evolutive approach in this thesis. From this standpoint, other associated and non-biological domains of application include complex software design, project management, system optimization, and SE techniques, among many more.

### 1.7.4. Summary

Table 1 presents a summary of perspectives, application domains, and toolsets associated with this research.

| | Fields | Key Perspective | Toolsets | Application Domains | References |
|---|---|---|---|---|---|
| Evolutive Architecture and DSE | **Design Engineering (DE)** | Systems Design<br>Geometrical Design<br>Assembly integration | Drawing, CAD, schemes, BIM, diagrams, CAD, PLM, math, etc. | Electro-mechanical systems, complex machines, product designs, process designs, robotic systems, cybernetics, etc. | (Cross, 2008)<br>(Pahl et al., 2007)<br>(Ullman, 2009)<br>(Curedale, 2013) |
| | **Systems Engineering (SE)** | Systems design<br>Systems parametrics<br>Systems Integration<br>Project management<br>Risk Assessment | Diagrams, lists, documents, models, math, coding, scripts, etc. | Complex systems, enterprises, SoS, complex machines, interfaces, requirements, etc. | (Buede, 2009)<br>(Badiru, 2019)<br>(Liu, 2015)<br>(INCOSE, 2015) |
| | **Architecture Development (AD)** | Architecture design<br>Project management<br>Integrations and Mfg.<br>Visualizations | Drawings, diagrams, models, renders, CAD, BIM, documents, etc. | Buildings, structures, complex systems, virtual systems, civil engineering, etc. | (Dehlinger, 2009)<br>(Neufert & Neufert, 2000)<br>(Lawson, 2014)<br>(Roth, 1994) |
| | **Natural Evolution Studies** | Biological evolution<br>Systems engineering<br>Computer science<br>Software design<br>Hardware design | Models, diagrams, schemes, evolutive trees, genetics, coding, etc. | Biological studies, genetics, evo-devo studies, eco-evo-devo studies, software development, algorithm development, AI, etc. | (Zeiger, 2009)<br>(Chen & Han, 2002)<br>(Bentley, 1999)<br>(Gros, 2015) |

*Table 1. Integrated perspectives and domains within evolutive architecture research.*

## 1.8. Definitions

In the context of this research there are a few definitions requiring clarification since they bound the scope, application, and domain of this dissertation. This is key since it is applicable to many industrial and technical sectors. The following sections explore common definitions across literature, while highlighting which definition is used within this thesis.

### 1.8.1. Architecture

Table 3 summarizes multiple definitions of *architecture* across fields, categories, domains, and applications, among the thousands of them available. Since the concept is well distributed only the most relevant domains are presented.

| Id | Source / Category | Definition | *Domain* | *References* |
|---|---|---|---|---|
| **AD1** | **General** | Epistemologically architecture [1] techne, [2] mechanical art, [3] design, and [4] fine art after Kristeller's studies. | Historical | (Parcell, 2012) |
| **AD2** | **Construction** | "is both the process and the product of planning, designing, and constructing buildings or other structures" | AD | (Oxford University Press, 2003) |
| **AD3** | **Functional / Logical** | "…The functional architecture of a system contains a hierarchical model of the functions performed by the system, the system's components, and the system's configuration items…" | SE | (Buede, 2009) |
| **AD4** | **Software** | Software architecture is the structure behind the creation of a system including software elements, relationships, and their characteristics.<br>"…Software architecture is the conceptual glue that holds every phase of the project together for its many stakeholders…" | SE / Computer Science | (Clements, 2011)<br>(Carnergi Mellon, 2017) |
| **AD5** | **Hardware** | It includes all physical components, relationships among them, and their characteristics regarding machines, devices, components, etc. | SE/DE Computers | (Yadin, 2016)<br>(INCOSE, 2015) |

*Table 2. Architecture definitions.*

**Architecture** is understood in this research as the overarching and organizational design principle of a complex system and all its parts beyond 'the inevitable art of the human activity framework' (Roth, 1994) such as buildings, dwelling, art pieces, etc. Therefore, this concept refers to the highest system level of a complex subsystem-based artifact (hardware and software), including its representation and relationships among components [Def01].

### 1.8.2. System

There are also many definitions of *system* in the literature across domains and fields as Table 3 presents:

| Id | Source / Category | Definition | *Domain* | *References* |
|---|---|---|---|---|
| **S1** | **General** | "a regularly interacting or interdependent group of items forming a unified whole, such as: [1] a group of interacting bodies under the influence of related forces, [2] an assemblage of substances that is in or tends to equilibrium, [3] a group of body organs that together perform one or more vital functions, [4] the body considered as a functional unit" and a "group of devices or artificial objects or an organization forming a network especially for distributing something or serving a common purpose" | All | (Merriam Webster, 2020) |
| **S2** | **General** | ''a collection of hardware, software, people, facilities, and procedures organized to accomplish some common objectives.'' | SE | (Buede, 2009) |
| **S3** | **NASA** | "A *system* is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce system-level results". The value of the system is in the synergy beyond its parts and the "big picture" perspective. | SE Aerospace | (NASA, 2007) |
| **S4** | **Badiru** | "A system is represented as consisting of multiple parts, all working together for a common purpose or goal. Systems can be small or large, simple, or complex. Small devices can also be considered systems. Systems have inputs, processes, and outputs." | SE | (Badiru, 2019) |
| **S5** | **Winner** | "A group of elements which are relevant (and not merely useful) for achieving a purpose, which interact with each other, and which have a structure within predefined boundaries." | Automotive | (Winner, 2013) |
| **S6** | **Liu** | "A system can be broadly defined as a set of integrated components that interact with each other and depend upon each other, to achieve a complex function together. A system can be decomposed into smaller subsystems or components and a system may be one of the components for a larger system." | SE | (Liu, 2015) |
| **S7** | **Wasson** | "System. An integrated set of interoperable elements, each with explicit specified and | SA | (Wasson, 2005) |

| | | | | |
|---|---|---|---|---|
| | | bounded capabilities, working synergistically to perform value-added processing to enable a User to satisfy mission-oriented operational needs in a prescribed operating environment with a specified outcome and probability of success." | | |
| **S8** | **ISO/IEC/IEEE 15288** | "Systems […] are man-made, created to provide products or services in defined environments for the benefit of users and other stakeholders." | SE | (INCOSE, 2015) |
| **S9** | **INCOSE** | System is "[…] an integrated set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services and other support elements." | SE | (INCOSE, 2015) |

*Table 3. Definitions of system.*

Thus, the modern notion of system presents the concept of a whole made of components that support a function. This conception of system involves an input-process-output model (Badiru, 2019) that also considers controls and enablers. Any system also presents the following characteristics (Buede, 2009; INCOSE, 2015):

- Operating environment
- Environmental interactions: external & internal
- System boundary
- Life cycle and phases
- Main function or purpose
- Hierarchical structure
- Reliability as a whole
- Interacting subsystems and components, with attributes that regulate the behavior of the system through their multiple interrelations (Badiru, 2019).

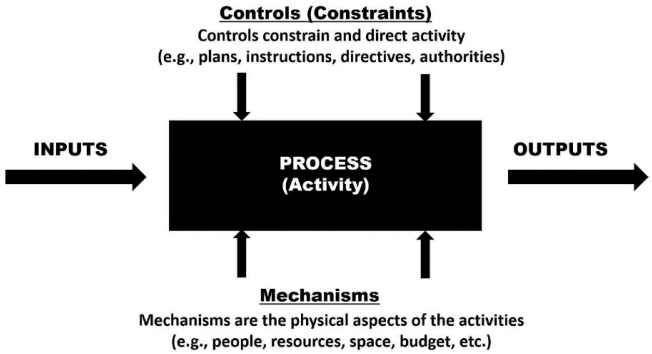



*Figure 4. Input-process-output model of a system (Badiru, 2019).*

Systems could be natural, man-made, static, dynamic, adaptive, evolutionary, conceptual, physical, open, and closed (Liu, 2015). Based on the literature review, and under the context of this thesis regarding hardware-based developments, the definition for system is based on the INCOSE approach with a small variation due to the nature of evolutive architectures, as they will be presented later (chapter 4). The definition within this research is the following:

**System** "is a synergetic, multidisciplinary, integrated and evolvable set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, construction, services, and other support elements." From this thesis standpoint, any system presents geometry, behavior, and substance, and it is defined by quantifiable and qualifiable parameters [Def02].

### 1.8.3. System Architecture

Another term that is broadly used in this research is *system architecture*. This one also presents many definitions on the technical literature across fields and application domains as Table 4 presents:

| Id | Source / Category | Definition | *Domain* | *References* |
|---|---|---|---|---|
| **SA1** | **General** | "System element architecture is defined by two entities: [1] *System of interest* (SOI) which is comprised of the mission system and the support system, and [2] *operating environment.:* | SE | (Wasson, 2005) |
| **SA2** | **INCOSE** | System architecture is "[...] the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and the in the principles of its design and evolution." | SE | (INCOSE, 2015) |
| **SA3** | **Hardware + Software** | "[…] and ensemble of elements (ultimately hardware and software components) that collaborate to fulfill defined requirements allocated to a node or systems (implying that a clear system boundary and user interfaces are defined) […]" | DSE | (Borky & Bradley, 2018) |
| **SA4** | **Systems Engineering** | "[…] defines a comprehensive solution based on principles, concepts, and properties logically related to and consistent with each other." "…System Architecture is abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. It also focuses on high-level structure in systems and system elements. It addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest. […]" | SE | (Faisandier et al., 2020) |

| | | | | |
|---|---|---|---|---|
| **SA5** | **Complex systems** | "[…] an abstract description of the entities of a system and the relationships between those entities, […] represented as a set of decisions […]" | DSE | (Crawley et al., 2016) |
| **SA6** | **Model-based SE** | '[…] to express the strong relation and dependency between the system and its systems architecture, a composition association describes this relation. [..] it is the functional architecture, [..] depending on a number of principles regarding its organization, the design, and the system's evolution, […], interactions within the systems and its context […]" | MBSE | (Weilkiens et al., 2015) |
| **SA7** | **Software Systems** | "[…] the rationale to ensure that the architecture's components, connections, and constraints define a system that will satisfy a set of defined stakeholder needs for the system." | Software | (Gacek et al., 1995) |
| **SA8** | **NASA** | System architecture includes systems models, behaviors in those models, system components, interfaces, as well as technical budgets. This term refers to system requirements, operations, and other artifacts within the global system. | Complex Hardware | (NASA, 2015) |

*Table 4. System Architecture definition*

**System architecture**, in the context of this thesis is the concept model defining logic, purpose, geometry, structure, behavior, material, aesthetic, and cultural properties of a system or group of systems with independence of its field of application. This definition applies to both hardware, software, and hybrid systems. This principle is based on general SE, INCOSE, and IEEE principles (Faisandier et al., 2020) [Def03].

## 1.8.4. Systems of Systems

Table 5 captures some definitions regarding *system of systems* (SoS).

| Id | Source | Definition | *Domain* | *References* |
|---|---|---|---|---|
| **SoS1** | **INCOSE** | "A system of systems (SoS) is a system of interest (SOI) whose elements are managerially and-or operationally independent systems." | SE | (INCOSE, 2015) |
| **SoS2** | **DoD** | "A set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities." | SE | (Kossiakoff et al., 2020) |

*Table 5. System of Systems (SoS) definitions*

An **SoS** is a system in itself, and it is defined by multiple challenges (INCOSE, 2015), such as: authorities, leadership, constituent system perspectives, capabilities, requirements, autonomy, interdependencies, emergence, testing, validation, and learning constraints, as well as its design principles [Def04].

## 1.8.5. Systems Engineering

There are indeed multiple definitions of systems engineering across the literature, therefore only a few have been selected as reference because of the relationships among the approach of this research and other domains.

| Id | Source | Definition | *Domain* | *References* |
|---|---|---|---|---|
| **SE1** | **INCOSE** | "[…] Systems engineering is a discipline that concentrates on the design and application of the whole (system) as distinct from the parts. It involves looking at a problem in its entirety, taking into account all the facets and all the variables and relating the social to the technical aspect. (FAA 2008) […]" | SE | (INCOSE, 2015) (Valencia et al., 2011) |
| **SE2** | **Liu** | "[…] it is an applied science […] concerned with the big picture of the system: it is a top-down design processing […] starting with the needs from user/stakeholder expressed in the format of requirements. […] is a multidisciplinary field with four categories: [1] art and science domain, [2] engineering domain, [3] management domain, and [4] supporting roles […]" | SE | (Liu, 2015) |
| **SE3** | **Buede** | "[…] engineering discipline that develops, matches, and trades off requirements, functions, and alternate system resources to achieve a cost-effective, life-cycle-balanced product based upon the needs of stakeholders. " | DSE | (Buede, 2009) |

*Table 6. Systems engineering definitions across technical documentation.*

**Systems engineering**, based on the INCOSE definition (INCOSE, 2020a), is understood as the interdisciplinary 'approach to enable the successful design', implementation, 'use and retirement of an engineered system, using systems principles and concepts, and scientific, technological, and management methods' [Def05].

## 1.8.6. Design Engineering

Usually, engineering design refers in literature to the design process, while design engineering refers to more

aesthetical concepts. In the context of this research *engineering design* and *design engineering* are considered synonyms like in many other publications. Among some of the definitions considered are the following:

| Id | Source | Definition | *Domain* | *References* |
|---|---|---|---|---|
| **DE1** | **Dieter** | "[…] to design is to pull together something new or to arrange existing things in a new way to satisfy a recognized need of society […] this means synthesis, design, problem decomposition, and analysis. […] The four C's of design include: creativity, complexity, choice, and compromise." | DE Process | (Dieter and Schmidt, 2012) |
| **DE2** | **Engineering Design** | "Designing in engineering has the purpose of creating future operating artifacts (TS) and the operational processes (TP) for which they can be used, to satisfy the needs of customers, stakeholders, and user. These artifacts can actively we operative or be operated as tool by a human. […] design engineers explore alternatives solutions, and delivers proposals for appearance and present, manufacturing specifications for a design […]"<br>"[…] Design engineering is progress towards designing an object or process that fulfills a purpose, and that includes a substantial engineering content. [...]" | DE | (Eder and Hosnedl, 2010) (Samuel and Weir, 1999) |
| **DE3** | **Engineering** | "[…] the discipline, art, and profession of acquiring and applying scientific, mathematical, economic, social, and practical knowledge to design and build structures, machines, devices, systems, materials and processes that safely realize solutions to the needs of society […]" | DSE | (Childs, 2013) |
| **DE4** | **Design process** | "[…] the design process, then, is the organization and management of people and the information they develop in the evolution of a product. " | DSE | (Ullman, 2010) |

*Table 7. Design engineering definitions across technical literature*

Therefore, **design engineering** (Cross, 2008) within this dissertation is considered as the holistic process of [1] understanding and decomposing need and requirements, [2] synthesis a solution (often also geometrical), [3] analysis of alternatives, and [4] visualization of results through development of products, services, and systems in general, and hardware-based system in particular [Def06].

### 1.8.7. Systems Design

Regarding the approach within this dissertation, the concept of **systems design** is understood as the action and methodology of designing and conceptualizing a system to fulfill specific as well as open requirements in the context of complex architectures. 'Design definition is the process of developing, expressing, documenting, and communicating the realization of the system through a complete set of design characteristics described in a form suitable for implementation.' (Faisander and Adcock, 2020) [Def07].

### 1.8.8. Other Explicit Definitions

There are other terms used across this dissertation and its associated research, coming from other sectors and domains that are less common within DE and SE contexts. The following table summarizes them.

| Id | Concept | Definition | *Uses & Domains* | *References* |
|---|---|---|---|---|
| Def08 | **System Maturation** | It is the set of processes and actions by which a system becomes more mature including geometrical, behavioral, and substance through time, detailing, and growth. | DSE process, biology studies, system dev. | (Ullman, 2010) (Oxford University Press, 2020) |
| Def09 | **Adaptability** | This refers to the capability of a system and its associated processes to adapt to environmental, design, operative, and implementation changes. It is one of the ARR evolutive overarching characteristics. See section 4.2.1 for details. | DSE, DE, biology, cognitive studies | (Conrad, 2012) (Burke et al., 2006) |
| Def10 | **Adaptive** | It refers to a system that participates and integrates adaptability principles throughout its design and implementation. A system based on adaptation principles. | DSE, SE, DE, biology, control & information | (Dieckmann et al., 2004) (Mareels & Polderman, 1996) |
| Def11 | **Reactivity** | This refers to the capability of a system and associated development processes to interact with its environment and among its subsystems. It is one of the ARR evolutive overarching characteristics. See section 4.2.2 for details. | DSE, SE, DE | (Barnard et al., 2000) (Fox, 2016) |
| Def12 | **Regeneration** | This refers to the capability of a system and its associated development processes to use, manage, reuse, recycle, and replenish resources require for its development, operations, and operations. This also includes physical, energy, | DSE, SE, DE, biology, architecture | (Lyle, 1996) (Mang et al., 2016) |

| | | | | |
|---|---|---|---|---|
| | | workforce, virtual, and digital resources, among others. It is one of the ARR evolutive overarching characteristics. See section 4.2.3 for details. | | |
| Def13 | **Evolutionary System** | "[…] are entities, described as systems, which have been generated within the framework of an evolutionary process." | DSE, SE, biology, computer science | (Vijver et al., 2013a). |
| Def14 | **Evolutive Architecture** | Evolutive architectures are considered those system architectures designed, optimized, and implemented using applied evolutionary methodologies towards their concept development, system engineering, physical and logical optimization, physical manufacturing, and construction among other lifecycle key phases. These characteristics are independent from the field of application and they could be present one, several or all phases of the development cycle. See chapter 4 for more details. | DSE, SE, DE | (Hingston et al., 2008) (Charlesworth and Charlesworth, 2017) |
| Def15 | **Design** | Within this research this refers to development processes and outcome activity to describe, manage, communicate, and implement a system as well as all relationships among its components. See section 4.3.1 for more details. | DSE, SE, DE | (Samuel and Weir, 1999) (Asimov, 1976) (Roth, 1994) |
| Def16 | **Operations** | This concept related to the analysis, definition, description, and management of the system functional behavior, as well as other associated processes such as development and implementation. This includes workforce and knowledge too. See section 4.3.2 for more details. | DSE, SE, DE | (NASA, 2007) (Mahadevan, 2010) |
| Def17 | **Implementation** | This concept is used in this dissertation to capture processes, materials, and other resources required to turn a design into a reality regardless of its physical, logical, digital, and virtual nature. See section 4.3.3 for more details and links among concepts. | DSE, SE, DE | (Farid and Suh, 2016) (Gilmore, 2014) |
| Def18 | **Geometry** | This related to all 'properties and relations of geometrical elements such as points, lines, surfaces, solid, and higher dimensional analogs' describing the shape or configuration of a system over time, as well other logical, structural, material, and interface considerations regarding systems and components. See section 5.3.1 for details. | DSE, SE, DE | (Merriam Webster, 2020b) (Elam, 2001) (Kimura, 2001) |
| Def19 | **Behavior** | This is "the functional and behavioral range of anticipated actions describing how the system will be operated under all possible use-case scenarios". This includes both system architecture and associated development process. See section 5.3.2 and 4.3.2 for more details. | DSE, SE, DE | (NASA, 2007) |
| Def20 | **Substance** | This refers to chemical, mechanical, physical, and biological properties, characteristics, and processes involved in the physical implementation of a system architecture. This also includes fabrication, manufacturing, construction, and resource management, as well as the materialization of parts and subcomponents. See section 5.3.3 for more details and links among concepts. | DSE, SE, DE | (Sass, 2011) |

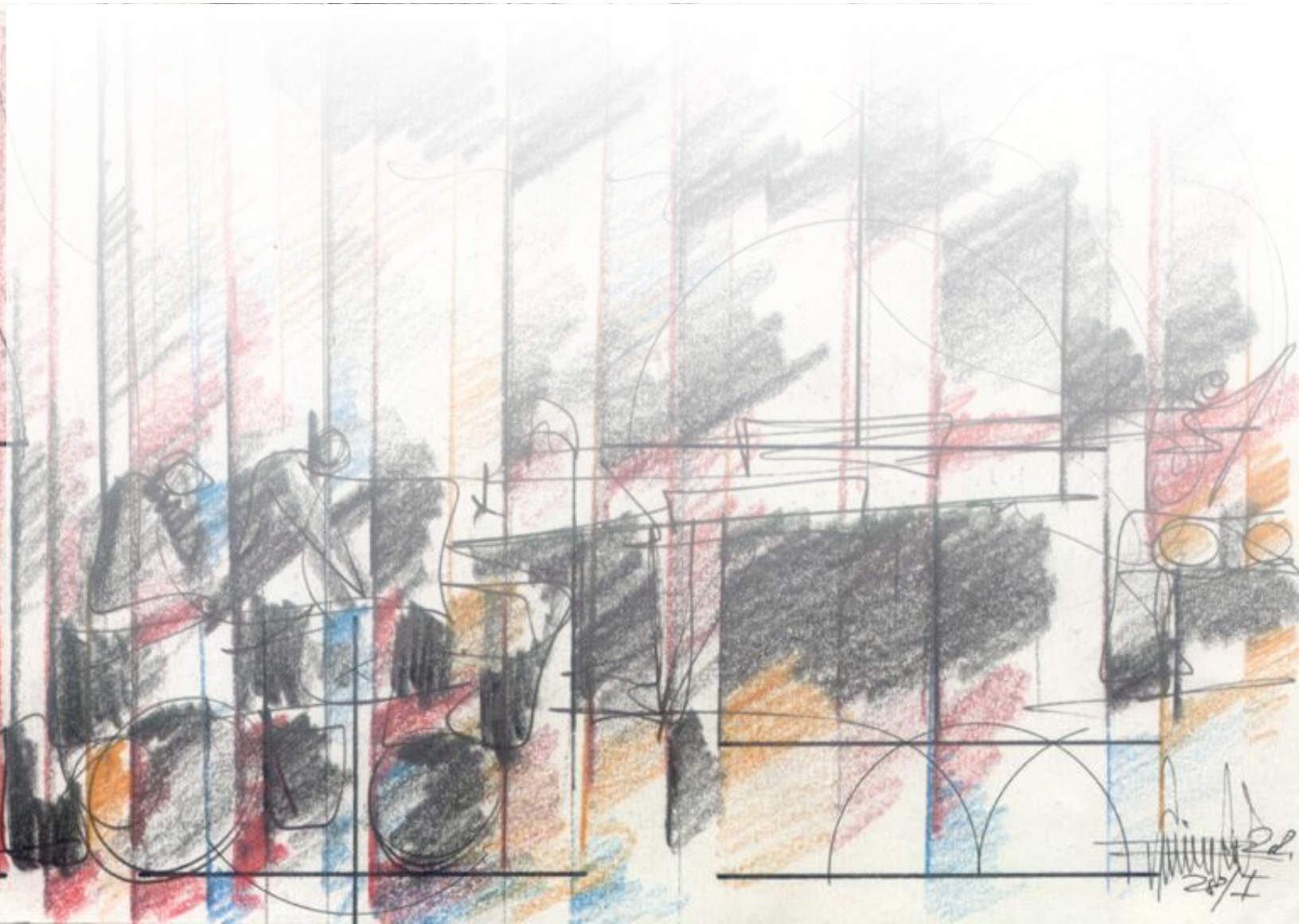
*Table 8. Definitions, uses, domains, and references.*

# SCARCITY, COMPLEXITY, AGILITY, & HERITAGE

## Evolutive Architecture Context

## CHAPTER 2

*"It is not they don't see the solution.*
*It is that they do not see the problem".*
***G.K. Chesterton***



# 2. Context of Scarcity: Needs and Resources

As a species we have been led by the need to do more. The urge to find, discover, or reach new levels has driven innovation through millennia, often tackling what and how we do things. We have even classified periods of our history (e.g. bronze age) by the design and manufacturing capabilities we developed (Harari, 2018). Survival needs, competitive advantage, and intellectual curiosity, among others have incentivized us to push all limits, often against cultural preconceptions, personal fears, and the everlasting hassle of finding enough resources to start such new ventures.


*Figure 5. Evolution in the type of information being transmitted by communication devices from 19th century to the 2020s.*

During the second decade of this century, the need to deal with increasing levels of complexity in terms of design, implementation, and management still keeps growing (Kravtsov and Kadtke, 1996), because we keep demanding more of our architectures independently of the field of application. A good example of this is the evolution in capability, and therefore complexity, of communication systems during the last century (see Figure 5). In the context of this research, as section 1.8.1 presented, the term architecture is referring to the highest system level definition of a complex subsystem-based artifact (hardware, software, or both). For instance, a building, a car, and an electromechanical consumer product are good examples of generic architectures among many others. Furthermore, data shows we are facing an upcoming phase in terms of resources availability due to climate uncertainty (WMO, 2020) and population growth (United Nations et al., 2019), while an increasing number of new and disruptive technologies (Buchholz et al., 2020) could become key in tackling such challenges. Hence, the balance between 'what we need' (requirements) and 'what we can do' (resources / capabilities), becomes an open field for exploration nowadays. This lands itself very well into a new paradigm for how we design and implement complex systems. Thus, addressing the relationship between needs (requirements) and resources (constraints), through the perspective of the offer-demand theory (Sloman et al., 2018) allows us to understand the complexity of a system architecture development (Figure 6) as a balance between those forces. The more needs or requirements are covered with less resources, the more efficient a system becomes. So, the slope of that curve could be understood as the complexity of such architecture. Following an efficient complexity curve (blue line) is often complicated, since economical, workforce, cultural, and technical constraints tend to flatten such curve. As a result, similar needs could be covered with a more efficient use of resources. However, providing a leap in that efficiency is often only possible through systematic new paths or disruptive technologies. Enabling and structuring such leap is the target of an evolutive design process developed in this thesis.

Regardless the field of work (e.g., architecture, car manufacturing, finance, product design, and medicine, etc.) the need to go beyond in terms of performance, novelty, efficiency, uniqueness, and adaptability is becoming a major force in any complex technical design endeavor. In today's context of design engineering and system engineering practices multiple drivers influence this balance of forces, and they will be studied in following sections. These factors are the foundation for new approaches towards both complex architecture systems and their associated design methods.

The consequences of both resource scarcity and the human drive to go beyond, push new design efforts and methods to be able to do more with less (Radjou and Prabhu, 2014), forcing in essence such effort towards the top left of the needs-resources curve presented in Figure 6. Markets, customers, and requirements demand more of any system architecture, affecting 'what' they are as a system, and 'how' they are developed as a method. That pressure ripples through all the multiple development phases in the lifecycle of a design development such as design, optimization, prototyping, implementation, management, and sustainability (Pahl et al., 2007).

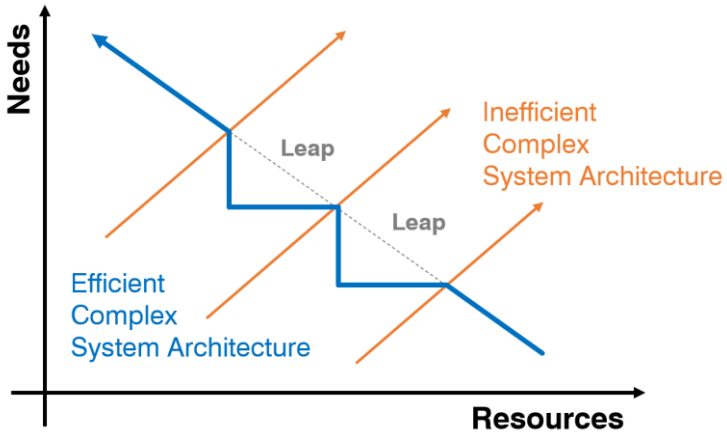



*Figure 6. Performance efficiency of system architectures upon the needs versus resources balance.*

Across time, tendencies, and practices there are, and have been multiple ways to tackle that balance between forces represented in Figure 6. While section 3 will present an elaborated and detailed literature review across the areas of design engineering, system engineering, and evolutionary principles, it is worth mentioning some overall approaches to clarify the context of this research. The balance between requirements and resource utilization could be understood also as the balance between technology, user, complexity, and cost. From that perspective we can identify several tendencies such as

- *Low-tech* that approaches complex problems with design and simple technology (Hirsch-Kreinsen and Jacobson, 2008), addressing the lack of resources as well as associated human needs (Philippe, 2020). This tendency had a big influence in the 70s with the do-it-yourself (DIY) approach (Wolf and Mcquitty, 2011). Under this trend we could also include small-tech, no-tech, slow-tech, and passive design, among other variations of related principles.
- *Social Design* tackles complexity with social and human needs at the center of such process (Margolin and Margolin, 2002). The use and level of technology is not as important as the responsibility behind it. This approach has applications across the board, affecting complex fields such as urbanism and architecture (Michael and Lin, 2018).
- *Frugal design* as described in this intro is about doing more with less (Radjou and Prabhu, 2014) while bringing the notion of control and equity within that balance through innovation (Micaëlli et al., 2016). The key is the efficiency of the approach, and it is applicable to different levels of technology as well as types of balances.
- *High-technology, deep-tech, or frontier-tech,* on the contrary answer such critical battles by relying heavily on cutting-edge technical solutions that may not necessarily reflect other social and innovation aspects (Steenhuis and Bruijn, 2006). The influence of this perspective could be seen in architecture solutions (Macdonald, 2019), information technology (Cortright and Mayer, 2001) project, AI new developments (Malach-Pines and Özbilgin, 2010), etc.

Nevertheless, this research embraces this full spectrum with a broad perspective by being agnostic of the technology level used in such balance, as well as other aspects such as innovation or social approaches. The objective though is to address the capability of the system first, and the subsequent design method after. Any other area of this activity affecting the complexity of such challenge could and should be addressed regardless the posture that was taken. In other words, from a truly broad perspective this is about getting the best and most optimized system result.

Thus, the principal focus of this research is about how we could achieve more efficiently, better system performances and capabilities when developing new complex systems that have no previous heritage. Furthermore, the inherent scarcity of resources also highlights the needs of a design and system engineering approach adaptable enough towards short and long-term changes regarding requirements, constraints, methods, etc. In other words, how we could design faster and smarter by doing not only more but better with less is the objective of this theoretical framework (validated over years of practical experience across industries). Tackling key stressors that affect the balance in the development of any complex architecture is always needed, but it is especially relevant under the foreseeable scarcity conditions affecting the technical practice of architecture and engineering in the coming decades. Thus, next sections tackle multiple context design stressors.

## 2.1. Resource Scarcity

The availability of all required resources to design, implement, and operate a complex system architecture could be stressed by multiple external and internal factors withing such development process. As Figure 6 showed, highly efficient architectures (and processes) respond to higher needs or requirements using less resources, while low efficient ones do the opposite. The objective is to increase the system performance for a given complexity, while at the same time enabling an adaptable methodology that aims to enable performance quantum leaps onto such systems for a given level of heritage and scarcity. The need-resource curve in Figure 7 shows those performance quantum jumps, showing a system architecture that responds to many more needs with less resources. These jumps are the ultimate efficiency goal for an evolutive design process. However, the efficiency of a system design method responds to a series of key design resources including among others knowledge, workforce, computing power, materials, energy, etc. (Pahl et al., 2007). In the beginning of the 21st century there are several areas contributing to a potential resource scarcity capable of significantly stressing current and future design context conditions. These stressors displace the curve to the left (Figure 7), so less needs can be addressed for a similar architecture approach. They become a major contributor to be addressed towards more efficient design methods.

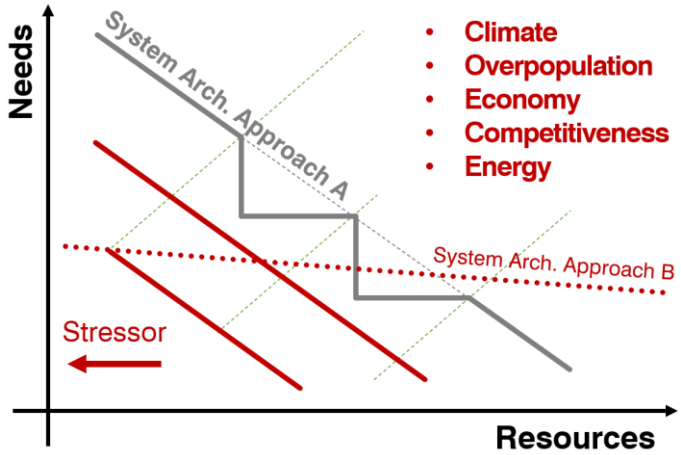



*Figure 7. Stressors displace the architecture efficiency on the need-vs-resource graph.*

### 2.1.1. Climate uncertainty

Climate change is a very complex and extensive subject beyond the scope of this research. However, the potential consequences of this global phenomena (Masson-Delmotte et al., 2019) must be considered as an overarching constraint forcing the need of new design approaches affecting communities (Boswell et al., 2019), policies (Harvey et al., 2018), and multiple industries such as architecture (Smith, 2006) among many others which are very resource dependent. As a simplified approach we could state that since mid-20th century, scientists have been warning of the consequences of an unsustainable development, leading to an increase in the release of CO2 and other greenhouse gasses, as well as the subsequent global warming (Siegmund et al., 2019). Such delta in planetary temperature levels presents many potential related effects such as increased ice melting rates, lack of planetary surface albedo, water sea level rising, changes in energy distribution through air and water movements because of changes in water salinity, temperature, etc. (Robertson et al., 2018). This not only affects local climate conditions, but also plants, animal species, and of course human activity.

Dwelling is a major area of human activity affected by climate change. It also affects climate change and is directly related to the continuous increase in human population globally. These reasons highlight the need of new design and construction paradigms, while they showcase the type of influence these stressors could present. Current estimations by the UN present a likely increase of 3 billion people by 2050 if current conditions are maintained (United Nations et al., 2019), with projections above the 10 billion people beyond 2100 (Roser, 2013) as Figure 9 shows. Assuming an average number of 1.5 people per home, and a simplified reduction of 25% due to existing houses, this means around 1.5 billion new homes

would need to be built between 2020 and 2050 (Smith, 2018). In other words, a city center like Paris of two million people should be designed, built, and delivered every week including houses, schools, streets, hospitals, etc. during that period. Buildings today consume 40% of all generated energy worldwide (EIA, 2019) and up to 50% of all solid waste worldwide is construction materials (Kaza et al., 2018). Therefore, this represents a major challenge with huge and unsolved implications when it comes to resource availability, workforce, construction technology, design methods, and energy efficiency affecting directly or indirectly all design and development efforts due to its magnitude.

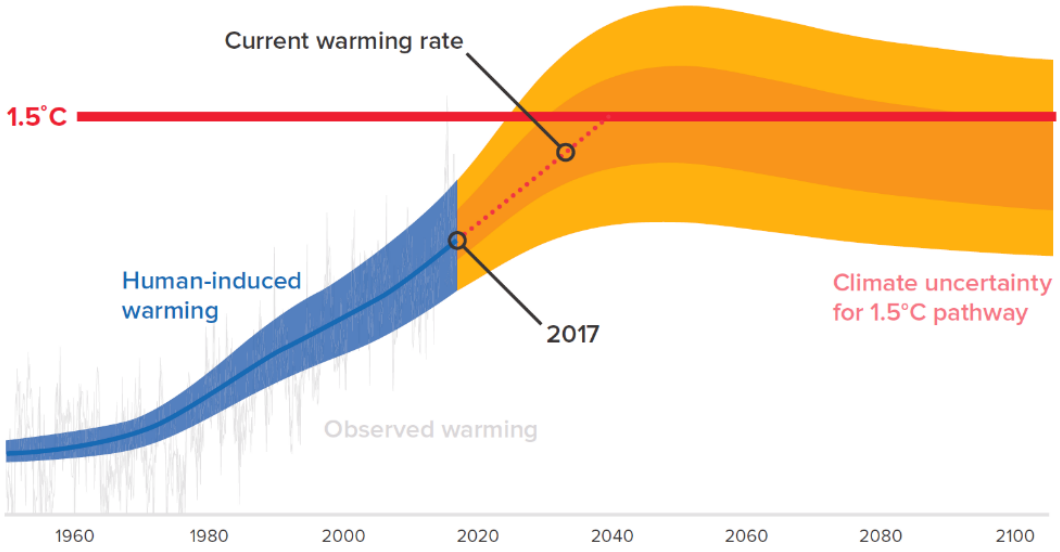



*Figure 8. Global warming forecast. IPCC Special Report on Global Warming (Masson-Delmotte.V., et al., IPCC, 2019)*

## 2.1.2. Population growth

This challenge is even bigger, especially when likely dislocations due to climate change could lead up to 1 billion climate migrants by 2050 (International Organization for Migration, 2014). Furthermore, up to 1.6 billion people did not have a proper shelter already in 2015 (UN, 2019). The use and construction of dwellings represents a basic human economic activity with a value around US$280.8 trillion in 2017 (Barnes, 2018). This indirectly relates to design and implementation activities of home equipment devices and energy production systems, among many more markets and industrial sectors.

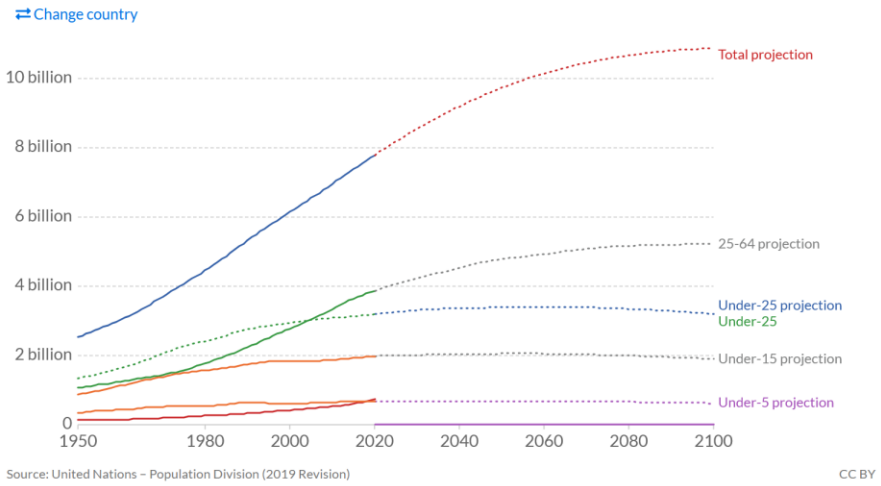



*Figure 9. Population by age bracket (UN Projections), After Our World in Data (Roser, 2013).*

Constraints driven by both climate uncertainty and world population growth combined are increasingly bringing new levels of resource scarcity affecting populations, ecosystems, economics, and institutions (Evans, 2010). This growing trend affects consequently all industrial, agriculture, and technology fields worldwide in different ways, while it emphasizes the need for quantum leaps in terms of system efficiency as a possible response. Architecture and construction specifically, and any complex system architecture in general (e.g., cars, trains, solar farms, computers systems, etc.) would require a new design and implementation approach if current levels of comfort and capability must be addressed under an increasing population. Consequently, these constraints will force and spark new approaches and needs in architecture and engineering practices dealing with complex systems that this thesis takes as one starting point to address new methodologies.

### 2.1.3. Economic Constraints

The combined effect of climate change and population growth could impose the need to reduce the number of resources used in any system (Figure 10) at a global scale, if no new technologies and concepts are introduced since Earth is for that purpose and simplifying a close-loop system (Müller, 2017). Within an economic system where services and products are rendered for currency, this brings the need to assess new cost reduction approaches (less resources) and viable risk strategies (more resilience) at both system design and design process levels (Brown, 2013). As Figure 10 shows, to achieve certain needs (e.g., solar power for a dwelling) an architecture uses certain resources (e.g., number of solar panels, transport, deployment, installation, etc.) The more complex the system is (e.g., F1 car, airplane, power plant, etc.), the bigger portion of the resources used in the developing phase lies within the design process itself (e.g., workforce hours). This is especially relevant if the process is set up with a cost committed approach [Ullman's manufacturing and cost cited in (Jack, 2013) and (Kihlander, 2009)]. However, this could be very constraining when unique solutions, prototypes, or small series are the objective since the initial design and testing cost cannot be shared or distributed among multiple users or product sales. Thus, the more complex and unique a system architecture becomes, the higher is the relative cost to create a single one in terms of design and validation towards its implementation (fabrication, construction, and manufacturing) (Larson, 2010). This is reinforced by the lack of an economy of scale factor, as well as a heritage of directly applicable solutions. Therefore, a reduction in both implicit and explicit costs to create new system architectures that could provide better performance levels beyond any existing heritage becomes a powerful drive in any market-driven economy (Bade and Parkin, 2012). Such development advantage is not only driven by technology, new concept design, or disruptive techniques enabling the system architecture itself, but also the design engineering methodology used to create it since development cost most likely is the most critical initial barrier. Hence, this relationship is critical, and becomes an objective for this thesis. It is a complex challenge with multiple contributors such as: technology knowledge, computation capability, workforce availability, schedule, funding, prototyping cost, implementation capability, operations scheme, and design principles, among many others.

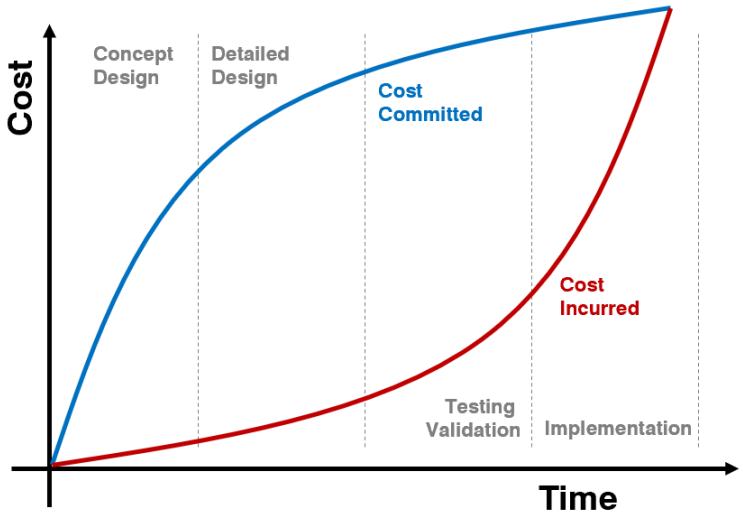



*Figure 10. Cost evolution during project phases. (Kihlander, 2009)*

### 2.1.4. Competitiveness

Therefore, if less resources could be used in the design, implementation, and operations of a system that performs at similar or better performance levels, this would mean a cost reduction, and a more competitive system architecture providing an advantage against competitors as well as older versions (improvement). Current market and industry trends invoke the principle of doing more with less, as economy and business frugal principles show (Radjou and Prabhu, 2014).

Competitiveness is in today's world more and more driven by enabling and rewarding flexible research and development, innovative solutions, affordable quantities, and customer involvement. Under that light, new design methods must provide ways to bring agility towards prototyping and development, as well as infusing data-driven steps into the design method so adaptability to new needs is easier (Daszko, 2018). Furthermore, methodology optimization should be systematically explored and integrated, as well as newer techniques that can increase speed and reduce cost. Thus, innovation is at the core of development processes enabling sustainability and competitiveness (Kuncoro and Suriani, 2018).

A design and systems engineering methodology within this new context should be about bringing faster, easier, and more efficiently better products to market, while reducing resource utilization and cost. This is indeed applicable to known economic conditions, but it becomes even more relevant under foreseeable and growing scarcity constraints.

### 2.1.5. Workforce and Capabilities

Another source of change and scarcity is the current situation of workforce (PWC, 2020). This is emphasized by new social and work dynamics established with the infusion of cultural changes and new techniques such as AI-driven and machine-driven methodologies (West, 2018). Disruption is indeed one key concept within this point. While tools and techniques at the disposal of organizations are starting to disrupt workflows (e.g., 3D printing and machine learning), they also mean that the way we work is changing. However, not only are tools developed during the "gig-economy" a source of change, but the planet itself is also in that mode. From climate change and social changes, to environmental situations like the pandemic in 2020, the future of work is becoming outdated and work dynamics are being challenged by new realities and possibilities (Vollini et al., 2020). Technology change drives today's economic growth and quality of life improvements, thus design, implementation, and operation methods need to address those social, generational, and professional changes and opportunities today.

The consequence of a high specialization within organizations is, among others, the lack of system-level thinking, that often leads to a growing need of generalist professionals across industries, markets, and businesses (Lurie et al., 2002). Those generalists are systems engineers and systems architects connecting disciplines and know-how to support products and services provided by an organization. In many ways they are the glue among teams, professionals, methods, and deliverables. Within an increasingly complex world specialists are both needed (Epstein, 2019) and often misused, however they are key for any healthy organization to go beyond their current state upon any heritage foundation.

Furthermore, the importance of heritage and the growing complexity of today's world also force us to find new and better methods to deliver products, services, and partnerships. Within that context, growing technology trends such as data-driven and AI-driven workflow enhancements are also a potential source of scarcity towards the use of more traditional methodologies. The integration of machines, machine-driven enhancement tools, and humans poses both huge opportunities and challenges such as: [1] the re-education of workforce (Rampersad, 2020) and AI training, [2] the creation of collaborative methodologies between machines and people (Daugherty and Wilson, 2018), and among others [3] a unique opportunity and need to reshape engineering practice. The evolutive design approach takes this as a foundational point, emphasizing tool-agnostic principles towards architecture system design methods applicable to these challenges.

### 2.1.6. Energy and $CO_2$

An everlasting constraint in human history has been the need for energy (Smil, 2018), in today's mid and long-term future where climate change, population growth, cost reduction, technological conditions, and market competitiveness among others need to be considered (Madureira, 2014) even more so. Today main energy sources of energy worldwide are basically fossil fuels (oil, coal, gas) and nuclear, with a growing participation of renewable energies (Letcher, 2008). However, as Figure 11 shows, those needs are going to grow significantly (EIA, 2019), and so their associated CO2 emissions (Marchal et al., 2012) especially with an end-use consumption shifting towards electricity across fields (EIA, 2019). However, with 10B

people by 2050, current levels of energy efficiency need to increase at least 65% of $ per W and per person (Cody, 2017) in order to maintain the same level of comfort we have today.

The implications of this shift in the energy paradigm affect global warming and pollution levels among many other environmental and health challenges. Thus, achieving better energy performances across complex system architectures (e.g., homes, cars, machines, etc.) using more optimized workflows is critical. This is very relevant because human trends show that we tend to constantly surpass any given energy production capability very quickly as the graphs shows.

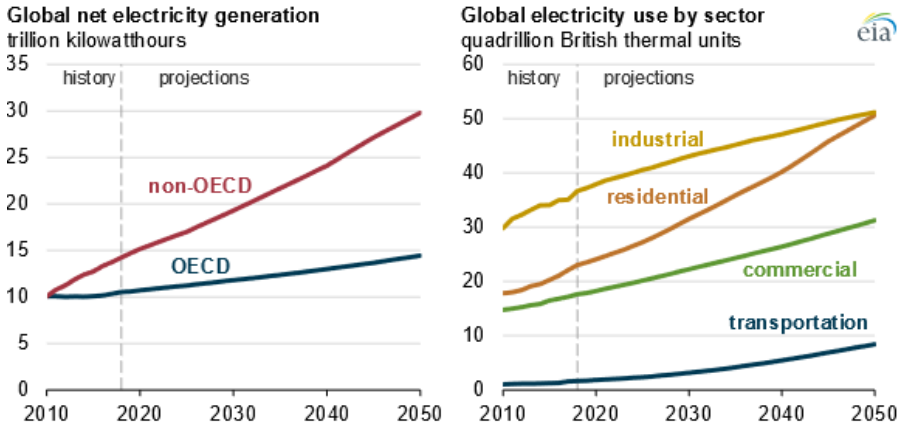



*Figure 11. Global electricity generation values and use by sector (EIA, 2019).*

Nevertheless, the energy balance of a hardware-based architecture must be measured from ideation to operations considering design efforts, workforce, computing power, materials, manufacturing, transport, repairs, etc. In other words, it should include the whole process of how we do things (McDonough and Braungart, 2010). Therefore, at a high level if a system architecture can accomplish better with less, its energy needs could directly be reduced. At the same time, the more adaptable such system is, the less energy is needed to rebuild, remake, and change components, leading to better direct and indirect energy efficiencies. Furthermore, energy scarcity is not only about the difficulty of its generation and availability, but other indirect consequences such as CO2 emission, processing of natural resources, cost of living, social development, etc. Perhaps dwelling activity (Gevorkian, 2009) exemplifies this associated scarcity (and indirect cost) better than any other field due to its market size worldwide. It is also the perfect example showcasing the needs for better design approaches, because of the inherent complexity of the systems in the field (Bauer et al., 2009).

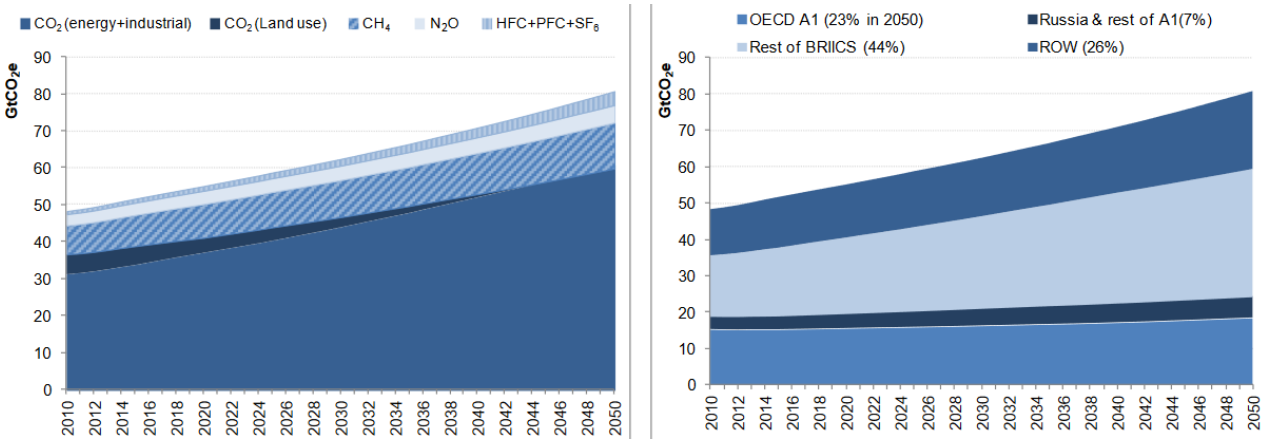



*Figure 12. GHG Emissions Baseline 2010-2050 by gasses and region. Source OECD (Marchal et al., 2012).*

## 2.2. Complexity

The nature of these challenges makes complexity both a stressor in the design process as well as in the context of technical practices. Complexity influences new needs for system design practices, as well as system architecture capabilities in the beginning of the 21st century (Alexiou et al., 2009). During the past decades, developments in electronics, computation, telecommunications, and manufacturing have made system complexity a characteristic of increasing importance. For instance, traditional land-line phone evolved into smartphones, simple combustion-engine automobiles are leading the way to autonomous cars, and VHS video clubs enabled online personalized streaming services. In summary our society demands complexity (Alexiou et al., 2009). In fact, the practice of systems engineering and lately also systems architecture (as a multidisciplinary approach toward complex systems design) is increasingly being dedicated to both managing and improving system complexity, as well as associated design processes (Frey et al., 2011).

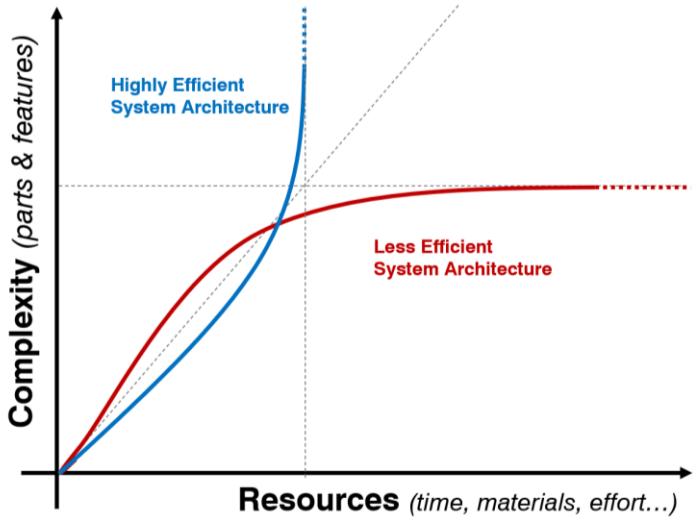



*Figure 13. System architecture efficiency based on complexity level versus resource utilization.*

Dealing with complexity could be quite costly when traditional engineering design methods are used, since they are algorithmic, deterministic, and point-design driven (Braha et al., 2006). However, designing complex engineered systems (CES) must deal with the self-organization and open nature of complexity. On the other hand, if we understand complexity as the characterization of a system with multiple interacting parts, then it is at the heart of the system efficiency and performance levels. For instance, if we compare a robotic limb with an organic leg, the latter has many more elements and interacting parts (e.g., cells), however its level of performance, dexterity, repairability, and implementation easiness is much higher. In essence, more complexity could bring more efficiency, thus an improved design methodology needs to be based on an evolutionary foundation and open relationships among its parts (Braha et al., 2006).

Therefore, the inherent complexity of a system is at the same time a way to improve resource utilization efficiency, as well as a source of scarcity if not managed properly. Regarding the first, natural and biological systems use much less resources, do not need manufacturing (they grow by themselves), and can adapt to a new environment (adaptation) over time (evolution) much better if they are compared to human (artificial) electromechanical systems.

On the other side, a lack of proper complexity management and adequate system design methodology for complex system architectures could be a huge source of scarcity. Complexity is driven by society, economy, nature, circumstances, requirements, and system capabilities. If not tackled properly, the alternative is a less optimum and often opposite path leading to loss, defeat, stagnation, and suffering. For instance, if dwelling systems needs for upcoming decades in terms of resource utilization, comfortability, and cost are not met, the result could affect generations to come affect generations to come impacting societal development at large. This is something that has happened before, for instance during the Middle Ages in Europe after the fall of the Roman empire (Benevolo, 1977) in terms of salubrity, economic stability, and dwelling

comforts. Design, engineering, and art complexity comes from many areas (Alexiou et al., 2009), for instance:

- *Multidisciplinary*: More and more disciplines are involved in the design process of a complex system (e.g., mechanics, electronics, thermal Eng., marketing, etc.), increasing the complexity levels of both system architecture and the related design process. The more software, hardware, and user concerns become part of a system, the more efficient solutions are needed as shown in Figure 13 (Frey et al., 2011). Furthermore the collaborative (Safavi, 2016) and concurrent nature (Salomone, 2019) of such process increase even more the importance of the design process itself.
- *Environmental*: Availability of resources in the natural environment often leads to higher levels of complexity as a strategy to cope with change (Norberg and Cumming, 2008). The interaction with the natural environment is about dealing with the unexpected, with changing parameters, and often with uncontrollable environments.
- *Resources (supply chain)*: Manufacturing and natural resources utilization have indeed an intrinsic complexity associated to them (Milner et al., 2013). Production constraints, material incompatibility, cost constraints, and more intricate geometries in product and system designs are among some of the current and growing challenges directly related to system complexity (Duehr et al., 2019).
- *Biological* systems are complex by nature and defined by self-organization principles. State-of-the-art techniques for software design (Hingston et al., 2008), and structural topology optimization (Rozvany and Lewinski, 2013), among many others are based on biological principles and subsystems interaction (Nomura and Asai, 2010).
- *Cultural*: Technology and engineering have increasingly become an intrinsic part of complex artistic manifestations such as architecture, theater, cinema, etc. (Casti and Karlqvist, 2003). However, at the same time many technology-driven sectors (e.g., IT, social media, web, etc.) keep integrating more often cultural aspects as part of their service and product development, for instance user experience (UX), human-computer interaction, and ubiquitous computing among many more (Ekman et al., 2015).
- *Architectural*: This is indeed the case when dealing with dwelling systems since the human factor is included, on top of all the previous points. Both at the largest scale of urbanism (Walloth et al., 2013) and the simplest level of individual homes (Venturi, 1990), architecture is often both based and driven by complexity.

## 2.3. Performance

The search for performance in terms of efficiency, speed, and capability for any system architecture is directly related to the process and tools used to implement such achievement. Thus, this process often depends on a specific culture, management style, and heritage associated to any given organization or institution (Kunda, 2009). Figure 14 shows how a specific architecture design (A) could incrementally improve its performance ($A_1$ to $A_n$) by serving more needs with the same amount of resources, but requires a change in both design and methodology to achieve a more efficient solution (B), so more requirements could be served with less resources.

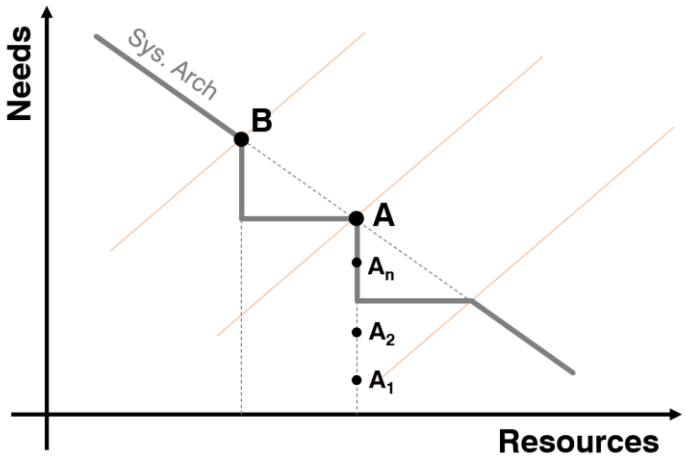



*Figure 14. Heritage and incremental improvements (A), versus performance leaps (B) enabled by design and methodology.*

For an accomplished organization, professional, or discipline practice used to a specific method or process, it is complicated to embrace change and follow new paths, especially when a specific way of doing has been prolonged over relative long periods of time with relative success (McCalman et al., 2015). However, the study of natural evolution (Herron and Freeman, 2013) shows us that adaptability through variation is the key towards dealing with change and entropy, which is indeed one of the most important physics laws in the universe (Charlesworth and Charlesworth, 2017) .However, inherent to that process is the fact that those changes are based on heritage, which means they are based on previous proven solutions and therefore have been validated at some point. In the everlasting balance game between resources and needs, the ubiquitous cultural heritage is often seen as barrier for innovators due to the inertia against change that it brings, and as a risk for more conservative managers. Hence, heritage solutions and methods could and should be turned into a positive and necessary component within an evolutive design process. This is especially relevant if current capabilities and implementation solutions of a system are considered as building blocks towards any new system architecture based upon them. They can provide both a feasibility assessment and a design foundation. This is what nature does, it builds upon validated solutions, to optimize more fitted performances which continuously adapt to the environment through genetic, temporary environmental, and genotype-by-environment variations (Borgnakke and Sonntag, 2013). The balance and integration of current and future methods is key for this research, as well as it is to any design and systems engineering methodology seeking better performance and faster methods (Braha et al., 2006). Furthermore, this approach also works when there is no heritage towards a new system architecture and a new approach needs to be implemented, but there is an abundance of feasible subsystem technologies. This is a key objective of this systems engineering and design process which is oriented towards those quantum evolutionary leaps (Figure 14) in the systems performance.

## 2.4. Multidisciplinarity

As previous points highlighted, complexity is intrinsic to human development, and stressors such as competitiveness, lack of resources, status, etc. push it towards improving the performance level of system architectures, as well as to create completely new approaches and methodologies that enable them.

However, such complexity is multidisciplinary in nature and becomes both a stressor and a constraint, towards products and processes. Multidisciplinarity or interdisciplinarity could be addressed from technical, social, and humanities standpoints (Finkenthal, 2008), and it has multiple fields of application as standpoints (Frodeman et al., 2017). Thus, the need to address a common challenge is both critical and often challenging when considering: [1] multiple design disciplines, [2] connections across parameters and geometries, and finally [3] workforce and organizational management aspects. Studying them in detail brings several conclusions as follows:

- *Design*. Independent of the field of application, a multidisciplinary approach is both an enabler and a stressor. While such perspective enables results that are not possible otherwise (e.g., adding energy studies to a mechanical design process), it also forces the process to address many more constraints.
- *Connections*. Those constraints also lead to the establishment of a process or a framework where they can be tackled. When design efforts include not only analytical parameters (e.g., systems engineering) but geometrical information as well (e.g., architectural design), complexity increases. This is especially critical towards feasible, reliable, competitive, and efficient systems, with a potential great influence in the culture of an organization.
- *Management*. Such culture involves managing different types of professionals from creators and generalists to highly specialized and technically driven teams. Nevertheless, this also means that resources management needs to be tackled at a different scale and from a different perspective. From computational power to schedules, requirements, and constraints they all become more complex from a multidisciplinary perspective that requires coordination.

These aspects are certainly interrelated, and therefore a key question within this area is how to find synergies across all of them. Because this stressor conditions design efforts, as well as cost, schedule, and workforce (knowledge) activities, among other managerial but necessary aspects, it is certainly at the core of any future complex system development.

Thus, making more with less, could be then identified as a clear objective towards the improvement of system performance and capability, as so it is towards its design methodology. Complexity brings the need of a multidisciplinarity approach, which needs to be managed to efficiently address resource utilization (scarcity) and time constraints.

## 2.5. Agility

Nowadays, time is becoming indeed more and more a design stressor. The pressure towards getting lower time-to-market solutions, growing user expectations tend to accelerate the need for faster processes especially from a marketing standpoint (Singhania et al., 2019). Social media and other new trends in the era of information technology also contribute to create and enhance that sense of immediacy (Petro, 2017).

Agility however is not only about time, since it has lot to do as well with the leanness in resource utilization of any design and development process. The less time and resources are needed, the faster, easier, and therefore more agile any methodology becomes.

Regarding temporal constraints, the notion of agility involves not only the speed of the design process, but also the capability of such process to quickly change due to variations in requirements, constraints, upgrades, etc. This is critical in any present and future system design engineering effort in highly competitive environments.

From a leanness standpoint, how those resources are used is the essence. In the development of a complex system, not only the number of resources is relevant but also how efficient that use is, since both enable flexibility and agility. These resources include workforce, people, power, computing power, hardware, time, schedule compatibility, among others.

## 2.6. Interconnection and Networks

Within the context of complex system architectures, another key stressor and enabler quite characteristic of current times, is the interconnection among systems and the concept of network. These have multiple sides and represent a valuable and new contemporary feature, that can also be seen in the natural world.

The development of computer science during the last century and the increasing use of data across products and services is becoming today a new standard changing both designs and methodologies. This is something upcoming sections will tackle from technique and modeling standpoints. However, it is already a trend changing forever how things are done. The integration of data-driven techniques not only make systems smarter (e.g., smartphones or autonomous cars versus traditional versions), but they also foster the connection between the system and its environment (e.g., car + GPS).

A perspective based on data brings a standpoint of interconnection between all subsystems integrating a system of systems (SoS), as well as in-between the system and its functional or environmental framework. In essence, data measurement means comparison, and comparison means interconnections and network thinking (Mitchell, 2006).

Consequently, modern complex systems more and more rely on data to improve their operations, optimize their performance, and even to condition new generations of their designs. Such complexity brings the concept of interconnected network (Duato et al., 2003) its components and subsystems, as well as to their environments and frameworks of operations. Therefore, designing complex architecture systems is becoming more and more a networked process that needs to consider its present and future environmental context. This is a both new enabler and stressor of current times, with critical consequences across products, services, platforms, and processes (Parker et al., 2016).

## 2.7. Design Heritage

Heritage solutions that are validated and proven approaches towards specific design challenges are indeed a powerful stressor in terms of systems design. Heritage brings pressure, influence, and opportunity towards risk assessment, decision making, and design principles for any complex system design and implementation. This applies to any organization, field, or even professional practice. In high technology fields such space exploration, the notion of heritage, and thus the readiness and feasibility of a future technology or system has been standardized, measured, and ruled. Technology readiness levels (TRLs, Mankins, 2009) present a strategy across fields to assess risk, planned technology development, and most importantly assess system feasibility and project management (Blokdyk, 2019). Thus, heritage becomes a highly relevant stressor, as well as an enabler towards new system developments since it conditions system assessment, development, and planning especially under a context of increasing scarcity. Heritage provides a strong foundation, which does not have to be considered as a constraint towards any development.

While the inclusion of heritage in a design process could be perceived as a hindrance towards the infusion or even

the disruption of new solutions (Henchoz and Mirande, 2014), it could also be a reinforcement that opens doors towards truly quantum-leap solution across technical fields. The next section will introduce the notion that technical heritage, natural selection, and evolution processes are connected at many levels.

## 2.8. Innovation

Changes in society, the environment, and the operational context of a system often drive a constant need for new solutions, improvements, and upgrades that can ripple throughout design and development methods in any industry. Such need usually is managed and conditioned by resource availability, technical constraints, market needs, behaviors, and cultural factors (An and Rau, 2019), among many others defining a posture that could be conservative, incremental, or radical. In other words, innovation is indeed an interrelated network of inventions, innovations, and needs (Frodeman et al., 2017) that could be understood through both static and dynamic models (Narayanan and O'Connor, 2010). Disruption is the more radical approach towards innovation, and it has deeper consequences in the context of system design (Williams, 2010) and across the full system lifecycle (Paetz, 2014) while conditioning market infusion (Figure 15).

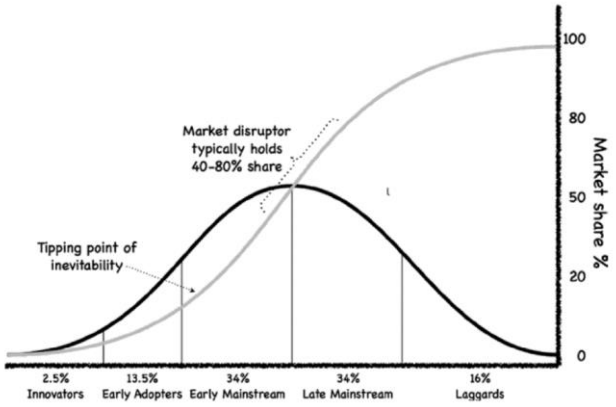



*Figure 15. Everest Rogers' diffusion of innovation per Paetz, 2014.*

In our current and globalized world, innovation as a construct is starting to be understood as something on the edge of a greater change or wave (de la Tour et al., 2020). Traditionally very disruptive technologies capable of creating full technical ecosystems such as the invention of internet, the distributed electric power, the radio, biotech, or blockchain tend to require large amounts of investment and time, before the technology is ready. However, nowadays this is changing and new trends like deep-tech (European Union, 2019) see a combination of biology, computer technology, and new manufacturing and energy technologies, among others as a way to speed this dramatically (De la Tour et al., 2017).

## 2.9. Cultural Disruption: Methods and Products

In summary today's complex architecture systems across technical and creative fields, require a multidisciplinary standpoint for their development. From single components (e.g., sunglasses) to large complex systems (e.g., modern building), they all need a multidisciplinary approach towards design, optimization, and validation. These standpoints not only include quantifiable disciplines such as mechanical engineering, thermal design, structural analysis, data management, etc. (Pahl et al., 2007), but also qualifiable ones such as aesthetics or user experience (UX). New methods, such as evolutionary techniques have been increasingly embraced from digital (Bonanomi, 2019), software, and computer science perspectives. However, hardware-based processes still rely heavily on linear and non-evolutionary workflows (Braha et al., 2006) rather than more disruptive approaches capable of dealing with more uncertainty and complexity.

However, these iterative design processes often use a serial approach, so the result of a disciplinary step becomes the input for the next one, and so on. For instance, an artist conceptualizes a design that the engineering team will have to make implementable considering architectural and mechanical aspects, and then structural, thermal, material assessment, etc. Finally, a manufacturing team will make it cost effective so it can be produced (one-off, series, or mass produced) while

marketing and finance teams flush out all other subsequent details (Kamrani et al., 2016). This process can be tedious, costly, and very complex, regardless the sector. Mass production increases complexity due to large production volume making the most of automated processes. On the other end, short-series products endure the burden of ad-hoc methods and higher cost. From both perspectives, the increasing complexity driven by new design requirements and production stressors tend to challenge the method itself, and often requires exploring, testing, infusing, and maturing improvements.

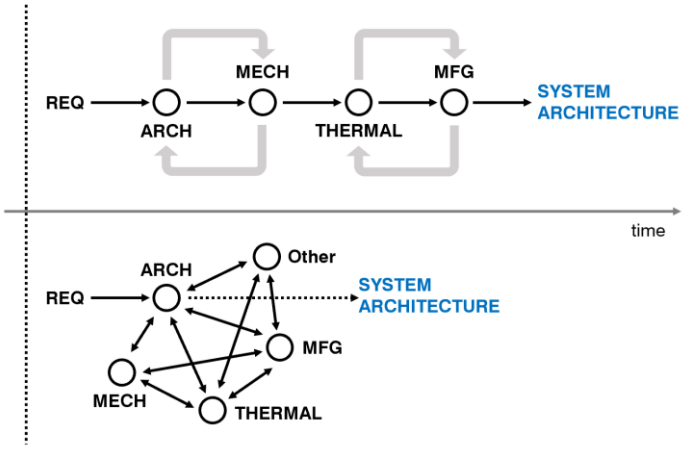



*Figure 16. Serial versus network design methodologies with time*

The last decades have seen a growing increase in the rapid prototyping approach (Liou, 2007), partially due to improvements in both digital modeling and 3D printing for instance. The resulting workflow is well stablished today, going from fast-handmade architecture or product designs in workshops and studios worldwide, to elaborated 3D printed prototypes in engineering (Cooper, 2001), product design, archeology, or medical (Bártolo et al., 2012) fields. The design approach is completed with fully functional prototyped robotic and electromechanical systems (Macdonald et al., 2014, Jones and Flynn, 1993). This has also been enhanced recently with machine-driven methodologies such as generative design workflows (Wujec, 2017). These advancements are becoming more affordable and available in the areas of design (e.g., BIM), analysis (e.g., Multiphysics FEA), and rapid prototyping tools (Killi, 2017), as well as digital and rapid manufacturing (Hopkinson et al., 2006). These new tools allow addressing multidimensional problems, with both synergetic and hands-on experience so architects, designers, and engineers can make decisions faster and disregard unfeasible paths. Furthermore, not only tools and methods have evolved, but the management of design competencies and organizational schemes (Bonjour and Micaëlli, 2010) has also been evolving and is considered as a key aspect in the culture of a company.

The combination of these methods allows us to approach the challenge of complex system architecture development from a full cycle perspective, so design, manufacturing, and operations are starting to be become more closely interconnected and potentially optimized simultaneously. Computational trends based on data-driven and machine learning methodologies (e.g., artificial intelligence or AI) such as generative design (Gross et al., 2018), also present a feasible framework to tackle all aspects of the creative and implementation business. Figure 17 shows a concept research project developed between Autodesk and JPL that used AI-driven techniques to optimize structures and reduce mass while considering multiple manufacturing methods (traditional and otherwise). With or without artificial intelligence, if data-driven computational methods are combined with a complete full-cycle design flow, multiple discipline standpoints can be connected and balanced towards a faster and more adaptable digital manufacturing approach (Wang and Nee, 2009).

The need to consider new methods as well as the new capabilities that are already available, defines the last stressor considered within this research, cultural disruption. More and more, these new ways are a must because: [1] new design requirements are not feasible without them, [2] competitiveness forces the reduction in cost while increasing capability and agility, [3] overall resource scarcity forces new trends. While disruptive innovation becomes the modern currency within new companies (e.g., start-ups), in older and bigger organizations (Fried and Hansson, 2010) the question is about how to fully embrace such new processes in harmony with an already established (and perhaps rigid) cultural heritage. New methodologies (Figure 16), workflows, and solutions do stress established methods in any given organization. They affect

risk perception and influence any outcome based on environmental and cultural changes. Nevertheless, traditional, and culturally accepted approaches could be combined with more disruptive methods. Most likely they must do so since heritage methodologies often not only present a solid foundation but also a broadly distributed platform for news ways to be implemented.

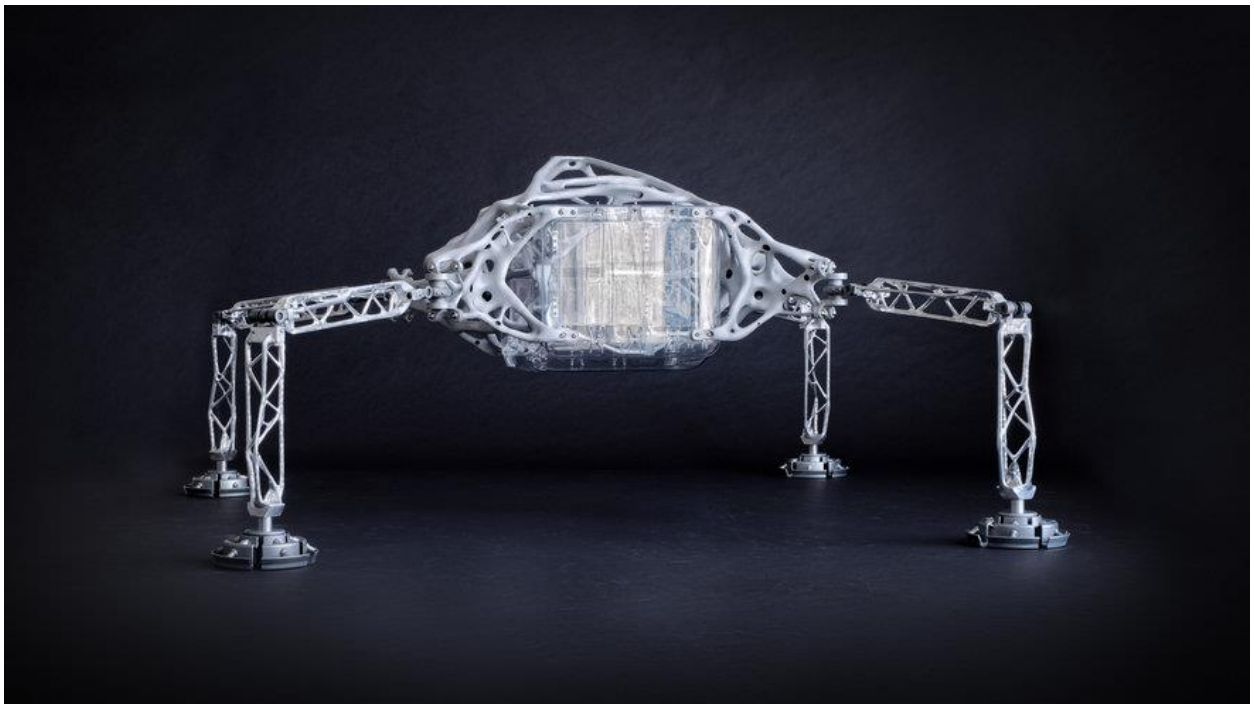


*Figure 17. Concept by Autodesk – JPL using AI-driven generative design tools to optimize structures. (Autodesk, Core77 et al., 2019)*

## 2.10. Conclusion

Designing complex hardware-based systems today must address a series of stressors due to increasing changes in the context of many technical practices. These stressors, which previous sections summarized, not only influence any system design practice currently, but they are also a growing trend. The lack of actual heritage for a new system, the increase in resource scarcity, and the cultural influence of an established way of doing business raises a question that affect both the product result as well as the design process behind it: How do we make a multidisciplinary design process efficient and disruptive enough (Rowan, 2019) when a challenge is being addressed for the first time while potential outcomes require a radical new approach and new methodologies are most likely needed?

Considering the inherent complexity behind such system architectures across fields, the goal in answering that question could be more about creating a solid, universal, and adaptable foundation that enables such new design, rather than a static approach which could easily become too tailored to a specific field or given deterministic context. Within the nature of these stressors lies the need for adaptability in any approach tackling them in a robust way. A method to find faster and better mature system architectures offers a powerful platform to reduce the use of resources (e.g., workforce, computation, etc.) across the system lifecycle, from exploration and ideation to implementation and operations.

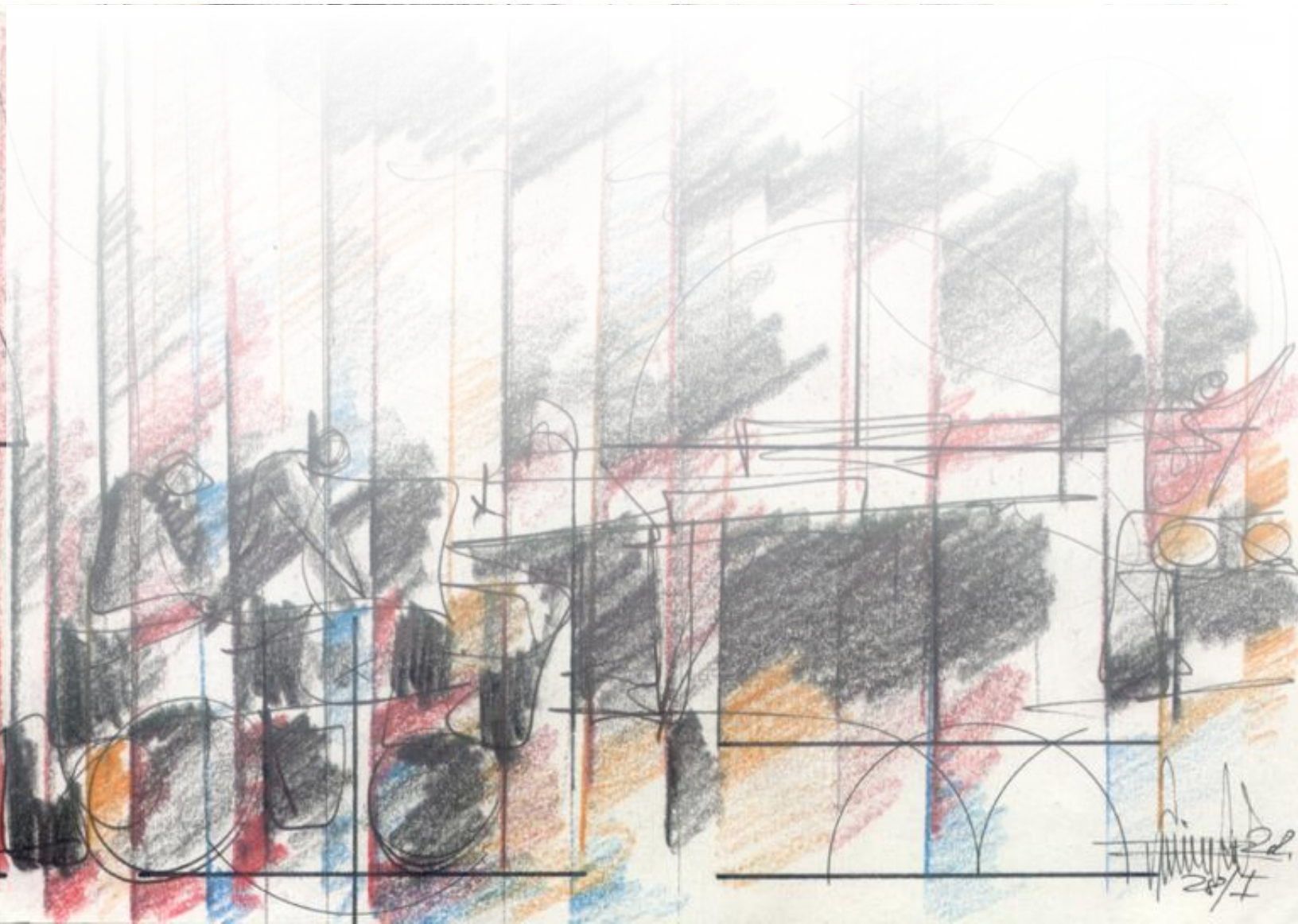
Today's world is dramatically changing from multiple perspectives at once. Any process aiming to design, develop, and implement present and future systems needs to address and embrace those changing conditions. Furthermore, those systems themselves should be able to address rapid changing conditions, reduce resource utilization, and embrace all disruptive capabilities that new design and implementation methodologies can enable.

# DESIGN, SYSTEMS, AND EVOLUTION

## Literature Review

## CHAPTER 3

*"He who makes a question becomes a fool for five minutes.*
*But he who does not asks a question remains a fool forever."*
***Chinese Proverb***



# 3. Design, Systems, and Evolution: Literature Review

This chapter addresses heritage, state-of-the-art, and related gaps within theories, tools, and applied methodologies involved in the design, description, development, and study of complex system architectures. These approaches and practices come from engineering, biology, and computer science realms and they represent the foundation toward the development of a novel and complementary evolutive system design approach.

Thus, this chapter is structured in four parts. The first three sections address methodologies within these areas:

- **Design engineering** (3.1). This section reviews design methodologies and theories across human history from a multidisciplinary standpoint towards system architectures and complex systems.
- **Systems engineering** (3.2), including practice, techniques, and approaches used and validated across industries.
- **Evolutionary theories and design** (3.3). This section includes principles, methods, and applications across domains and disciplines. This section is quite foundational since it tackles both an overview of natural evolution principles, as well as their application to current evolutionary computational and other engineering techniques.

These reviews of the state-of-the art methods are always done under the perspective of hardware-based system architectures. The overarching objective it to address design methodologies considering constraints such as complexity, heritage, scarcity, and agility, among others. While each section presents conclusions and gaps under the light of this research, the last section 3.4 introduces an overall conclusion as a keystone for this thesis.

Multiple reasons are behind this literature review across domains. They are summarized in the following points:

- **Designing for complexity**, or in other words, tackling the process of designing and managing the design process of a complex system architecture has been an evolving practice since the beginnings of civilization. In recent decades, the notion of design has not only been applied to hardware but also to software and other services. It is key then to understand and identify key gaps across the full spectrum of such activity while considering: [1] time, [2] life cycle, [3] field, [4] software and hardware capability, [5] efficiency, and [6] speed, among many more. Within the domain of hardware-based systems, this point tackles the geometrical definition and management of parts, components, assemblies, and other technical visualizations.
- The other side of this process is to manage **non-geometrical aspects of a complex system architecture**, including documentation, development, definition, optimization bases, etc. This is the domain of systems engineering (SE) and hence this literature reviews both theory and practice trends within this area. Nevertheless, within the vast field of SE this research especially addresses methodologies that are oriented towards a more efficient way to tackle large and complex systems, independently of their software or hardware nature.
- During the last decades especially, complexity and efficiency have often been tackled across technical fields with **nature-inspired methodologies**. Biology in general, and natural selection in particular have indeed become two critical areas in such approach. For instance, evolutionary computational techniques such as genetic algorithms in the 90s spun off a new approach towards both programming and systems optimization. Hence, it is critical to review all available literature towards: [1] core natural principles used by such techniques, [2] practical applications from systems, computation, software, and hardware design standpoints.

While these areas and domains might seem unconnected, the reality is that they are tightly involved in the design and design optimization of any complex system. On the other hand, such broad perspective could also present multiple potential gaps across these fields, and most importantly across key connections and synergies between them. A method to tackle a multidisciplinary problem requires a multidisciplinary foundation. Following sections present findings and reviews of such state-of-the-art techniques as well as those critical design gaps among them.

Section 1.8 already presented multiple definitions used within this research. Nevertheless, the fields of engineering design and systems engineering could be interwoven across some of these methodologies and techniques. For instance, concepts within the fields of design thinking and engineering systems thinking are overlapped (Greene et al., 2017). Hence, in these cases such links should be noted, and their approach will be studied only in one of the sections.

# 3.1. Design Engineering Paradigms

## 3.1.1. Approach and Categories

Within the research area of design engineering there are two interconnected main subjects addressed by this literature review. First, are **design theories and models** (Chakrabarti and Blessing, 2014) developed over the last decades, which address different approaches toward design processes themselves, as well as targeted outcomes. From methods to facilitate the discovery of innovative solutions, to approaches developed towards their implementation through machine learning and AI, this research considers both sides. Furthermore, the spectrum from modern design theory protocols (Gero, 2011) to design approach techniques such as TRIZ (Fiorineschi et al., 2015) is also fully covered. Due to the broad extension of the human design activity, these design engineering theories are organized in categories as follows:

| ID Code | Category | Time Period | *Field* | *Driven by* | *Tools* (See Table 11) |
|---|---|---|---|---|---|
| DE1 | **Classical** | 300 BC to 14th C. | Architecture | Artist perspective | To1 |
| DE2 | **Renaissance** | 14th C. to 17th C. | Architecture, Art | Artist, Method | To1 |
| DE3 | **Enlightenment** | 17th C. to 18th C. | Architecture, Art, Objects | Artist, Knowledge | To1 |
| DE4 | **Modern** | 19th C. to 20th C. | Arch., Product, Process | Objective, Method | To1 |
| DE5 | **Descriptive** | 50s to 10s | Product, Process | Concepts | To1, To2 |
| DE6 | **Prescriptive** | 40s to 10s | Product, Process | Analysis | To1, To2, To3 |
| DE7 | **Design thinking** | 50s to 10s | Product, Process, Service | Framework | To1, To2 |
| DE8 | **Innovative** | 50s to Today | Product, Process, Service | New systems | To1, To2 |
| DE9 | **Method-driven** | 90s to Today | Product, Process, Opt. | Statistics | To1, To2, To3 |
| DE10 | **Process-driven** | 10s to Today | Product, Service, Opt. | Ontology | To2, To3, To3 |
| DE11 | **Integrative** | 10s to Today | Product, Optimization | Algorithms | To1, To2, To3, To4 |
| DE12 | **Evolutionary** | 90s to Today | Product, Optimization | Evolution principles | To2, To3, T4 |

*Table 9. Design engineering theories organized in categories by key characteristics and historical period.*

The second review area represents applied **design methods and tools** enabling specific uses and subsequent design workflows. For instance, these include fast concept hand-drawing techniques and integrated BIM methods (Deutsch, 2011). These methodologies enable different types of results at different design phases. This research is agnostic in terms of tools, but these families of solutions present a specific way to tackle design challenges, which can condition its practice.

## 3.1.2. Design Phases for Products and Processes

Regardless the field of application, going from one idea to the actual fully-functional physical system is usually a long process that requires multiple iterative steps (Dieter and Schmidt, 2012). Any design-to-implementation process aspiring to produce a hardware-based system at the end most likely will involve multiple looped steps and workflows to design, validate, and implement an idea. Moreover, this most likely considers both software and hardware. Across theories, models, and authors multiple and different steps or phases are considered from both practice and theory standpoints.

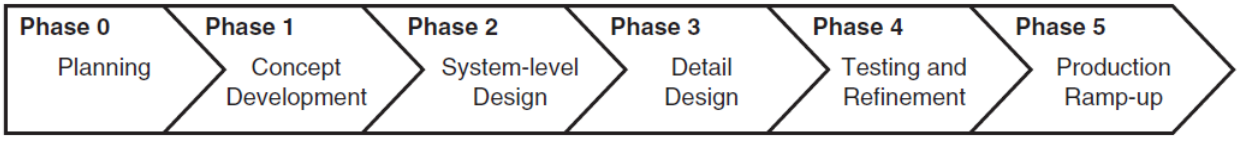



*Figure 18. Product development phases after Dieter and Schmidt, 2012.*

In general terms, multiple phases of product design could be organized in five main groups (Buede, 2009; Cross, 2008; Dieter and Schmidt, 2012; Dym, 2013): [1] problem exploration, definition and planning, [2] concept / preliminary design, [3] embodiment design, [4] detailed design, [5] evaluation. While descriptive methods start with a problem exploration phase followed by concept design activities, prescriptive methods begin the process with the analytical study of the problem, followed by analysis and synthesis of the preliminary concepts. Understanding and identifying the overall design phases, as well as main barriers and connections among them is key to better understand the broad spectrum of design theories. Furthermore, synergies, overlaps, and connections among design phases, also help improving the quality and reliability of the process as well as the system architecture itself. Table 10 shows multiple considerations regarding different design phases across techniques as well as their main barriers and key connections. The next section 3.1.3. studies in detail those groups of design engineering theories and summarizes their most important characteristics.

| | Phase / Theory | A. Classic | E. Modern | F. Descriptive | G. Prescriptive | H. D. Thinking | I. Innovative | J. Method-driven | K. Process-driven | L. Integrative | M. Evolutionary | Details | Main barrier | Key links |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Planning** | x | x | x | x | x | x | x | x | x | | *Resources, expectations, cost, scheduling, workforce, team, etc.* | *Client* | *All* |
| 2 | **Problem** | | x | x | x | x | x | x | x | x | x | *Customer needs, requirements, constraints, problem definition, information gathering, feasibility, heritage, reverse engineering, etc.* | *Client, Heritage* | *3,6,7,13* |
| 3 | **Concept D.** | x | x | x | x | x | x | x | x | x | x | *Generation, evaluation (e.g., decision matrix and Pugh matrix), morphological analysis, synectics, brainstorming, etc.* | *Designer* | *4,9,12* |
| 4 | **Embodiment D.** | x | x | x | x | x | x | x | x | x | x | *Preliminary design, architecture, materials, manufacturing, configuration, parametric, tolerances, diagrams, layouts, etc.* | *Culture Resources* | *3,9,10,12* |
| 5 | **Detailed D.** | x | x | x | x | x | x | | x | x | x | CAD, drawing, specifications, etc. | *Culture* | *4,6,7,9,10* |
| 6 | **Analysis** | | x | x | x | x | | | x | x | x | FEA, technical analysis, etc. | *Tools, Knowledge* | *5,7* |
| 7 | **Optimization** | | | | x | | | | x | x | x | Optimization techniques | *Tools* | *6,5,3* |
| 8 | **Testing** | | x | | x | | | | x | x | | Prototyping, testing methods, etc. | *Culture* | *7,5* |
| 9 | **Document.** | x | x | x | x | | | | x | | | Knowledge, visualization, eval. | *Tools* | *5,11,12* |
| 10 | **Implementation** | | x | | x | | | | x | | | Qualification, manufacturing, post-processing | *Technology* | *5,9* |
| 11 | **Delivery** | | x | | x | | | | x | | | Production planning | *Client* | *5,10* |
| 12 | **Marketing / Com.** | | x | | x | | | | x | | | Visualization | *Society* | *All* |
| 13 | **Operations** | | | | | | | | x | | x | | *Client* | *All* |
| 14 | **Decommission** | | | | | | | | x | | | | *Culture* | *1,3,* |
| 15 | **Recycling** | | | | | | | | x | | | | *Culture* | *All* |

*Table 10. Engineering design phases across multiple design theories.*

## 3.1.3. Literature Review

The goal of this section is not to present a complete list of techniques from a chronological or development standpoint, but rather a comprehensive reference of the most relevant theories and methods to this date affecting both [1] how a hardware-based complex system can be described and designed, and [2] all design thinking methodologies associated to its development. Within the vast exploration of design methods over the years (Wynn and Clarkson, 2018), this review as the previous section 1.7 about domains explained, merges both architecture and engineering practices across history.

### *3.1.3.1. Traditional Design Theories*

Perhaps one of the first attempts towards the study of what it takes to create **architecture**, and therefore its design methodology could be found during the Roman Empire. The ten volumes of *De architectura*, written by Roman architect, author, and civil engineer Marcus Vitruvius Pollio (80-15 BC) address the three key principles of building design such as *firmitas* (strength), *utilitas* (utility), and *venustas* (beauty) (Pollio, 2018). These principles became later a foundational part of many modern design theories, and they established the first design principle behind not only building developments, but also complex machines at the time such weapons, dewatering machines, and military devices, among others. His approach not only considered the design of the object itself, but also the role of the designer and its context addressing weather, location, logistics, etc.

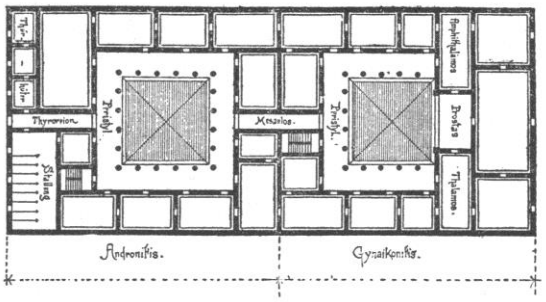



*Figure 19. Ancient Greek house drawing by Vitruvius*

During the **Renaissance**, Leon Battista Alberti contributed to the beginnings of a systematic design theory by addressing critical theoretical principles in the design of buildings (Lewis, 2020). The number of components, the control of their outline, and their position became the guidelines of his design methodology as explained in his ten books *De re aedificatoria* (On the Art of Building). The development of complex systems such as a buildings, during this time was based on numerical relationships ruling the work of architects and artists such as Leonardo Da Vinci (Zöllner et al., 2003), Brunelleschi, and Bramante (Roth, 1994). Thus, the object of the design process still goes through a heuristic process of conceptualization, but there is already an initial analysis of the problem and a synthesis of the solution. Quite often, a trial and error approach validates the final implementation, and influences the theorical principles afterwards.

During the age of **Enlightenment**, a renewed sense of rationality in the design practice was developed, bringing the notion of unreality as a way for design to explore possibilities that cannot be implemented in real life. The works of architects such as Ledoux or Boullee are paradigms of this approach, opening the space to a rational exploration of impossibilities.


*Figure 20.The Modulor. Le Corbusier 1943.*

The industrial revolution started to merge again engineering and classic architecture, leading to new range of materials, uses, and technologies. From the works of Violet-le-Duc and the Arts and Crafts movement in the late XIX century to the *Art Nouveau*, the practice of design not only affects the product itself (e.g., building, furniture, decoration objects, etc.), but also encourages reflection about the process itself (Benevolo, 1977). This served as a foundation of many developments occurring in the practice of **engineering and architecture design** during the Modern era in the beginning of the XIX century. Among many contributors of this time to the development, theory, and practice of complex systems from both architecture and product design there are two very relevant. First, the Bauhaus school founded by Walter Gropius in 1919, which made an emphasis not only on the practicality and innovation of the product, but also the constraints of cost and production. Secondly, the figure of Le Corbusier is also very critical among all architects, designers, and engineers of the time. He introduced key systematic principles regarding both the design construct and the design process itself, while enabling an optimization of such designs through combinations of key design features such as

reinforced concrete columns, standardized stairs, and open floors (Benton and Cohen, 2019). In his book *Five Points of a New Architecture,* he presents a systematic approach to the practice of architecture. His work also highlighted the importance of initial analysis in the synthesis of architecture solutions (Jencks, 2000), also affecting product design after World War II.

**Descriptive** design methods have been used since the beginning of time. However, it is in the contemporary world and since the beginning of the XX century when the theories about the design practice and design engineering methods really advance. Such design methods were later described by people like French in 1985 (Cross, 2008) in which they start with the exploration of the problem, then with the generation of a concept that is evaluated later, and finally with its communication. This approach could be fast, but it is limited in the use of optimization and parametric tools to assess other solutions. Modern design approaches divide this complex problem into a hierarchical construct, using abstraction, modularity, and problem-thinking from both top-down and bottom-up approaches, as well as the linearly and iterative waterfall approaches (Shukla and Krishnan, 2016). A simplified, yet very powerful approach to the design process was given by Morris Asimow based on a series multiple design sub-processes (Asimov, 1976). These sub-processes are based on information gathering (general and specific), design operations, and the evaluation of the outcome. This simple approach is repeated iteratively to explore alternative solutions, develop mathematical models, define subsystems, and address its implementation (Dieter and Schmidt, 2012).

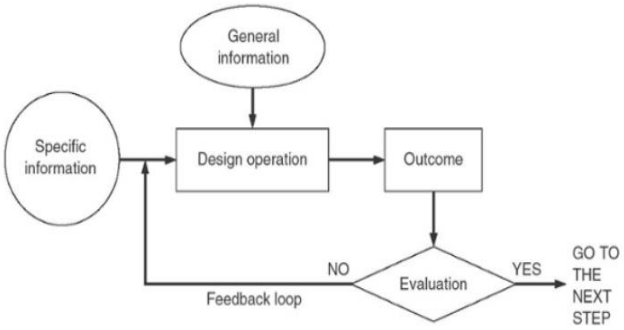



*Figure 21. Asimow design process (Dieter and Schmidt, 2012).*

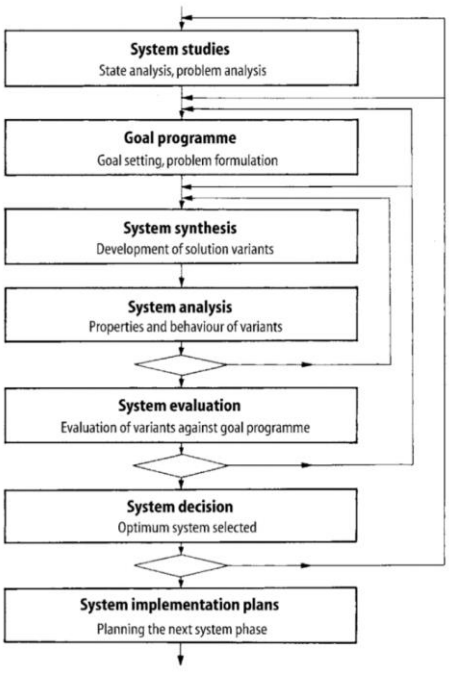



*Figure 22. Systematic System Design (Pahl and Beith 2007).*

The modern systematic design approach has developed since the late sixties by authors such as Pahl, Beitz, March, and organizations such as the *Verein Deutscher Ingenieure* (VDI). They presented a **prescriptive** methodology (Cross, 2008), based on system interrelationship. Energy, material, and signals are the key drivers at the base of functional, working, constructional, and system relationships that organize the designer approach toward the challenge ahead. Under this light, "designing is the optimization of given objectives, against conflicting constraints" (Pahl et al. 2007). Thus, both product and process design are organized around basic phases of analysis, synthesis, and development. This specific design method influences the full design process including: problem and task definition, information gathering, concept generation, evaluation, embodiment design and detail design (Dieter and Schmidt, 2012). While there are iterative cycles in between such phases, that enable among other things, the optimization of the final solution, this approach could become rigid and often leads to strict procedures within the culture of an organization, due to lead-times and risk posture. This methodology is widely distributed, and often aims towards developing a point design or a defined family of solutions. It is more prone towards quantitative requirements rather than qualitative ones, and it discretizes all disciplines involved.

Within the systematic view, theorists such as March proposed a system based on a production-deduction-induction scheme. The analysis of requirements produces presuppositions that the designer could use to foresee and analyze the performance of a design, as well as to make changes accordingly (Cross, 2008). This is an interesting approach since while being systematic in nature, it addresses the design approach from an intuitive and psychological standpoint. Nevertheless, the traditional systematic design approach towards analysis, synthesis, and evaluation, could also be altered

creating a bridge between descriptive and prescriptive methods when it comes to the design process. As such, the design process could start from a synthesis approach leading to analysis and evaluation (Seider et al., 2016), as chemistry engineering exemplifies. Under this prescriptive perspective the design space was defined by Ullman, Wood and Craig in the 90s (Ullman et al., 1990) and Dym reapplied it in this century (Dym, 2013) as Figure 23 shows:

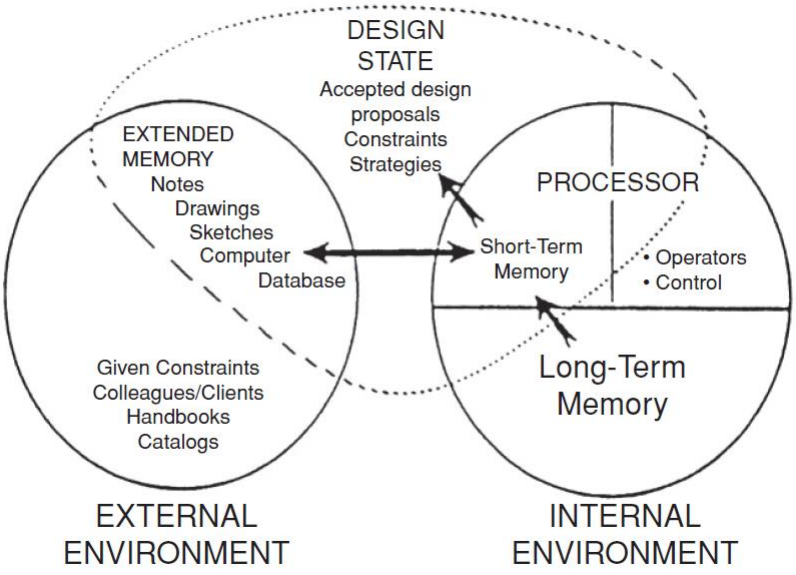



*Figure 23. Design environment (Ullman et al., 1990)*

A constant and universal rule in this evolution of design engineering is that the more complex problems are, the better methods designers need (Jones, 1992). Many of these design processes are conceptually organized as linear or in a waterfall scheme with multiple iterative phases. However, also in the 50s, Evans describe the methodology to design complex ships within a spiral (Figure 24) approach (Evans, 1959), addressing the complexity in between those phases.

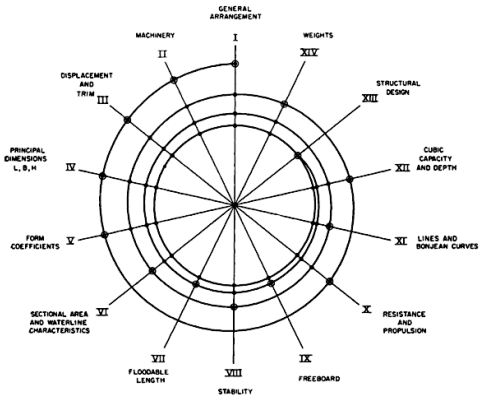



*Figure 24. Basic design approach. (Evans, 1959)*

With origins in the 50s and authors such as John E. Arnold, **design thinking** theories addressed a series of cognitive processes in order to develop new and innovative concepts across industrial, social, information technology, software, educational, and service areas, among others (Curedale, 2013).

Design thinking presents three main areas, as defined by Plattner et al., across all tools and techniques within this approach (Plattner et al., 2010): [1] exploring problem space, [2] exploring solutions space, and [3] aligning both iteratively. While the applications are plenty for this approach, from a hardware-based standpoint the prototyping phase is critical (Greene et al., 2017). This is clearly enhanced by the rise of new rapid manufacturing techniques (Hopkinson et al., 2006) such as 3D printing, as well as the infusion of smart devices and mechatronics in our daily lives. This involves a design process tackling: [1] empathy with the problem (understanding and observation), [2] synthesis, [3] ideation, [4] prototyping, and [5] testing under an iterative approach among these steps. These are also related to the creative process phases as described by: [a] insight (problem formulation, [b] preparation (conscious solution attempt), [c] incubation (no conscious effort), [d] illumination (emergence of ideas), and [e] verification (conscious development) (Lawson, 2014). Within the design thinking perspective, problem and solution evolve together (Dorst and Cross, 2001) enabling further innovation. Such approach presents flexibility, but at the same time lacks

the structure that systematic approaches present. Agility and analysis are not necessarily part of this approach which is more concentrated on initial phases of design engineering.

This creative platform led to other approximations based on the involvement of the human perspective in problem-solving techniques such as human-centered design (HCD), with the objective of addressing user needs, improving user experience, reducing stress, improving competitive advantage, and enhancing sustainability (Rosenbrock, 1989).

Similarly, User-centered design (UCD) develops a framework that considers goals, users, environment, tasks, and product workflows (Norman and Draper, 2018). The objective of such approach is to simplify and clarify the process, while

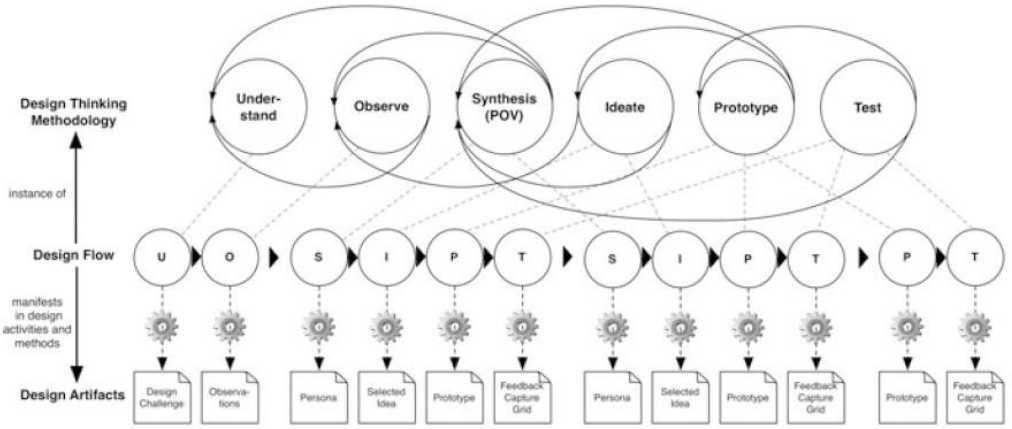



*Figure 25. Design thinking methodology (Plattner and Meinel, 2009)*

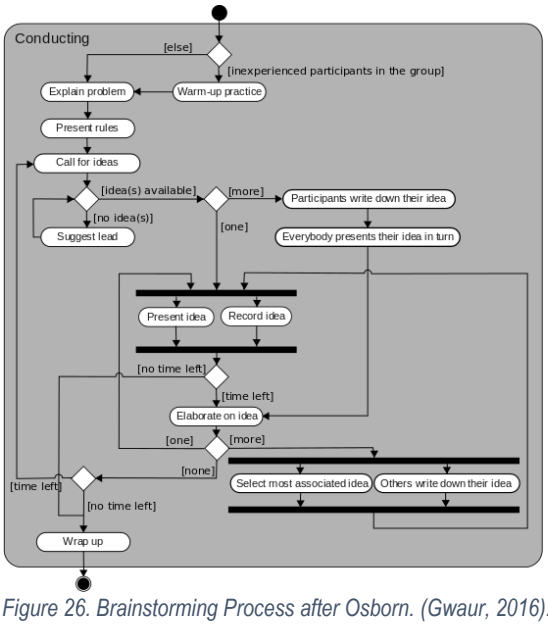



*Figure 26. Brainstorming Process after Osborn. (Gwaur, 2016).*

making the most of system restrictions and constraints. For this purpose, the system considers the person, the scenario, and the use case. Similarly related techniques enhance the design methodology with an emphatic use of tests (Rubin et al., 2008) and questionnaires (Vredenburg et al., 2002) as key tools.

Within the space of design theories and methodologies there is another group of focused techniques developed towards the pursuit of more innovative designs. Among of them is the widely distributed technique of brainstorming, which was created in the 60s as a way to explore the trade space of ideas (Osborn, 1993). This is a facilitated and very effective activity that gathers all ideas (including wild ones) from participants allowing people to build new ideas upon someone else's. Then these ideas are combined synergistically to allow a collective development of new and unforeseen options. See Figure 26 (Gwaur, 2016) to see the initial process within this approach.

Along these lines, Synectics is also a problem solving methodology for groups developed by George Prince and Gordon Williams (Gordon, 1961; Prince, 2012). The process uses the metaphor as an idea development technique to make interesting ideas feasible by identifying new paths of action.

Related to these approaches, TRIZ theory was developed by Altshuller in the 80s. It is based on 40 systematic principles (Altshuller, 1984) that define in principle any complex system. He obtained these principles after studying thousands of inventions. These principles when applied to a new system often contradict themselves. By managing such contradiction, he developed a problem-solving approach that tackles the evolution of the system and the development of

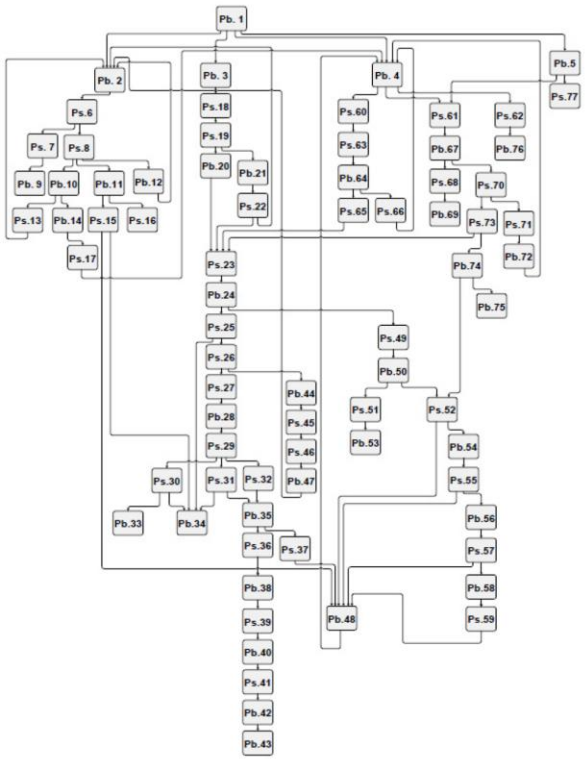



*Figure 27. Example of network of problems. (Fiorineschi et al., 2015)*

new and innovative solutions. This technique has influenced many others techniques such as systematic inventive thinking or SIT (Horowitz, 1999) and OSTM-TRIZ (Fiorineschi et al., 2015) that uses a network of problems (NoP, Figure 27) related to key TRIZ principles (Becattini et al., 2015). The use of NoP to divide a problem into smaller problems is based on the work of Khomenko about how to start the design process of complex systems. This could be summarized by several basic operations between problems and partial solutions such as [1] a problem that implies a problem, [2] a problem that can be solved and lead to a partial solution, [3] a problem that can be partially solved by a partial solution, [4] a partial solution that becomes a problem, [5] a problem that could be solved by a partial solution, and [6] a partial solution that remains a solution (Cavallucci, 2017). The design problem is therefore decomposed in subsystems, subproblems, etc. Then it is mapped using these mechanisms which will lead to a network of contradictions based on known TRIZ principles.

Algorithmic processes such as ARIZ have also evolved from TRIZ to use current improvements in available computing capabilities. They are based on contradiction matrixes and evolution laws to predict improvements on the system. Furthermore, the analysis of substance fields (SuField) allows one to address the structure of the system through an algorithmic approach and to transform it so more solutions could be obtained. These techniques are widely used in multiple business sectors, although they present limitations when creating 'quantum leaps', or in other words new and highly innovative systems for other uses. All these methods divide a complex problem or system into smaller challenges.

C-K theory was developed by Armand Hatchuel as an innovative design approach based on a series of operations between the concept space (C) and the knowledge space (K) as Figure 29 shows (Ingi, 2009). The goal was to create a method that could bring innovative solutions, independent of the field, but also capable of embracing 'crazy' or disruptive concepts. The disjunction mechanism proposes new concepts, and those are expanded within the C space. Then using conjunctions new knowledge is created. Within this process new concepts can be created or conceptualized easily.

Similarly, morphological analysis was developed by Fritz Zwicky to address multidisciplinary complex problems that cannot be quantifiable (Ritchey, 2002). With applications on many technical and industrial fields, this approach assesses the concepts through a series or cross-consistency assessments (CAA). These allow the problem to be divided so 'trivial' questions can be removed, which simplifies, and eases the design process. However, this approach does not include the possibility of addressing geometry properly. Furthermore, some multidisciplinary problems are too complex to be divided into components or parts that could be addressed by this method.

Another category of design theory refers to those **method-based** approaches developed upon scientific, algorithmic, mathematical, and statistical principles. Among those is axiomatic design (AD) (Farid and Suh, 2016), which is quite definitional. It was developed towards the beginning of the century to look at the design process from a mathematical standpoint. Based on axioms that present an independence from functional requirements and a reduction of information, a

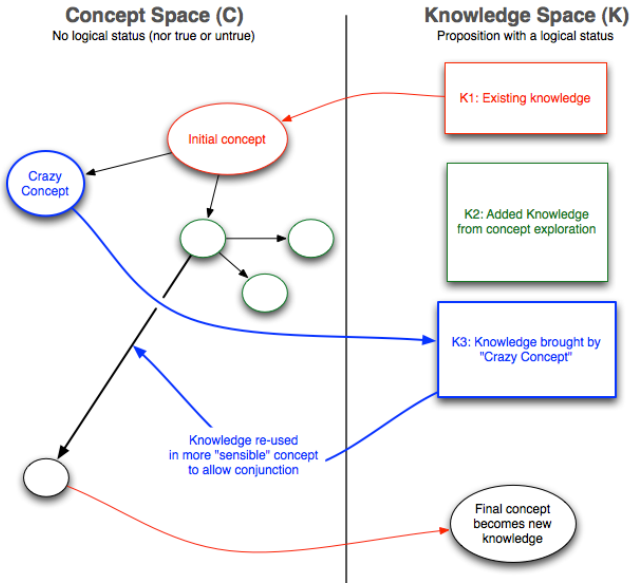



*Figure 29. Representation of C-K process, and creation of 'crazy' concepts.*

matrix is used to analyze the process to advance decision making within the system design. This method has been applied in the optimization of lifecycle product developments extensively (Gumus, 2005).

Design research as a field (Robert and Curedale, 2013) encompasses many perspectives towards understanding, and therefore improving the design process. Within it, design research methodology (DRM) uses the scientific method to refine and better define requirements, as well as to enable general improvements of the design method by bringing systematically previous results and overviews of existing research into the design process with scientific rigor. DRM brings more rigorous methods and guidelines that can be applied to the design research while enabling a more efficient design workflow (Blessing and Chakrabarti, 2009). The objective here is to address how to make a product more successful, how such product is created, and how to increase the probabilities for such product to be successful. This framework is intended to support both design and process development as well. This method includes the following phases (Figure 30): [1] research clarification (literature analysis), [2] descriptive study (empirical data), [3] prescriptive study (assumption and synthesis), and [4] descriptive study. Networks of influencing factors are developed to understand the design situation, while reference models represent both the current design situation and the impact model towards a desired situation.

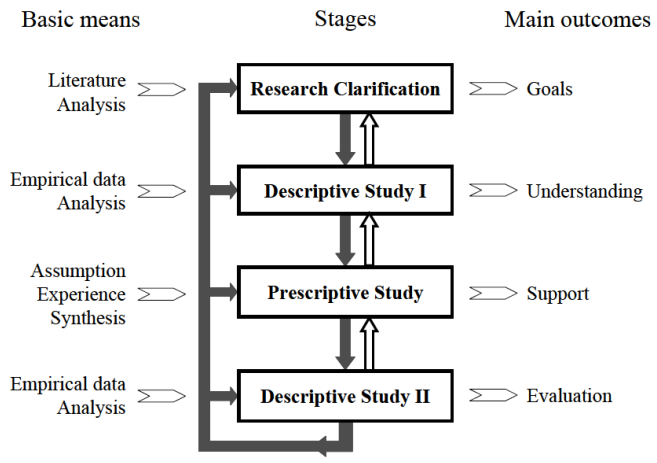



*Figure 30. DRM framework after Blessing and Chakrabarti, 2009.*

A limitation of DRM is the fact that is does not provide a design technique by itself, but rather a framework to support the design process. Its thoroughness also makes the process not very flexible towards new ideas and techniques since an impact model needs to be in sight to define a goal. This presents quite a contrast with more flexible concepts such as C-K theory towards new descriptive ideas. Table 12 presents a summary of all key aspects and characteristics relevant to this research approach as well as other related and similar techniques.

To summarize some other mathematic-driven approaches, Taguchi (Nair et al., 1992) and Six Sigma (Pande et al., 2000) methodologies are based on statistical data as well as parametric values with the objective of improving the quality and reliability of current system designs and associated processes.

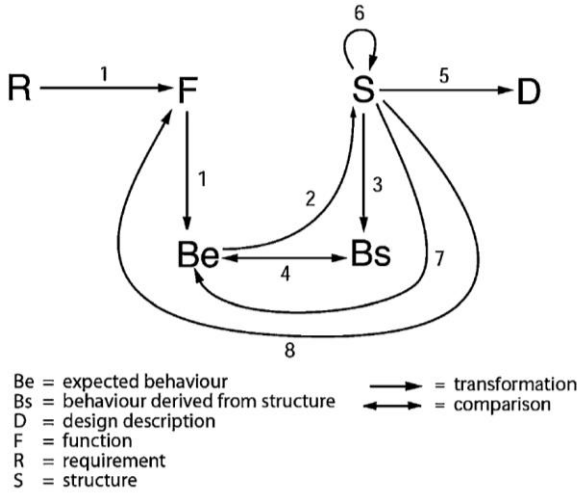



*Figure 31. Gero's FBS framework (Gero and Kannengiesser, 2014).*

Studying design theories from a **process-driven design** standpoint there are several key relevant approaches. Design ontologies such as Gero's function-behavior-structure or FBS (Gero and Kannengiesser, 2014) study how the design process actually happens, presenting later applications and implications. FBS is based on three constructs as Figure 31 shows. These include [1] function (F) which represents the teleology of the systems (what purpose of the artifact is), [2] behavior (B) or what the system does, and finally [3] structure (S) or what artifacts or systems are made of and their internal relationships. Designing within this framework is based on operations between these phases (Figure 31). Formulation (1) goes from F to B, while synthesis goes from B to S. Behavior is split into expected and derived behavior, and it is separated from the structure, with reformulation describing all iterations between them. Analysis goes from S to B, and finally documentation departures from S. While this approach explains quite well some the basics of design mechanisms, the process does not include very well other aspects such as materiality (substance) in the design process, conceptual designing, highly dynamic processes, and fast environmental changes where the design activity actually happens (Gero and Kannengiesser, 2004).

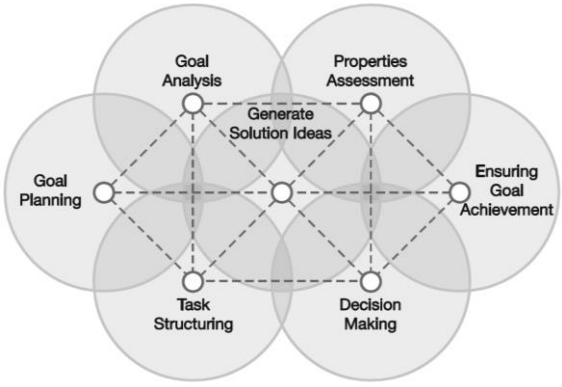



*Figure 32. Munich procedural model. (Chakrabarti and Blessing, 2014) after (Lindemann, 2009).*

The Munich Procedural Model (MPM) is a process-driven approach based on previous systems engineering and design engineer approaches as Lindeman presents on Chakrabarti's book (Chakrabarti and Blessing, 2014). Within this approach a series of key points need to be addressed for a design to be completed. This approach is mainly used for analysis purposes and problem solving. These parameters (Figure 32) are integrated within a networked scheme and they include: goal planning, goal analysis, properties assessment, ensuring goal achievement, decision making, task structuring, and solution generation (Lindemann, 2009).

Along these lines, the FORFLOW model (Chakrabarti and Blessing, 2014) was also developed to address product development planning presenting six major steps: clarification, function and structure determination, solution principles and structures, concept development, system design, production supervision, and starting point (Rodenacker, 2013).

Concurrent engineering (CE) is a design engineering methodology (Salomone, 2019) widely distributed through some complex industrial sectors such as aerospace, energy, or product design that tackles complex system design methodology. Instead of following an iterative or waterfall approach, in which each discipline or design phase happens one after the next one, (Prasad, 1995) in CE they all happen simultaneously (Figure 33). A series of interconnected models capture all key functions of a complex system architecture such as energy, structures, thermal, communication, manufacturing, electronics, etc. Once objectives, requirements, and constraints are set, engineering teams or individuals can keep assessing and modifing their design models based on feedback from other disciplines, while their changes affect others (Backhouse and Brookes, 1996a). This process continues until an optimal solution is obtained, and often it is connected to historical heritage data and other statistical information models (Eastman, 2012). This process could be extremely fast in finding a compromised solution, although due to the speed and the type of historical data being used, it could be quite problematic towards developing new innovative or disruptive solutions especially without previous and relevant heritage.

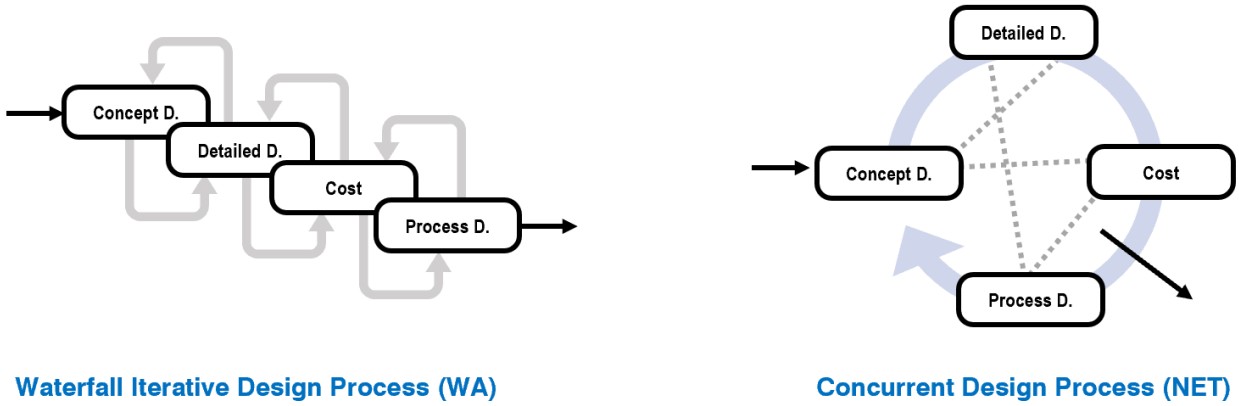



*Figure 33. Waterfall linear design process versus concurrent networked design process.*

Set-based design is a subset of this methodology. It is a highly effective and efficient concurrent design methodology developed by Toyota in the 90s (Liker et al., 1995). In Toyota's model, the emphasis on communication across teams under a matrix organization approach is as important as the technical design work itself. Ironically, the objective of this concurrent process is not a point-design solution (Figure 34), but rather a series of solutions at the system and subsystem level. Then analysis, prototyping, manufacturing, etc. as well as key negotiations with vendors and suppliers of each subcomponent allow the process to narrow down the system architecture and its components (Sobek et al., 1999). The process reduces constraints required to achieve performance and allows the system of vendors to fill the gaps. This requires a lot more work in the initial effort but ensures a more efficient process along the way. These variations allow better communication, greater parallelism, and data-based decisions, as well as workforce learning and development. For its implementation, a network of design agents is set-up for a negotiation process among them. They include the following subjects such as concept, styling, design, components, and manufacturing.

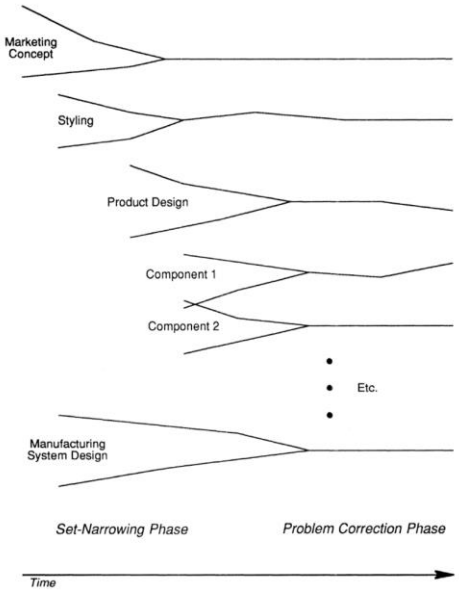



*Figure 34. Toyota's CE approach. (Liker et al., 1995)*

Regarding modern architecture design, RIBA charted the design process in four main steps (Lawson, 2014) including: assimilation, general study, development, and communication. These could be expanded to inception, feasibility, outline, scheme design, detail design, production info, bills, tender action, project planning, operation, completion, and feedback. This traditional process for architectural practice is iterative and linear, however jumps among those steps can happen across the process.

Within an **integrative** design methodology (Cross, 2008) design and analysis could be done simultaneously. Thus, the role of the designer iterates from problem to concept space continuously. Advancements during the last decades regarding data-driven techniques not only enable this approach even more, but they also open the door to a different approach to the role of the designer. There are several techniques and methodologies within this approach.

Integrated product and process design and development or IP²D² (Magrab and Magrab, 2010) was developed due to the influence of early stage decisions in the final cost of a system architecture development. The general objective of this design process (Figure 35) is to reduce cost, increase quality, and increase the process efficiency, as well as to allow the creation of more capable workforce teams in performing such processes. Team members participate in the decision process, which is information-based. These inputs are scientific, and they are based on the experience of team members. This process is concurrent in nature and presents four stages: [1] product definition, [2] concept development, [3] design and manufacturing, and [4] launch. In this process the role of technology maturation is key, affecting decisions about manufacturing, control, operations, and failure-modes.

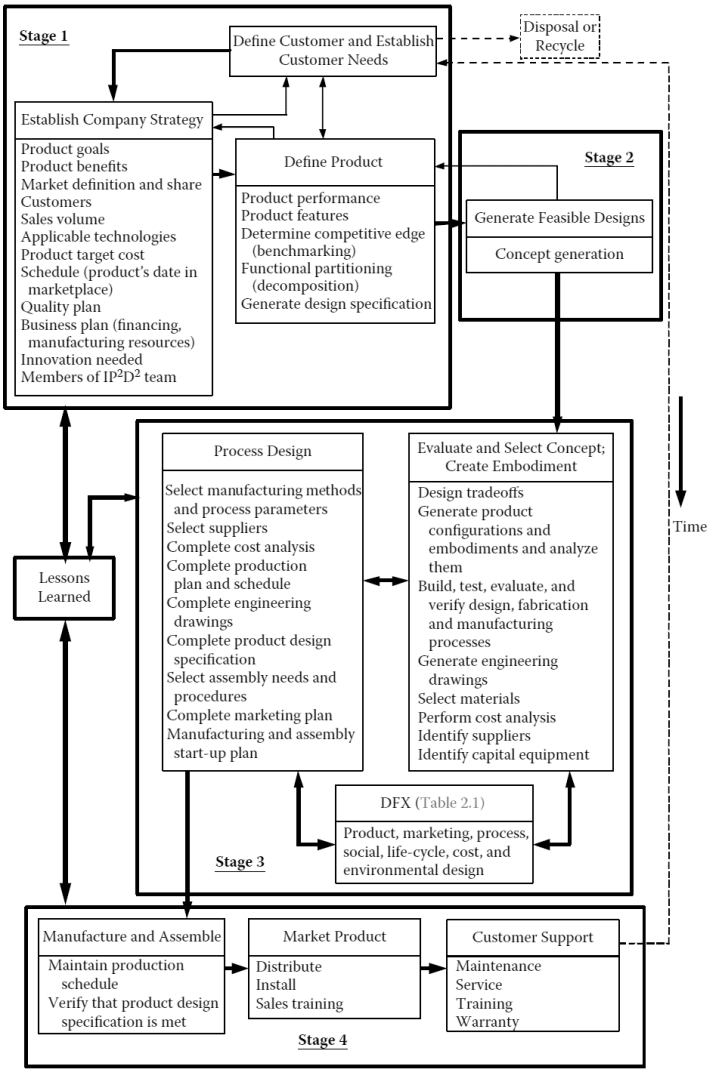



*Figure 35. Integrated product and process design and development or IP2D2 (Magrab and Magrab, 2010)*

The contact and channel approach (C&C2-A) is developed and related to the function of a system and its physical structure or embodiment (Albers and Wintergerst, 2014). This one is based on the fact that designers use geometry in early phases of the process to facilitate the design process, as well as in the importance of analyzing the geometry of current products to understand how they really work. The objective is to better understand the relationship between the function of the system, and its physical structure or geometry, emphasizing all relationships between quantitative and qualitative descriptions of such system. This approach defines three key elements: [1] channel and support structures (CSS) or physical structures, [2] working surface pairs (WSP) or interfaces, and [3] connectors (C) as Figure 36 presents. A limited number of these elements perform a given function, conforming a wirk-net. Multiples *wirk-nets* create a *work-structure*. This type of analysis optimizes the system by understanding its functionality.

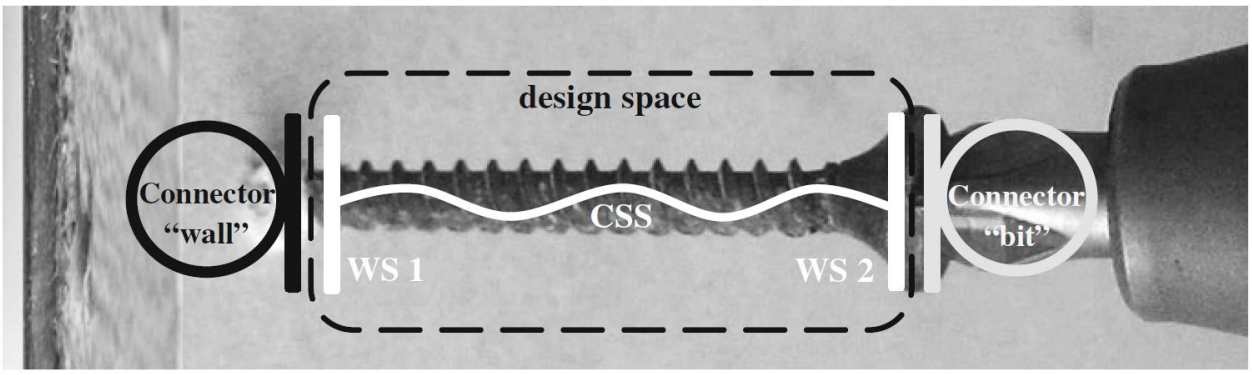



*Figure 36. C&C2-A approach with connectors, working surface pairs (WS), and channel support structures (CSS)(Albers and Wintergerst, 2014)*

Generative design (Shea et al., 2005) is a performance-based (Brandon and Kocatürk, 2009) algorithmic iterative design process. To obtain certain goals, a series of system design constraints are defined and an algorithm produces multiple outputs, geometrical or otherwise (Wu et al., 2019). Then designers can assess the relative cost of each parameter and perform variations in real time that ripple through the system. Instead of several designs, thousands of designs can be done simultaneously. In the context of hardware-based design this approach has been especially used and developed by multiple design software companies (Keane, 2018) toward the infusion of structural topology optimization techniques (Rozvany and Lewinski, 2013) in the last decades. This field is still in development, and it is aiming towards a full multidisciplinary full-cycle approach, in which the designer is key to create the proper technical questions. This approach presents the following loop design cycle: [1] performative simulation, [2] generation, and [3] evaluation (Brandon and Kocatürk, 2009).

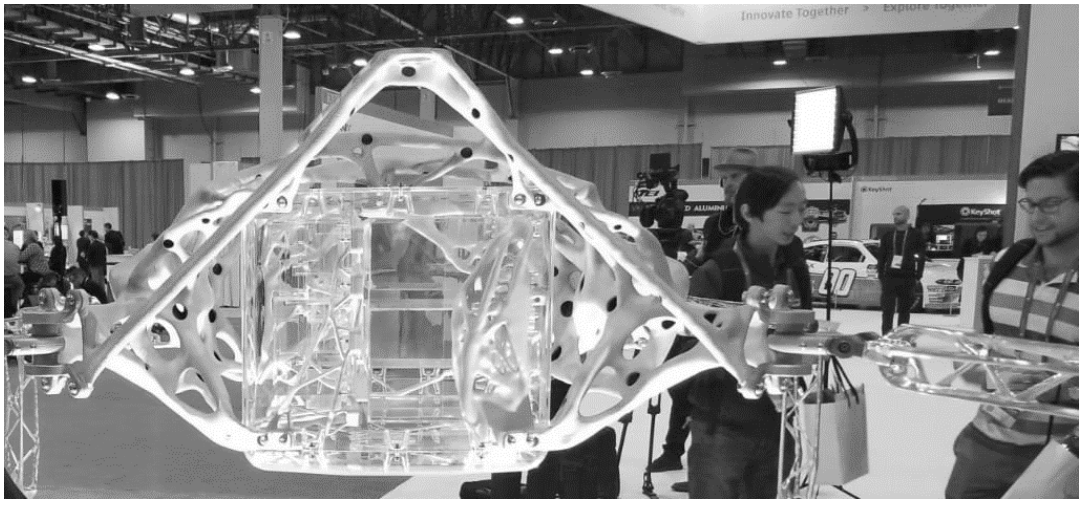


*Figure 37. Example of a generative design applied to structural optimization design.*

### *3.1.3.2. Design Engineering Tools*

Once design engineering theories and models have been studied, the evaluation of design tools available for these methods is also necessary. This analysis allows to address gaps and connection across methodologies, as well as to assess how such tools influence both design theories and practices. Table 11 shows several main groups of tools identified within this research, and Table 9 links design theories and models among these toolsets. Finally, Table 12 presents a detailed summary and analysis of theories, models, and tools.

| ID Code | Category | Time Period | *Based upon* | *Driven by* |
|---|---|---|---|---|
| To1 | **Analog** | 300 BC to Today | Manipulation of real objects | Designer |
| To2 | **Digital** | 70s to Today | Development of virtual constructs | Designer, computer power |
| To3 | **Code-based** | 90s to Today | Data and algorithm programming | Designer, models, theories, computer power |
| To4 | **Integrative** | 10s to Today | Integration of implemented artifacts, functional actuators, system digital models, dataflows | Designer, rapid manufacturing, computer power |

*Table 11. Design engineering tools by categories.*

One of the activities that highlighted the beginning of humankind culture was, or in other word sketching and marking enabled by multiple tools. With the first sketch on the wall of a cave humans started turning marks on a surface into abstract concepts (Gombrich, 1995). This was a necessary mechanism towards written language and drawing among other constructs. In the context of creating hardware-based system architectures, there is a broad spectrum of tools being used.

Among analog tools, we include those based upon the manipulation by hand of objects and markings on different mediums. These techniques can be applied to both physical and digital frameworks. These include the following.

- **Conceptual wording.** This is based in the use of words to describe, think, communicate, analyze, and discover. It is a very powerful concept design tool (Cross, 2011). The association of concepts with words allows the use of metaphors and allusions of concepts, so complex ideas can be managed without the use of geometry. Among other techniques writing, word listings, whiteboarding, storytelling, six-thinking hats, brainstorming, mind maps, and pros & cons are often used during the whole design and implementation processes. This is especially relevant during the generation of new ideas and concepts either individually or as part of a group activity.

*Figure 38. Sketching on a notebook*

- **Sketching**. This is one of the most powerful tools within the design engineering arsenal and it is also one of the oldest when designing hardware-based systems (see Figure 38) or objects. From mechanical and electrical engineering to architecture designs, sketching is a powerful method to explore, convey, and validate ideas (Ullman et al., 1990). This technique using free-hand drawing allows to create very detailed and proportionate designs (e.g., renaissance studies and engravings) as well as fast, intuitive, and insinuating drawings (e.g., modern architecture sketches). This technique also allows to describe individually, collectively, digitally, and physically complex geometries, concepts, processes, etc.
- **Technical Drawing**: The next step beyond sketching is the detailed graphical representation of complex systems using drawing tools and codes. This can be done on paper or digitally, and it allows to capture, study, and communicate geometry, organization principles, arrangements, behaviors, tolerances, implementation instructions, etc. This technique allows to represent and study architecture design, mechanical assemblies, electrical circuits, or microchip blueprints. Furthermore, this old tool also allows to assess system feasibility towards implementation.

- **Tinkering**. Using crafts, fidget toys, and other simple physical elements to conceptualize, communicate, or visualize ideas is a common and useful tool in multiple creativity environments (Nyqvist, 2016). Among many others these include techniques such as toys, clay modeling, collage, doodling, tinker toys, etc. These tools can also be used digitally within multiple software frameworks allowing virtual, augmented, and digital creations.

These techniques enable and substantiate multiple fast-paced design workflows due to the high interaction with the individual. Within the physical world, and nowadays also within digital or virtual realities, they allow a rapid feedback between the idea and the construct. Thus, they present a good platform towards [1] studying the problem, [2] exploring and inspiring new and innovative solutions, as well as [3] providing detailed technical documentation and direction.

The second group of **digital design engineering tools** is possible due to key advancement in the last decades in computer systems. These can only be used digitally within software frameworks. The most relevant are the following.

- **Computer aided design and manufacturing (CAD/CAM).** This technique uses software frameworks where the designer can create precise geometrical models and assemblies (Leondes, 2019). They also allow to capture and create solid and surface models, assembly constraints, geometrical tolerances, mechanical behaviors, materials properties, and even structural analysis, among many others. Furthermore, simulations and manufacturing studies can also be accomplished using these tools.
- **Building information modeling (BIM)** was developed originally by the military and provides a multidisciplinary design framework where the real building can be mimicked, copied and created digitally (Kensek, 2014). This technique allows to incorporate geometry, assemblies constraints, energy studies, illumination studies, construction phases, uses, schedules, behaviors, structural schemes, cost studies, technical schemes (e.g. HVAC), physical properties, and operation studies, among many others (Deutsch, 2011). Like CAD digital components and assemblies can easily be dragged and infused into the model to create more complex assemblies as well as to address parametrical studies and variations.

*Figure 39. Technical drawing by hand.*

- **Building energy modeling (BEM).** Based on BIM, this is a software framework and multi-purpose tool to assess, design, validate, and qualify building designs based on energy analysis (Brackney et al., 2018). The use of energy as a design tool or design principle for complex system designs (Cody, 2017) is a new and very interesting approach tackling both the implementation of the system as well as operations and manufacturing processes.
- **Model-based system engineering (MBSE).** These tools are mainly based on system modeling tools (Borky and Bradley, 2018), software frameworks (e.g., SysML - (Friedenthal et al., 2008), and languages like UML (Fowler, 2018). These techniques are based on abstract models describing requirements, structure, parametrics, behaviors, lifecycle phases, and risk assessments among others aspects of a complex system architecture (Fernandez and Hernandez, 2019). They can also be used beyond systems engineering applications as design engineering tools towards assessing and studying trade space options and non-geometrical relationships among subsystems and components.

All these techniques can tackle complex systems designs independently of the field of application. The number of subcomponents is virtually limitless, and they are only constrained by the computational power of the equipment. They also enable a very different design flow independently from the design scale. These tools also allow very fast changes of standpoint, detail definition, and time phases. Thus, they can address both details (e.g., bolt definition) and overarching architecture design principles. Their use influences design models tackling the development of families of components, as well as their modifications and changes over time through a simplification of the process (less time) and a reduction of the cost (less workforce and tools). While all previous tools could use digital, virtual, and software frameworks, they do not necessarily need to use mathematical-driven principles in their workflow. However, there is a complementary family of techniques based on the use of **codes and mathematical** models. Several of these groups can be identified as it follows.

- **Math-driven tools** such as mathematic and multi-physics programing languages (Tiller, 2012), software frameworks (Wolfram and Illinois), 1999), and other programmable tools allow designers to quickly create analytical and descriptive models for design, study, assessment, and validation of multiple topics across all design phases of a system design and implementation. These models are the foundation of computer analysis and provide broadly available techniques such as finite elements analysis or FEA (Bathe, 2006).
- **Code-based tools.** While previous tools provide a framework with predefined computer functions, this toolset is based on the creatition of an algorithm or model from scratch by the user (Pierce and Pennsylvania), 2002). These code-based software tools (Figure 40) allow to prototype, design, and deliver custom models supporting hardware-based designs, as well as the delivery of fully finished mechatronics and other robotic systems that are enhanced by software.

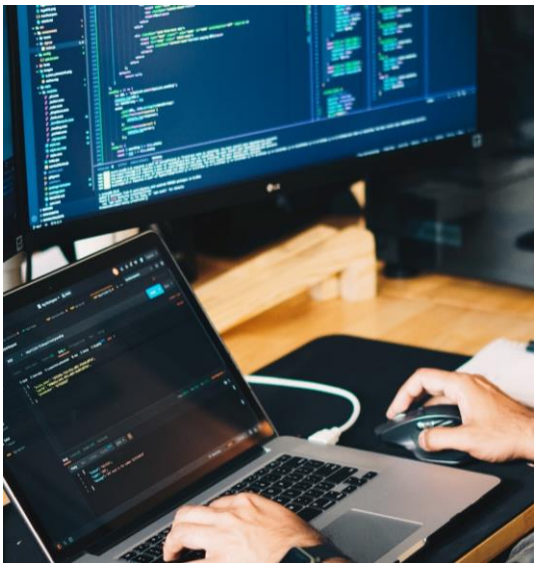


*Figure 40. Generic coding tools broadly used currently.*

- **Parametric tools** are a subset of previous techniques. They use physical models, CAD, math, code, and MBSE to address and study the creation of multiple design solutions, families of solutions, and variations regarding the same system. These present multiple dimensions, such as 1D (code), 2D (drawings and plates), 3D (volumes) (Kimura, 2001), and 4D (movement). They also allow to create catalogs and manage data regarding constraints, requirements, features, comparisons, etc. from multiple and different views.
- **Generative tools** are algorithmic and parameter-based design and assessments software tools (Shea et al., 2005). They are a subset of computational design tools (Autodesk, 2020), using mathematical laws and algorithms to create variations of parametrical variables. These tools are ideal to explore trade space options as well as to create designs based on analysis inputs such as FEA models. Optimized topologies (Rozvany and Lewinski, 2013) for additive manufacturing are also a good example. These techniques allow to reduce mass and assembly components, as well as simplify manufacturing among other benefits derived from such a new design workflow.

*Figure 41. Example of a mechanical assembly redesigned using generative design tools (Autodesk, 2020).*

Finally, the integration of all the above tools enables two main families of techniques that can create fully functional **integrative** models of a complex hardware-based systems architectures across all design phases, virtually and physically.

- **Digital Twin (DT)**. These are cyber-physical frameworks (Figure 42), models, and tools that allow to design, evaluate, and assemble digitally complex system architectures before they are physically built (Tao et al., 2019). This technique is increasingly present across the production lifecycle (Jones et al., 2020). The concept of system twin started with NASA during the Apollo program, and enable the creation of an engineering copy for testing, analysis, etc. Today DT is a digital copy of the real system addressing design, optimization, metrology, validation, and manufacturing, as well as other new areas of data-driven services (Boje et al., 2020). This framework is completed with a feedback loop once the system is implemented which allows real-time optimization and system performance tuning afterwards. Applications include manufacturing (e.g., industry 4.0), architecture (Farsi et al., 2019), etc.

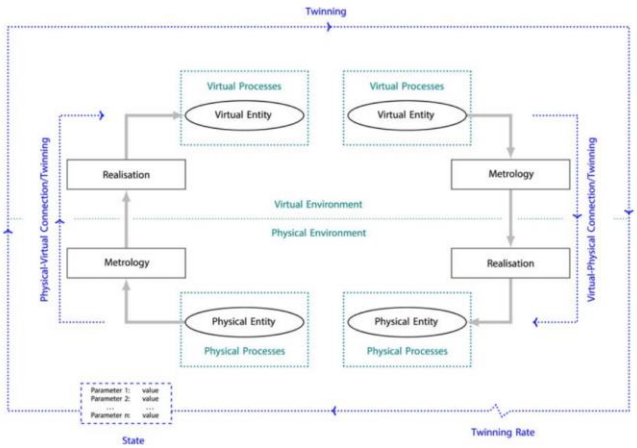



*Figure 42. Physical to virtual process and back (Jones et al., 2020).*

- **Rapid Prototyping** (RP) is based on the use of rapid manufacturing and prototyping tools such as 3D printing, breadboards, etc. (Kamrani et al., 2016) to assess, study, design, implement, validate, and communicate functional hardware-based systems. They are applied to mechanical systems, mechatronics, electrical systems, robotics, and software systems, among many others. This technique offers a fast approach towards implementing 'functional-enough' systems, however improvements in advanced manufacturing such as 3D printing allows this technique to produce faster and more fully defined functional components.

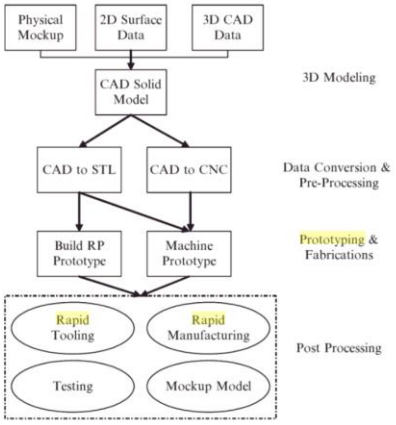



*Figure 44. Rapid prototyping workflow (Kamrani et al., 2016).*

*Figure 44. Examples of rapid prototyping tools.*

These final techniques bring a unique perspective upon the latest advancements in manufacturing and digitalization. They also connect the beginning (e.g., concept development) and the end of the design process (e.g., manufacturing, advance simulation, etc.) from the start of the design activity. These two approaches reshape the traditional evolution of the design process due to different reasons such as [1] concept design and manufacturing simultaneous start, [2] cyber-physical connections among disciplines and models (networked design process), [3] capture, reuse, and comparison of complex design work, [4] simultaneous system optimization, and finally [5] real-time adaptable complex design workflows.

### 3.1.3.3. Literature Review Matrix of Design Theories and Models

Based upon the introduction in section 3, Table 12 presents a summarized review of multiple design engineering methodologies and theories from a literature review and practice standpoint. A brief description of their key characteristics is provided as well as an evaluation of several aspects regarding hardware-based complex systems such as:

- **Foundation**. This is a summarized description of key principles and characteristics.
- **Design phase.** What phases are addressed by this approach? Basic design phases are numbered as it follows: [1] planning, [2] problem study, [3] concept design, [4] embodiment design, [5] detailed design, [6] analysis, [7] optimization, [8] testing and validation, [9] documentation, [10] implementation, [11] delivery, [12] marketing, [13] operations, [14] decommission, [15] product or process recycling (Seider et al., 2016) (Haik et al., 2010). A colored scale is presented based upon these phases and the level of structure and detail of each method (Figure 45).

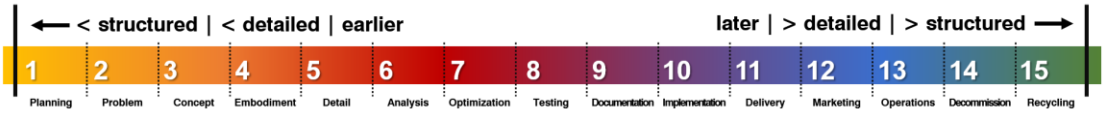



*Figure 45. Color scale addressing time, detail, and structure level of a design method.*

- **Geometrical information**. Does the design methodology allow to use, manage, author, and edit geometrical information (e.g., volumes, shapes, sections, tolerances, and other graphical constructs)?
- **Qualitative / quantitative (Qt./Ql.)**. Can the method be used to qualify, qualify, or both multiple design parameters?
- **Scope**. Can the design methodology handle point-design solutions (PDS), families of point-design solutions (FDPS), development process (DEV), continuous designs (CONT), or a combination (COMB) of them?
- **Adaptable**. Is the design approach adaptable to the challenge at hand through a flexible (FLE) or networked process (NET)? Does it present a more rigid approach such as linear methodologies (LI), iterative cycles (ITE), waterfall (WA), or spiral approaches (SPI)? Figure 46 presents graphically these types of methodologies.

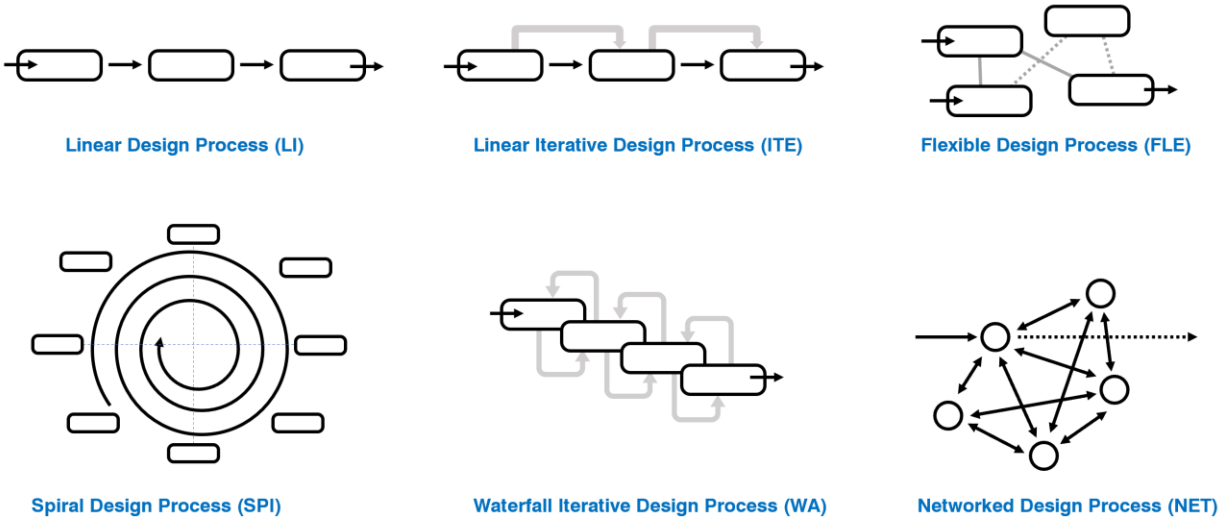



*Figure 46. Type of design processes and methodologies within the engineering design literature review.*

- **Perspective.** Is the design method based upon a discrete disciplinary standpoint (DD) or a synergetic disciplinary approach (SA)? Is it based on a 'divide-and-conquer' approach discretizing disciplines and subsystems, or can it tackle simultaneously multidisciplinary problems?
- **Optimization**. Does the approach allow a parametric optimization of the system design?
- **Tool platform**. What type of tool and technique does the design approach enable or support? This includes: [1] mathematical models, [2] drawings, [3] CAD/PLM, [4] graphs, [5] Eng. models, and [5] documents or text.
- **Reference**. List of most relevant technical references and professional practice inputs that were reviewed.

| Theory/Method | Foundation | Date | Phase | Geo. | Qt./Ql. | Scope | Adapt. | Pers. | Opt. | Tools | References |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Design Theories & Models*** | | | | | | | | | | | |
| **DE1 Classical D.** | Driven by art and architecture. | 1-2nd | 1-9 | Yes | QT/QL | | | | | | (Roth, 1994) |
| **Aristotle's** | Production based on thinking and making. | 300 BC | 1,2,3 | Yes | QL | N/A | N/A | SA | No | Concept | (Koskela et al., 2014) (Natali, 2013) |
| **Vitruvius's** | Strength + utility + beauty. Architecture driven theory. | 50 BC | 1,3,4, 5,9 | Yes | QT/QL | PDS | N/A | SA | No | Drawing | (Pollio, 2018) (Benevolo, 1977) (Roth, 1994) (Vitruvius, 2012) |
| **DE2 Renaissance D.** | Driven by art and architecture. | 15-17th | 1,3,4, 5,9 | Yes | QT/QL | | | | | | (Evers et al., 2015) |
| **Alberti's** | Component + outline + position. Architecture driven theory. | 16th | 1-5, 9-12 | Yes | QT/QL | PDS | N/A | SA | No | Drawing | (Lewis, 2020) (Benevolo, 1977) (Roth, 1994) (Williams et al., 2010) |
| **DE3 Enlighten. D.** | Driven by art, architecture, and objects. | 17-18th | 1-5, 9-12 | Yes | QT/QL | | | | | | (Etlin, 1996) |
| **Boulle & Ledoux** | Impossibility and abstraction as design tool. Architecture driven theory. | 18th | 25,9, 12 | Yes | QT/QL | PDS | N/A | SA | No | Drawing | (Benevolo, 1977) (Roth, 1994) (Williams and Ostwald, 2015) |
| **Violet-le-Duc** | Product + Process. Based upon use of materials & technologies. Architecture and product driven. | 19th | 2-,9, 12 | Yes | QT/QL | PDS | N/A | SA | No | Drawing | (Viollet-le-Duc and Hearn, 1990) (Roth, 1994) |
| **DE4 Modern D.** | Driven by product and process. | 20th | 1-12 | Yes | QT/QL | | | | | | (Frampton, 2020) |
| **Bauhaus** | Product and architecture driven. Practicality, innovation, constraints (cost), and production. Education. | 20s | 1-12 | Yes | QT/QL | PDS FPS | N/A | SA | No | Drawing | (Droste, 2015) (Lockwood, 2010) (Benevolo, 1977) |
| **Le Corbusier** | Systematic. Architecture driven. Combination of key elements. Scientific analysis. | 30s | 1-12 | Yes | QT/QL | PDS FPS | N/A | SA | Yes | Drawing Math | (Jencks, 2000) (Benton and Cohen, 2019) (Roth, 1994) |
| **DE5 Descriptive D.** | Explore, concept, evaluation, detailing. | 50s -10s | 1-6, 10 | Yes | QT/QL | | | | | | (Cross, 2008) |
| **Jones** | Design by drawing for continuous changes. | 70s | 2,3, 4,5 | Yes | QT/QL | PDS FPS | FL | SA | No | Drawing | (Jones, 1992; Lawson, 2014) |
| **French** | Need – concept – embodiment - details. | 90s | 2,3, 4,5 | Yes | QT | PDS | LI | DD | No | Drawing | (Shukla and Krishnan, 2016) |
| **Morris Asimow** | Multiple design operations are connected on a loop (general and specific information – deign operation – evaluation – outcome – next operation). | 70s | 1-10 | Yes | QT/QL | PDS FPS DEV | ITE | DD SA | Yes | Drawing CAD Math | (Dieter and Schmidt, 2012) (Asimov, 1976) |
| **Evan's Spiral** | Critical design spiral based upon iterative and heuristic principles. Based on designer preferences. Parametric. | 50s | 1-10 | Yes | QT | PDS | FL | SA | Yes | Drawing CAD Math | (Vossen et al., 2013) (Singer et al., 2009) (Evans, 1959) |
| **DE6 Prescriptive D.** | Analysis, synthesis, development, communication. | 60s -10s | 1-15 | Yes | QT/QL | | | | | | (Dym, 2013) |
| **Cross** | Problem analysis, conceptual design, scheme embodiment, detailing (objective tree, function assessment, performance, quality functions, morphology, weighted objectives, value engineering). | 40s -10s | 1-10 | Yes | QT | PDS | LI | DD | No | Drawing Math | (Cross, 2008) |
| **Pahl & Beitz** | Prescriptive method. It is based on system interrelationships (energy, signal, material). Analysis, synthesis, and development. Enabling complex system design. | 60s -10s | 1-13 | Yes | QT | PDS FPS | LI | DD | Yes | Drawing CAD Math PLM | (Pahl et al., 2007) (Cross, 2008) (Hubka, 2015) (Haik et al., 2010) |
| **VDI 2211** | Problem analysis, sub-problems, suitable sub-solution, overall solution | 80s | 1-6 | N/A | QT | PDS DEV | WA | DD | N/A | Drawing CAD Math | (Pahl et al., 2007) (Cross, 2008) |
| **March's** | Solution-focused approach. Synthesis-driven. Production, deduction, induction. | 80s | 1-8 | Yes | QT/QL | PDS DEV | NET | SA / DD | N/A | Drawing Math | (Cross, 2008) |
| **Archer's** | Systematic design thinking. Training, programming, data collection, synthesis, | 80s | 1-12 | Yes | QT | PDS DEV | ITE | DD | N/A | Drawing CAD | (Cross, 2008) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | development, communication. | | | | | | | | | Math | |
| **Suireg's** | Need, collection, analysis, synthesis, selection of feasible concept, simulation, optimization, implementation, test & evaluation. | 80s | 1-15 | Yes | QT | PDS FPS | LI | DD | Yes | Drawing CAD Math | (Zyl et al., 2007) (Suireg, 1981) |
| **Ullman** | Planning, concept generation, concept evaluation, product generation, product evaluation, documentation, communication | 40s -10s | 1-12 | Yes | QT | PDS FPS DEV | LI | DD | Yes | Drawing CAD Math PLM | (Ullman, 2009) |
| **Darke** | Briefing, analysis, synthesis, evaluation. | 70s | 1-6 | Yes | QT | PO | WA | SA | Yes | Drawing Math | (Lawson, 2014) (Darke, 1979) |
| **Pugh** | Concept, embodiment, detailing. It enables structure, judgement, and management of design. It uses a decision matrix. | 80s | 1-8 | Yes | QT | PDS DEV | ITE | DD | N/A | Drawing CAD Math | (Pugh, 1986) |
| **Seider & Lewin** | Process design. Synthesis, analysis, & evaluation. | 10s | 1-15 | No | QT | PO | WA | SA | Yes | Math | (Seider et al., 2016) |
| **Eggert** | Systematic Parametric. Formulation, generation, analysis, evaluate, optimization. Re-specification between 1 and 3. Re-design between 2, 3, 5. | 40s -10s | 1-12 | Yes | QT | PDS FPS DEV | LI | DD | Yes | Drawing CAD Math PLM | (Eggert, 2010) |
| **DE7 Design Thinking** | Cognitive, strategic concept development of complex problems. Understand, improve, apply. | 50s -10s | 1-6 | Possibl e | QT/QL | | | | | | (Curedale, 2013) |
| **Arnold's** | Creative engineering. Analysis, evaluation, and synthesis. Four areas of development: Incremental innovation, radical innovation, lower cost, and more salability. | 50s | 1-14 | Yes | QT/QL | PDS FPS DEV | ITE | SA | Yes | Drawing CAD | (Arnold and Clancey, 1959) |
| **Design Thinking Method** | [1] empathize with the problem (understanding and observation), [2] synthesis, [3] ideations, [4] prototyping and [5] testing. Wicked problems, problem framing, solution-driven, co-evolution of solution-problem, abductive reasoning. | 50s -20s | 1-14 | Yes | QT/QL | PDS FPS DEV | ITE | SA | Yes | Drawing CAD TEXT WEB | (Curedale, 2013; Greene et al., 2017; Plattner et al., 2010) (Brown, 2009) (Kolko, 2010) |
| **Human-centered Design** | HCD. Human perspective in every problem-solving step. Human skills, flexibility, knowledge, and creativity. | 80s | 1-14 | Yes | QT/QL | PDS FPS DEV | ITE | SA | NO | Drawing CAD Text | (Rosenbrock, 1989) (LUMA Institute, 2012) (Hancke et al., 1990) |
| **User Centered Design** | UCD. goals, user, environment, task, and workflows of a product. | 10s | 1-14 | Yes | QT/QL | PDS FPS DEV | ITE | SA | NO | Drawing Text WEB | (Vredenburg et al., 2002) (Norman and Draper, 2018) (Rubin et al., 2008) |
| **DE8 Innovative D.** | Innovation is a drive or a central objective. | 50s -20s | 1-7 | No | QT/QL | | | | | | |
| **Brainstorming** | Facilitated idea generation. Gathers ideas (including wild one) from participants, allowing people to build ideas upon someone else's. These ideas are combined synergistically to allow a collective development of new options. | 60s -20s | 1-2 | Possibl e | QT/QL | PDS FPS DEV | ITE | SA | NO | Drawing Text | (Osborn, 1993) (Wilson, 2013) (Rawlinson, 2017) (Hawkins, 2019) |
| **Synectics** | Group problem solving. It uses metaphors as an idea development technique to make them feasible. | 50s | 1-4 | N/A | QL | PDS FPS DEV | ITE | SA | NO | Drawing Text WEB | (Gordon, 1961) (Prince, 2012) (Wake, 2000) |
| **TRIZ** | 40 Systematic principles. Contradiction matrix, system evolution laws and SuField. Algorithmic nature. | 40s -20s | 1,2,3, 6,7 | No | QT | PDS FPS | ITE | SA | Possible | Text Math | (Altshuller, 1984) (Cavallucci, 2017) (Schöfer et al., 2015) (Montecchi and Russo, 2015) |
| **OSTM-TRIZ** | Network of problems (NoP) contradictions and solutions based on TRIZ principles. | 40s | 1,2,3, 6,7 | No | QT | PDS FPS | NET | SA | Possible | Text Math | (Fiorineschi et al., 2015) (Becattini et al., 2015) (Cavallucci, 2017) |
| **SIT** | Systematic inventive thinking. TRIZ-driven. Close box approach (def. problem space). | 90s | 1,2,3, 6,7 | No | QT | PDS FP | ITE | SA | Possible | Text Math | (Horowitz, 1999) (Blokdyk, 2018) |
| **C-K Theory** | Concept and knowledge space dialog. Design formalization independent of domains. Provides a framework to innovate | 10s | 1,2,3, 6,7 | Yes | QT/QL | PDS FP DEV | NET | SA | Possible | Drawing Text Math | (Hatchuel et al., 2004) (Hatchuel and Weil, 2002) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | within the design process. "Crazy" concepts are part of the process. | | | | | | | | | | (Massotte and Corsi, 2015) (Salustri, 2014) |
| **Morphology** | Complex problem solving based on cross consistency assessments (CCA). | 90s -10s | 2-5,7 | No | QL | PDS FP DEV | NET | SA | Possible | Drawing Text Math | (Ritchey, 2002) (Álvarez and Ritchey, 2015) (Flanagan et al., 2013) |
| **DE9 Method-driven Design** | Design theory based on specific methodology or enhancements. | 90s -20s | 1-7 | No | QT | | | | | | |
| **Axiomatic Design** | Axiom-driven approach for the design process. Algorithm-based. Complex problems are divided, and 'trivial' problems removed. | 90s | 2,3,4 | No | QT | PDS FP DEV | ITE | DD | Yes | Text Math | (Farid and Suh, 2016) (Park, 2007) (Saha, 2014) |
| **DRM** | Design research methodology provides a research framework using scientific techniques to support design and requirement definitions. | 10s | 2-5,7 | No | QT | PDS FP DEV | WAT | DD | Yes | Text Math | (Blessing and Chakrabarti, 2009) (Cash et al., 2016) |
| **Six Sigma** | Statistical method to improve product and process design quality. Multidisciplinary. | 90s -10s | 2-5,7 | No | QT | FP DEV | ITE | DD | Yes | Text Math | (Pande and Holpp, 2001) (Snee and Hoerl, 2003) |
| **Taguchi** | Statistical method to improve design quality, and variation studies. | 90s -10s | 2-5,7 | No | QT | FP DEV | ITE | DD | Yes | Text Math | (Nair et al., 1992) (Roy, 1990) |
| **DE10 Process-driven Design** | Design approach defined by the process. | 90s -20s | 1-14 | Possible | QT/QL | | | | | | |
| **FBS** | Function, Behavior, Structure ontology-driven design theory. | 10s -20s | 1-6 | No | QT/QL | PDS FP DEV | ITE | DD | Yes | Text Drawing Math | (Gero, 2011) (Gero and Kannengiesser, 2004) (Chakrabarti and Blessing, 2014) (Kan and Gero, 2017) (Vermaas and Dorst, 2007) |
| **MPM** | Munich Procedural Model. Problem-solving analysis. Seven steps. | 30s -10s | 1-14 | Yes | QT/QL | PDS FPS DEV | NET | SA | Yes | Drawing CAD Process | (Chakrabarti and Blessing, 2014) (Lindemann, 2009) |
| **FORFLOW** | Product design process. Clarification, function and structure, solution principles and structures, concept development, system design, production. | 30s -10s | 1-14 | Yes | QT/QL | PDS FPS DEV | ITE | SA | Yes | Drawing CAD Process | (Rodenacker, 2013) |
| **Concurrent Design** | Concurrent engineering networked process. Multiple disciplinary models connected for simultaneous design. | 90s -20s | 2-11 | Possible | QT | PDS DEV | NET | DD | Possible | Drawing CAD Process | (Eastman, 2012) (Backhouse and Brookes, 1996a) (Prasad, 1995) (Salomone, 2019) (Frey et al., 2011) |
| **Set-Based Design.** | Concurrent engineering method by Toyota. Broad design parameters left opened longer and converging gradually. Agent Interaction Diagrams. | 90s | 1-14 | Possible | QT/QL | FPS DEV | NET | DD/SA | Yes | Text Drawing Process | (Singer et al., 2009) (Liker et al., 1995) (Sobek et al., 1999) (Maulana et al., 2017) |
| **RIBA** | Architecture design based upon assimilation, general study, development, communication. Jumps across steps. | 70s | 1-10 | Yes | QT/QL | PDS FPS | WA | DD | Yes | Drawing CAD | (Lawson, 2014) |
| **DE11 Integrative** | Design + Analysis simultaneously | 10s -20s | 1-8 | Yes | QT | | | | | | |
| **IP²D²** | Concurrent engineering process. Data-driven. Four stages: product definition, concept development, design & manufacturing, launch | 10s | 1-14 | Yes | QT/QL | PDS FPS DEV | ITE | SA | Yes | Drawing CAD Process | (Maqrab and Maqrab, 2010) (Rufe, 2013) |
| **C&C2-A** | Function-based design. *Wirk-structure* made of *wirk-nets* made of CSS (channel and support structures, (WASP) working surface pairs, and (C) connectors for design & analysis. | 10s | 1-7 | Yes | QT/QL | PDS FPS | NET | SA | Yes | Drawing CAD Math | (Albers and Wintergerst, 2014) (Chakrabarti, 2019) |
| **Generative Design** | Algorithmic iterative design process, that produces multiple outputs based on constraints and towards certain goals. | 90s -20s | 2-7 | Yes | QT | PDS FPS | NET | DD | Yes | CAD Math | (Shea et al., 2005) (Brandon and Kocatürk, 2009) (Keane, 2018) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Designers can tune them. | | | | | | | | | | (Wu et al., 2019) (Marcus, 2014) |
| DE12 Evolutionary | See section 3.3 for details. Design + Analysis + Selection concurrently. This field also relates to systems engineering. | 10s -20s | 1-7,13 | No | QT | PDS FPS | NET | | | | |
| Evolutionary Design | Adaptable and innovative system design. Design optimization, algorithms, CAD. | 90s -20s | 1-7 | Yes | QT | PDS FPS | NET | DD | Yes | CAD Math Process | (Braha et al., 2006) (Bentley, 1999) (Hingston et al., 2008) |
| ***Design Tools*** | | | | | | | | | | | |
| To1 Analog | Physical, digital, or virtual. Scaled. | BC -20s | 1-15 | Yes | QT/QL | | | | | | |
| Concept-Words | Writing, word lists, storytelling, six-thinking hats, brainstorming, mind maps, etc. are used to describe concepts, processes, etc. | BC | 1-15 | No | QT/QL | PDS FPS DEV | FLE | SA | Partial | Concept | (Lawson, 2014) (Lees-Maffei, 2013) |
| Sketching | Powerful tool that can used to described complex geometries, concepts, process. Individual or collective technique. | BC | 2-9, 12 | Yes | QT/QL | PDS FPS DEV | FLE | SA | Partial | Drawing Concept | (Wang, 2002) (Cross, 2008) (Ullman et al., 1990) |
| Technical Drawing | Detailed graphical representation of complex system capture and communicate geometry, order, arrangements, behaviors, tolerances, instructions, etc. | 19th -20s | 2-9 | Yes | QT/QL | PDS FPS DEV | FLE | SA | Partial | Drawing Concept | (Ullman, 2009) (Dym, 2013) (Goetsch et al., 2015) |
| Tinkering | Using crafts, fidget toys, and other simple physical elements to conceptualized, communicate, or visualized ideas (e.g., clay modeling, collage, doodling, etc.) These tools are both physical and digital. | BC | 3-4 | Yes | QT/QL | PDS | FLE | SA | No | Concept Model | (Nyqvist, 2016) (Fishel, 2018) |
| To2 Digital | Digital, virtual, without scale. | 80s -20s | 1-14 | Yes | QT | | | | | | |
| CAD/CAM | Computer aided design and manufacturing uses software frameworks to create precise geometrical assemblies (solid modeling). | 80s -20s | 2-,9, 12 | Yes | QT/QL | PDS FPS DEV | NET | DD | Yes | Drawing Model Math | (Leondes, 2019) (Rao, 2004) (Soenen and Olling, 2016) (Sendler and Wawer, 2008) |
| BIM | Building information modeling provides a multidisciplinary design framework for geometry, phases, behaviors, and physical properties. Imitates digitally a real system. | 90s -20s | 1-7, 9-14 | Yes | QT/QL | PDS FPS DEV | NET | SA | Yes | Drawing Model Math | (Deutsch, 2011) (Kensek, 2014) (Kamrani et al., 2016) |
| BEM | Building energy modeling, is a software framework and multi-purpose tool to assess, design validate, qualify building designs based on energy calculations. | 10s -20s | 1-7, 9-14 | Yes | QT | PDS FPS DEV | NET | SA | Yes | Drawing Model Math | (Brackney et al., 2018) (Clarke, 2007) (Hemsath and Bandhosseini, 2017) |
| MBSE Design | Model-based system engineering tools could be use as design engineering tools, towards assessing and studying trade space options, etc. | 90s -20s | 1-15 | No | QT | PDS FPS DEV | NET | SA | Yes | Model Math | (Fernandez and Hernandez, 2019) (Borky and Bradley, 2018) (Dori, 2016) (Friedenthal et al., 2008) |
| To3 Code-based | Digital, math-based, without scale. | 18th -20s | 1-15 | Yes | QT | | | | | | |
| Math-driven tools | Math tools, such mathematic programing languages, software frameworks, and other allow designer to quickly create mathematical models for design, study, assessment, and validation. | 18th -20s | 1-15 | No | QT | PDS FPS DEV | NET | DD | Yes | Model Math | (Tiller, 2012) (Wolfram & Illinois), 1999) (Chaturvedi, 2010) (Bathe, 2006) (Cottrell et al., 2009) (Koutromanos, 2018) |
| Code-based tools | Software code-based tools allow to prototype, design, and deliver tools to support hardware-based design | 90s -20s | 1-15 | No | QT | FPS DEV | NET | SA | Yes | Model Math | (Barr & Massa, 2006) (Pierce and Pennsylvania), 2002) (Bradley, 2011) |
| Parametric | These tools use physical models, CAD, Math, MBSE, etc. models to address and study multiple solutions, variations, etc. | 70s -20s | 1-8 | No | QT | PDS FPS DEV | NET | SA | Yes | Model Math | (Kimura, 2001) (Dickerson and Mavris, 2016) (Corser, 2012) (Woodbury, 2010) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Generative Tools** | Algorithmic and parameter-based design and assessments tools. Multiple solutions are created based on parameters and laws. Ideal for trade space exploration. | 10s -20s | 2-7 | Yes | QT/QL | PDS FPS DEV CONT | NET | SA | Yes | Model Math | (Shea et al., 2005) (Wu et al., 2019) (Agkathidis, 2016) (Abruzzo et al., 2007) (Gengnagel et al., 2011) (Leach and Yuan, 2018) (Menges and Ahlquist, 2011) (Rodrigues et al., 2015) |
| **To4 Fully Functional** | Physical + analog, + digital + virtual. Scaled approach. | 50s -20s | 2-12 | Yes | QT/QL | | | | | | |
| **Digital Twin (DT)** | Cyber-physical models, framework, and tools that allow to design, evaluate, and assemble digitally a complex system before they are built and close the feedback loop once the system is implemented. | 10s -20s | 2-9 | Yes | QT/QL | PDS FPS DEV CONT COMB | NET | SA | Yes | Drawing Model Math | (Jones et al., 2020) (Yi et al., 2020) (Tao et al., 2019) (Evangeline, 2020) (Farsi et al., 2019) (Boje et al., 2020) |
| **Rapid Prototyping** | Use of rapid manufacturing and prototyping tools such as 3D printing, breadboards, etc. to assess, study, validate, and communicate functional mechanical, mechatronics, electrical, robotics and software systems, among others. | 50s -20s | 3-8, 10-12 | Yes | QT/QL | PDS FPS DEV COMB | FLE | SA | Yes | Concept Model | (Cooper, 2001) (Chua et al., 2010) (Bártolo et al., 2012) (Liou, 2007) (Rayna and Striukova, 2016) (Kamrani and Nasr, 2010) |

*Table 12. Design engineering theories and methodologies.*

### 3.1.4. Conclusion

After conducting an extensive literature review, which is summarized on Table 12, engineering design theories, models, and tools have been studied and evaluated from ancient times to the current digital state-of-the-art. Several points presented in section 3.1.3.3 were used to study those techniques and models from the perspective of a hardware-based system architecture design. This analysis allowed to identify several key gaps among them that are consistent across most techniques and models. These gaps have been identified based on the following points:

- **Global stressors** presented in section 2 influence both system architecture design and system design process. For instance, the capability of a technique to enable or simplify the design process towards the creation of disruptive ideas represents a criterion to assess the capability of such technique.
- **Complex systems related**. Design techniques present gaps and enhancements capabilities towards addressing complex systems design. These topics are used to assess their capabilities across the design lifecycle.
- **Hardware-based systems related**. Similarly, it is crucial to evaluate if the technique or method is specifically capable of handling hardware designs across all lifecycle design phases.
- **Design process efficiency**. Finally, key gaps in these methods regarding their contribution to the efficiency of the process and the result are another aspect to be assessed and evaluated.

These criteria points are relative. Thus, their characteristics, capabilities, and applications are the final contribution to assess these conclusive remarks based on all points identified in section 3.1.3.3. In this final assessment, such inputs are combined to identify all the most relevant and overarching gaps as it follows:

- **Synergy**. All design theories tackling the development of complex systems present a 'divide-and-conquer' approach. In general, a complex problem is subdivided into subsystems, disciplines, topics, and components which are tackled individually. Later these are integrated and optimized. The iterative nature of linear, waterfall, or even more flexible workflows comes often out of the need to find convergence across disciplines while considering workforce and organizational resources. This is especially relevant across prescriptive methodologies (Cross, 2011), while some methodologies such as DRM (Blessing and Chakrabarti, 2009), FBS (Gero and Kannengiesser, 2014), and MPM addressing the design process as a whole. These address multiple phases of any disciplinary practice at hand, but they still divide the complexity of a design challenge hierarchically. In opposition to contemporary models, pre-modern theories such as Vitruvius's approach (Vitruvius, 2012) address complex problems as a whole dividing their complexity not from a component-standpoint but rather from a perception standpoint. On the other side, state-of-the-

art (SOA) fully functional or integrative methods such as DT, generative design, and RP present synergetic disciplinary frameworks. Within those, multiple disciplines get combined address simultaneously software and hardware design topics. These tool-based methodologies or workflow also merge conceptual designs towards system manufacturing and production. Thus, it seems than modern design theories, as heirs of discreet and empirical perspectives coming from the industrial revolution (Deane and Deane, 1979), do not have a synergetic model capable of embracing complexity from a holistic standpoint. However, while classical tools partially have such approach, SOA design techniques already implement a feasible design context enabling the study of system complexity in detail and from a multidisciplinary perspective.

- **Qualification**. While classical techniques allow to tackle both quantifiable and qualifiable aspects, modern and contemporary techniques are focused on quantitative parameters, especially those based on mathematical principles (Farid and Suh, 2016). When contemporary techniques such as morphological analysis (Ritchey, 2002) can handle complex non-quantifiable challenges, they tend to do it from a non-geometrical standpoint.
- **Continuity and linearity**. Design models capable of tackling both complex geometries and quantifiable parameters present across all different groups of design theories some form of iterative linear process. This is something directly related to previous synergy and qualification gaps. Therefore, these methods tend to set a specific design objective that is concentrated on a point-design or unique solution. Therefore, tackling the generation of a family of solutions is not necessarily part of the workflow of these techniques beyond a small subset of parametrized solutions. On the other hand, some complex system designs such as those within the category hardware-based system architectures need a multidisciplinary approach. Such approach requires interconnection and refinement among different phases of the design process. In summary, these design techniques do not look at the design workflow from a continuous perspective, but the tools (e.g., CAD and BIM) present such capability. Among them, generative design techniques preliminary present such continuous workflows but do not have an applied multidisciplinary capability yet.
- **Adaptable**. While in general design tools are quite adaptable to changes, especially those better suited for fast-paced environments, the associated design process does not present the same level of adaptability across multiple design phases. Once a concept synthesis has been obtained only the modification of parameters allows rapid changes, but major divergences and changes in the system design require extensive efforts within the process.
- **Innovation**. In general terms, Table 12 presented techniques that especially address the development of innovative ideas such as design thinking (Lockwood, 2010). These have a structured approach towards highly detailed technical design of complex system geometries. However, those prescriptive techniques with highly structured and organized methodologies are not flexible enough to easily infuse new ideas at different phases of the design process. These methods allow to gradually bring more definition into the system design once a concept synthesis has been obtained. Such synthesis requires though a thorough analysis of requirements. Nevertheless, major design changes entail to restart the process all over again. These design methodologies do not have a clear and specific approach towards the value and use of heritage as part of the design process. Techniques such TRIZ divide previous related solutions using system design principles, but the relationship between innovation and heritage is not a part of the process.

In conclusion, there is a huge potential for new tools and methods capable of synergetic and multidisciplinary design outcomes. Design engineering techniques present powerful capabilities and proven approaches, but there is not a clear theory or methodology adapted to embrace these gaps across methodologies and eventually system characteristics.

These conclusions are part of the starting point for this research activity as the following chapters will elaborate. They represent in combination with the upcoming section a foundational baseline, since they address both proven capabilities and critical gaps in tools and techniques used in the design, implementation, and eventually operations of complex hardware-based system architectures.

## 3.2. Systems Engineering Paradigms

### 3.2.1. Approaches and Categories

The second field in this literature review within the context of a multi-domain research is systems engineering (SE) methods and theories. While previous theories and techniques presented a clear design purpose dealing with geometrical relationships, systems engineering is about describing relationships and creating models that represent the system abstractly. Systems engineering is about complexity (Borky and Bradley, 2018), or in other words it is about "internal and external interactions, structures, behaviors", and connections across components and system parameters. Typical areas of study and practice of these methods are systems and enterprises over their lifecycle (Buede, 2009). These methods have the objective of applying quantitative methods to "analyze, design, optimize, measure, document, communicate, and control" such constructs (Borky and Bradley, 2018). In this section, 'systems' and 'systems architecture' have the same definition that was provided in chapter one even if there can be differences and nuances among authors.

Similarly to the previous section, the main objective here is to understand, study, and compare multiple key theories, models, languages, and tools that are used generally in SE studies. There are specifically relevant in Model-based systems engineering (MBSE) as well as system of systems engineering (SoSE). Table 15 summarizes and provides organization and context to this extensive literature review in the context of hardware-based system architectures.

The origins of SE are in the military as a process to create requirements for military systems (Badiru, 2019). Since then, numerous standards and developments such as ANSI, MIL, ISO, CMMI, EIA, COSYSMO, etc. have been created and used by governments, agencies, organizations (e.g., INCOSE), and professional groups worldwide. The scope of systems engineering methods is very broad and includes requirements, operations, risk management, industrial processes, manufacturing, systems control, construction, architecture, aerospace, and energy, among many more.

Table 13 shows under this perspective how this literature review is organized in a series of categories including theories, standards, models, tools, languages, and frameworks. These groups represent an overall summary regarding how SE techniques are applied and used, with an emphasis on the development of complex systems. Section 3.2.2 presents the morphology of systems, while section 3.2.3 introduces a study of systems lifecycle from a SE perspective. Then section 3.2.4 introduces an overview of the SE practice landscape until today that is organized by topics, scope, and capabilities, summarized in Table 15. Finally, section 3.2.5 presents key findings and conclusions.

| ID Code | Categories | Sub-Family Code | Description | Driven by |
|---|---|---|---|---|
| SE1 | **Theories / Standards** | SE1.1 Historical<br>SE1.2 Standards | This includes general SE approaches, historical perspectives, and overall standards regarding the foundation and bases for the practice of SE. | **Theory**<br>**Community** |
| SE2 | **Models / Process** | SE2.1 Document-Based<br>SE2.2 Lifecycle-based<br>SE2.3 Cross-cutting | This group relates to SE overall processes and constructs addressing SE lifecycle phases, SE engineer roles, and interactions with stakeholders, among others. | **Practice** |
| SE3 | **Tools** | SE3.1 Documents<br>SE3.2 Diagrams<br>SE3.3 Matrixes<br>SE3.4 Analysis<br>SE3.5 Graphs<br>SE3.6 Charts<br>SE3.7 Code | They include specific technical methodologies and toolsets defining the practice of SE such as diagrams, computer applications, etc. | **Systems engineer** |
| SE4 | **Languages** | SE4.1 Modeling<br>SE4.2 Systems<br>SE4.3 Mathematical | These are mathematical and ontological languages and codes for the practice of system engineering. | **Standards**<br>**Practice** |
| SE5 | **Frameworks** | SE4.1 Software<br>SE4.2 Systems Eng.<br>SE4.3 MBSE<br>SE4.4 Architecture<br>SE4.5 General design | These are related to both methodologies and tools integrated within specific frameworks. These refine and enable both practices and methods. | **Practice**<br>**Capabilities**<br>**Software**<br>**Capabilities** |

*Table 13. Categories of systems engineering modeling tools, resources, and practices.*

### 3.2.2. Components of Complex Systems

As stated in section 1.8, the concept of a system is based upon a series of elements and components working together. Thus, a study of multiple SE approaches must address both the hierarchy and structure of those systems. This is done under a hardware-based perspective, and as such the following components (Kossiakoff et al., 2020) were identified:

- **Context.** This relates to the environment and framework where the system performs its function or purpose.
- **Interfaces.** Any system presents interfaces with other systems, its contexts, subsystems, and components based on signals, data, materials, or energy exchange.
- **Subsystems**. These could be considered as "a major portion of the system" performing a "subset of the overall function" (Kossiakoff et koal., 2020). These tend to be organized by disciplines (e.g., thermal, mechanical, design, etc.), functions, and management reasons, among others depending on the culture of the organization.
- **Component / Assembly**. These are lower-level entities and middle-level aggregations of subsystems.
- **Subcomponents / Subassembly**. They are composed of several parts performing elementary functions.
- **Parts**. These are elements at the lowest level of a system. They perform no significant function system-wise besides becoming elements within other components. They can be understood as the building blocks of the system.

Furthermore, there are several other levels above the system level such as:

- **Family of system (FoS)** is a group of systems with common characteristics.
- **System of systems (SoS)** is made of multiple independent systems that are integrated altogether.
- **Enterprise (SoSE)** includes multiple SoS given a general structure.

| Systems | | | | | |
|---|---|---|---|---|---|
| Communications systems | Information systems | | Material processing systems | | Aerospace systems |
| **Subsystems** | | | | | |
| Signal networks | Databases | | Material preparation | | Engines |
| **Components** | | | | | |
| Signal receivers | Data displays | Databases programs | Power transfer | Material reactors | Thrust generators |
| **Subcomponents** | | | | | |
| Signal amplifiers | Cathode ray tubes | Library utilities | Gear trains | Reactive valves | Rocket nozzles |
| **Parts** | | | | | |
| Transformer | LED | Algorithms | Gears | Couplings | Seals |

*Figure 47. Hierarchy of complex systems after Kossiakoff et al. 2020,*

Within the literature there are multiple types of parameters or variables related to the practice of SE. Some authors (Liu, 2015) identified several basic parameters categories as it follows:

- Design-independent parameters (DIPs), which are related to external attributes and context characters that affect the performance but not the system design.
- Design-dependent parameters (DDPs) that define the system itself and can alter its performance.
- Technical performance measurements (TPMs), which quantify DDPs.

### 3.2.3. SE Lifecycle Phases and Applications

In the previous section Table 10 and Figure 45 presented different phases in the development of complex system architectures. While these phases do not necessarily happen in the same order, there is an overall increase in details and structure along the way. The following Table 14 introduces a correlation between design engineering phases and systems engineering development lifecycle phases (Badiru, 2019; Buede, 2009; Liu, 2015; Valacich et al., 2017). In the same table there are references organized by phases with regards to several key aspects such as:

- **Systems Value** (Badiru, 2019). The use of SE pursues to improve and manage the outcome of the design process enabling these characteristics across the system design process: [v1] affordability, [v2] practicality. [v3] desirability, [v4] configurability, [v5] modularity, [v6] reliability, [v7] desirability, [v8] maintainability, [v9] testability, [v10] transmissibility, [v11] reachability, [v12] quality, and [v13] agility.
- **Application domains** are specific areas and fields of application for SE practices across multiple system lifecycle phases. Some of the most relevant domains are the following: [d1] requirements, [d2] physical, [d3] allocated resources, [d4] interface, [d5] integration, [d6] qualification, [d7] human factors, [d8] ergonomics, [d9] vehicle design, [d10] product design, [d11] process design, [d12] interactions, and [d13] risk, among others.

| | DE Phases | SE Phases | *Theories* | *Models* | *Tools* | *Languages* | *Frameworks* | *Function / Task* | *Key domain* | *Key Systems Value* |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Planning** | *Planning* | x | | x | x | | **Analysis** | *D1-D11* | *V1-13* |
| 2 | **Problem Def.** | *P. Analysis*<br>*Operational needs* | x | x | x | x | x | ● Requirement<br>● Mission<br>● Functional | *D1* | *V1-13* |
| 3 | **Concept Design** | *Logical concept Des.*<br>*Physical concept Des.*<br>*Concept exploration* | | x | x | | | **Design**<br>● Alternative evaluation<br>● Decision making<br>● Technical Performance<br>● Optimization<br>● Concurrent Eng.<br>● Economy | *D1-D4, D7-D11* | *V1-8* |
| 4 | **Embodiment Design** | *Preliminary Des.*<br>*Development* | | x | x | x | | | *D1-D4, D7-D11-12* | *V1-8* |
| 5 | **Detailed Design** | *Detailed. Des*<br>*Development* | x | x | x | | x | | *D1-D4, D7-D11-12* | *V1-8. V12,* |
| 6 | **Analysis** | *Analysis* | | x | x | x | | | *D2-D5, D13* | *V6, V13* |
| 7 | **Optimization** | *Optimization* | | x | x | x | | | *D1-D4, D7-D11* | *V1, V6* |
| 8 | **Testing** | *Testing* | | x | x | x | | | *D2, D6* | *V6, V9* |
| 9 | **Document.** | | | x | x | x | | | *D5, D6* | *V10* |
| 10 | **Implementation** | *Integration*<br>*Construction*<br>*Production* | x | x | x | x | | | *D4, D12* | *V13* |
| 11 | **Delivery** | *Deployment*<br>*Qualification*<br>*Validation* | x | x | x | | | **Implementation**<br>● Simulation<br>● Management<br>● Control | *D6* | *V6, V12* |
| 12 | **Marketing & Communication** | | | | x | | x | | *D10* | *V1* |
| 13 | **Operations** | *Operations*<br>*Refinement*<br>*Maintenance* | | x | x | x | | | *D11* | *V8, V11, V13* |
| 14 | **Decommission** | *Phase-out*<br>*Retirement* | | x | x | | | | *D11* | *V5, V2* |
| 15 | **Recycling** | | | | x | | | | *D11* | *V1* |

*Table 14. This table presents relationships across engineering design and systems engineering lifecycle phases.*

## 3.2.4. System Engineering Methods Literature Review

This section starts with the assessment of theories, models, tools, languages, and frameworks regarding the practice of SE towards the definition of complex systems. Systems engineer roles are not part of the scope of this research. The definition of 'system' and 'system architecture' has been already established in section 1.8.

### *3.2.4.1. Theories and Standards*

Historically, the beginnings of **generic systems engineering** as a concept is identified by many authors as a memo created by Bell Telephone Laboratories in 1948 (Buede, 2009), with also some initial books about this discipline appearing during the late 1950s and 60s. The RAND corporation introduced the concept of system analysis in the 1940s (Liu, 2015). Arthur David Hall established in 1962 that SE presented 5 phases (Buede, 2009; Hall, 1962): [1] systems and program planning, [2] exploration planning, [3] development planning, [4] development, and [5] current engineering (with an operational system).

Further developments in the practice and approach towards this emergent discipline were accomplished by government institutions such as DoD and NAVY for the development of complex weapons systems. The NAVY program for evaluation and review technique (PERT, 1958) was a manufacturing scheduling method based on activities, optimum costs, and other schedule criteria (Liu, 2015) setting a historical heritage for all SE methodologies.

With the influence of commercial practices such as Hughes Aerospace, NASA's Apollo program started the development of SE at the largest scale (Liu, 2015; NASA, 2007). Under NASA's perspective SE and project control techniques were connected with an approach based upon: [1] SE processes, [2] technical management processes, and [3] product realization through a series of well-defined linear phases across formulation and implementation (Figure 48). Stakeholder expectations, cost, system validation, and verification are key aspect of the NASA SE flow. Such SE practice aimed historically towards a baseline design (NASA, 2007) that is performed within a hybrid waterfall networked process as Figure 49 shows. This is based on key aspects such as: [a] mission objectives, [b] operational objectives, [c] mission success

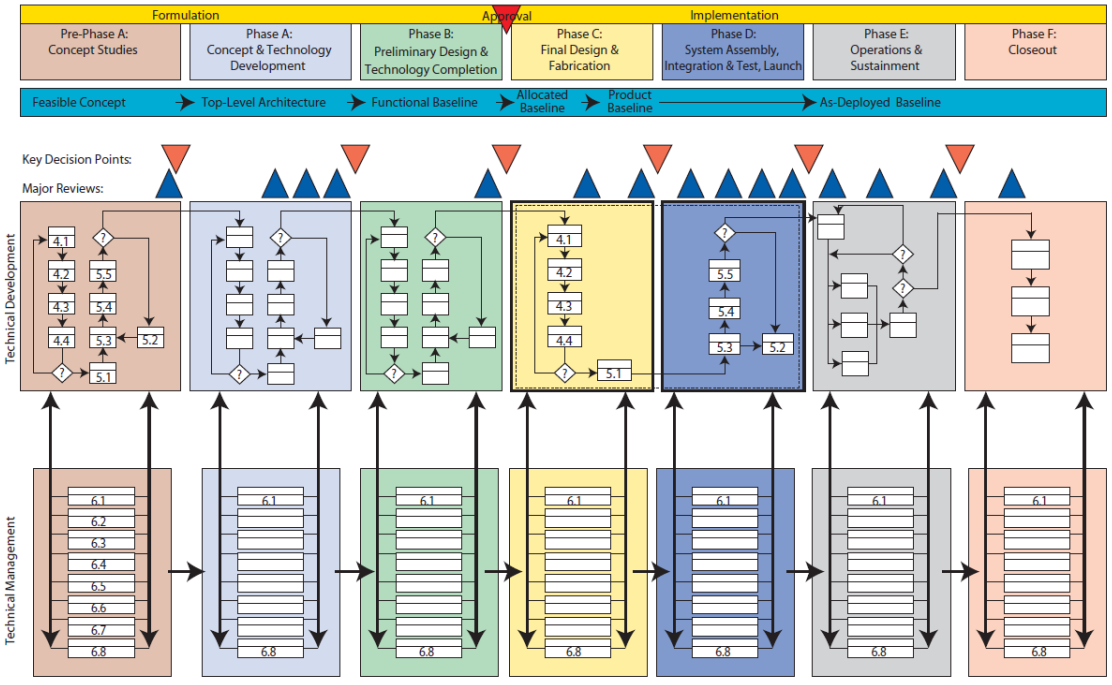



*Figure 48. Conceptualization of NASA project life-cycle process and phases (NASA SE Handbook, 2007)*

criteria, [d] general requirements, [e] decomposition constraints(functional, logical, behavioral, design), [f] trade studies, [g] design studies, [h] product breakdown structure, [i] derived and allocated requirements, and [j] con-ops (Hitchins, 2008).

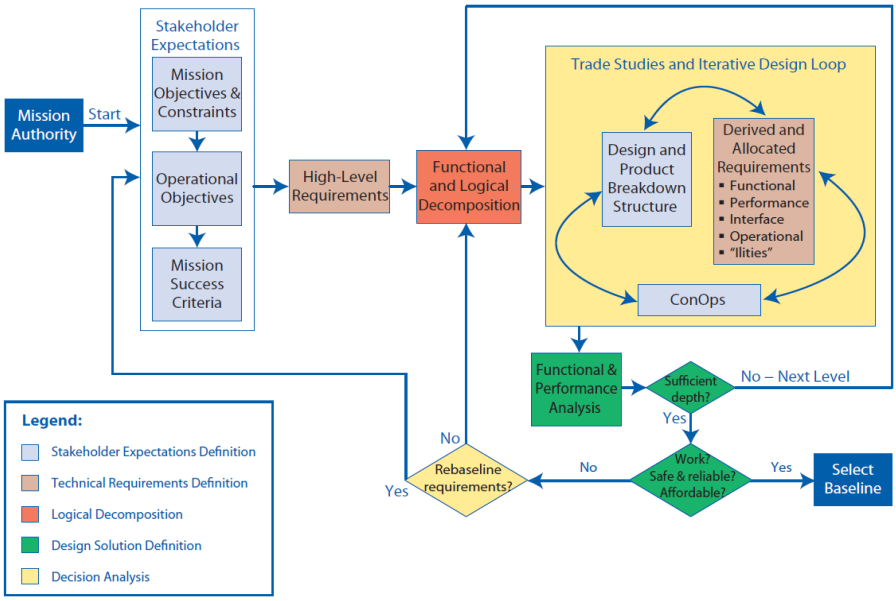



*Figure 49. NASA system engineering design process (NASA SE Handbook, 2007).*

Lifecycle and role-based approaches such as those developed by INCOSE (INCOSE, 2015) since 1990s influenced both the practice and education of SE practices. INCOSE (Liu, 2015) has developed an evidence-based SE competency framework (SECF) based on several key groups, such as: [1] SE management, [2] professional development, [3] core SE principles, [4] integration, and [5] technical competencies. They present multiple proficiency levels, as well as an organized competency description including areas, descriptions, purposes, and roles. This theoretical framework allows individuals and organizations to apply a systematic practice of SE. The INCOSE framework also covers the systems engineering management plan (SEMP), which is a document covering process planning, requirement analysis, functional analysis, synthesis, systems analysis, and control analysis. Furthermore, it also addresses key technologies and risks, while it also describes the integration of systems engineering efforts, activities, schedule, and metrics for the process.

The increasing complexity of systems across industries brings the notion of SE management in the context of SoS (Badiru, 2019). Systems of systems engineering (SoSE) is still a field under development but a few key domains are recognized in the literature review (Boardman and Sauser, 2006; Luzeaux et al., 2013) such as: autonomy, belonging, connectivity, diversity, emergence, resilience, and fluidity. Among relevant authors, the idea of an open system approach (Jamshidi, 2011) is critical emphasizing synergism, self-government, reconfiguration, symbiosis, and modularity. The concept of SoSE brings closer the notion of biological guiding principles (Sauser et al., 2010) in the current cutting-edge practice of systems engineering.

Within such practice of SE the creation of **SE standards** has also been notoriously relevant. In the 1990s the ANSI/EIA-632 protocol (Badiru, 2019) was based on a work breakdown structure (WBS) approach with potential consequences for some fundamental processes such as: [1] acquisition, [2] technical management, [3] system design, [4] product realization, and [5] technical evaluation. Among other historical SE standards (INCOSE, 2015) we can identify these:

- MIL-STD 499 (1969).
- IEEE 1220 (1999).
- ISO/IEC/IEEE 15288 (2002) tackles systems engineering, software engineering, and SE lifecycle among many other standards within ISO/IEC/IEEE such as 24765, 29148, 42010, 15289, 15939, 16085, 16326, 24748-4.
- ISO 31000 for risk management.

- ANSI/AIAA G-043A-2012e for operational concept documents.
- XMI (metadata interchange) is an OMG metamodeling standard for exchange information via XML (OMG, 2015).
- MOF (meta-object facility) is an OMG model-driven engineering standard for CORBA architecture (OMG, 2019).

The constructive systems engineering cost model (COSYSMO) was developed upon the basic five ANSI/EIA-632 principles extending them to another 33 activities. The COSYSMO standard was developed by Ricardo Valerdi at USC (Valerdi et al., 2003) to tackle life cycle phases, processes, and system models. Basic steps described by this standard are (Figure 50) the following: [1] conceptualize, [2] development, [3] operational test and evaluation, [4] transition to evaluation, [5] operations, maintenance, enhancement, and finally [6] system replacement and dismantling.

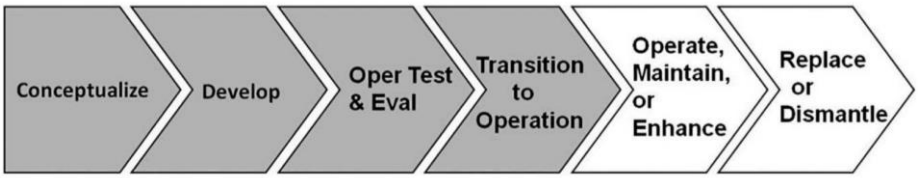



*Figure 50. COSYSMO standard phases (after Badiru, 2019).*

The capability maturity model integration (CMMI) developed by ISACA was release in 2002 under version 1.1 as a process-based improvement protocol for software development. CMMI identifies a series of maturity levels for general processes such as: [1] initial, [2] managed, [3] defined, [4] quantitatively managed, and [5] optimizing as Figure 51 shows (Godfrey, 2008). Under this approach there are three areas being addressed as both product and service, such as [a] development (CMMI-DEV), [b] establishment and management (CMMI-SVC), and [c] acquisition (CMMI-ACQ), (CMMI Product Team, 2018). Core processes of this approach tackle configuration, planning, risks, and system analysis among many more. In this context the capability maturity model (CMM) also relates to the level of formality, detail, and optimization of the process.

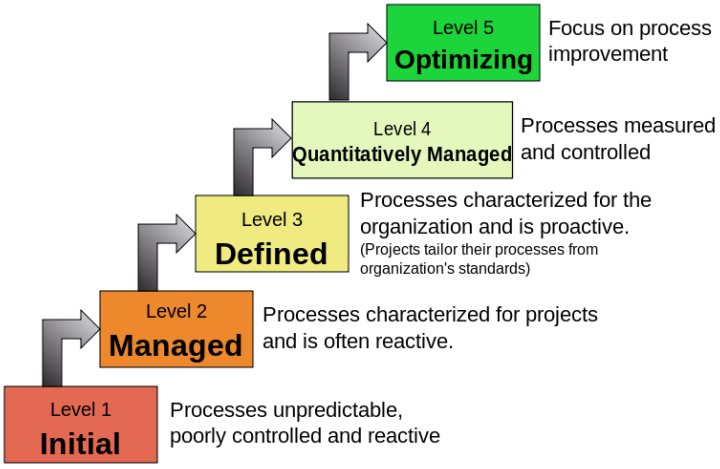



*Figure 51. CMMI maturity levels (Godfrey, 2008).*

In the current age of information, the modern practice of SE beyond professional protocols is based on a constantly evolving body of knowledge and practice which is defined by challenges and technical capabilities. Modern theories (Kossiakoff et al., 2020) are based on systems lifecycle and a series of functional system elements such as [1] signals, [2] data, [3] materials, and [4] energy. This contemporary lifecycle presents the following steps: [a] concept development, [b] engineering development, and [c] post-development. Thus, DoD, ISO, IEC, and NASA standards present the bases for an elaborated decision and SE development process as Figure 52 shows (Kossiakoff et al., 2020; Lapham et al., 2014).

At the same time, computer capabilities developed during the last decades enable the idea of system thinking (Senge, 2010) as a way to look at the SE practice from a "deep" analysis standpoint (INCOSE, 2015). This line of thought includes aspects, such as [a] system dynamics (e.g., multidisciplinary simulation languages), [b] action research and soft systems (e.g., attitudes, procedures, etc.), and finally [c] pattern discovery (e.g., taxonomics, standards, templates, etc.).

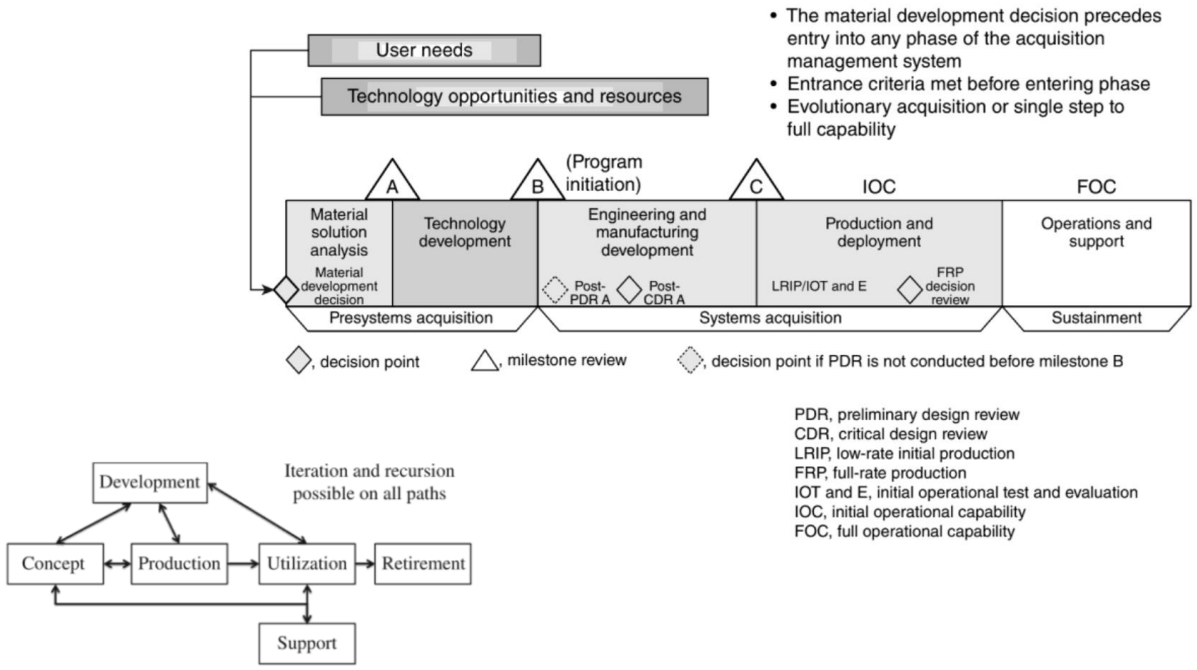



*Figure 52. Right. Acquisition management system of DoD, after Kossiakoff (Kossiakoff et al., 2020; Lapham et al., 2014). Left. ISO/IEC lifecycle after INCOSE (INCOSE, 2015).*

### *3.2.4.2. Models and Processes*

The study of SE models and processes from a traditional and **document-based** standpoint highlights several key processes in the systems engineering practice (Kamrani and Nasr, 2010; Martin, 1996) that can be summarized as the following documents. They are related to multiple systems engineering phases:

- *Systems engineering management plan* (SEMP) is a document describing SE planning activities after the concept design phase. It reviews and assigns updates and functions: configuration, requirement definition, verification etc.
- *Systems engineering master schedule* (SEMS) includes event-based milestones, relationships, and criteria.
- *Systems engineering detailed schedule* (SEDS) is a detailed task-oriented document complementing SEMS.
- *Work breakdown structure* (WBS) is a classic multidisciplinary document based on a Gantt that captures aspects such as services, data, resources (e.g., hardware, software, infrastructure), workforce, cost, and work effort control.
- *Technical performance measurement* (TPM) is a progress assessment document used in risk mitigation.
- *Requirement documents* (RD) are capturing documents (often driven by corporate or sector cultures) that define systems objectives and thresholds, such as: [1] *request for proposal* (RPF) that captures stated requirements by the customer, [2] *requirement analysis* (RA), [3] *affinity diagrams* (AD) for large volumes, and [4] *house of quality* (HOQ) diagrams which allows as a tool to turn requirements into design and user specifications (Liu, 2015).
- *Functional flow block diagram (FFBD)* is a multilevel and step-by-step graphical document that shows the functional operational structure of a system (Badiru, 2019) as a sequence of operations. As Figure 53 shows (Defense Acquisition University, 2005; Manske, 2008) this document includes the following elements: [1] functional graphic

block, [2] function number, [3] functional connection, [4] functional flow directions, [5] numbering changes, [6] stopping criteria. This principles are broadly used in business process mapping (BPM) as a SE practice to define structure, responsibility, and products (Darwish, 2011) for enterprises and business.

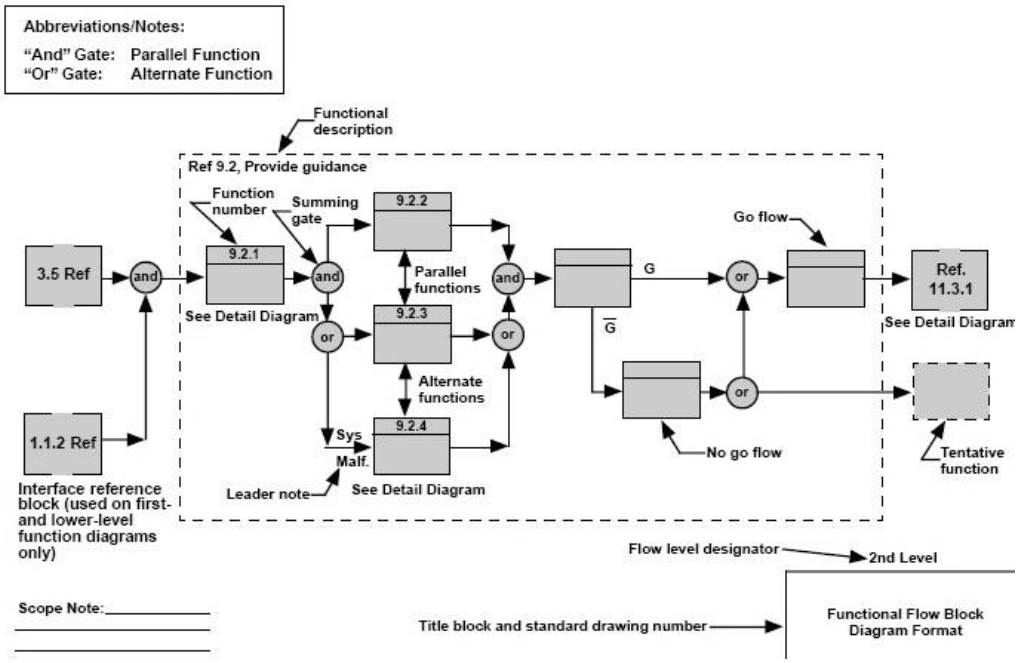



*Figure 53. Functional Flow Block Diagram, after Manske (Defense Acquisition University, 2005; Manske, 2008).*

However, the practice of SE since the 1950s has provided a series of workflow models as a set of processes and frameworks to develop a system from a **lifecycle-based** perspective. The historical top-down systems engineering (TDSE) process starts with a deep meta-system analysis of the system and its context including other related or interconnected systems (Buede, 2009). Such analysis then leads to the definition of the system and all its parts or components as configuration items (CI). Figure 54 shows how the definition of the system is linked to a verification effort at each level between design and testing, and across all steps within this traditional approach.

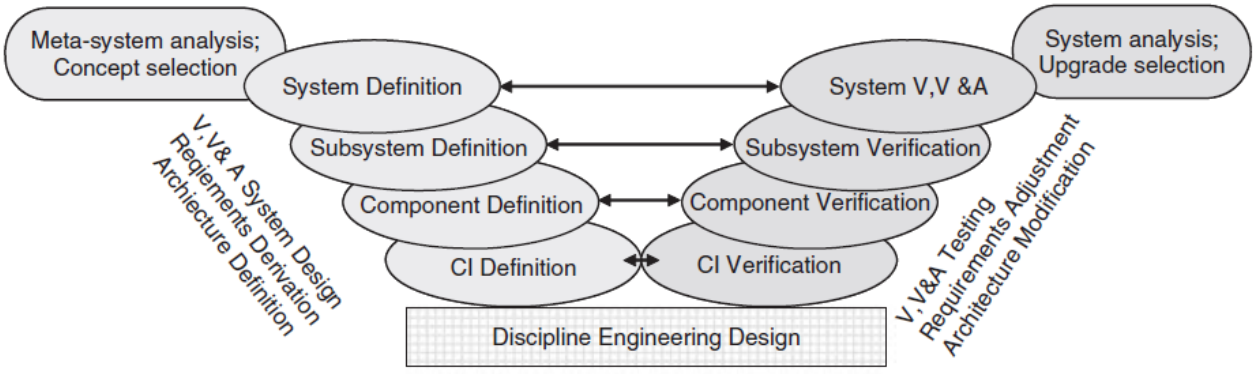



*Figure 54. Historical top-down systems engineering (TTDSE) process (Buede, 2009).*

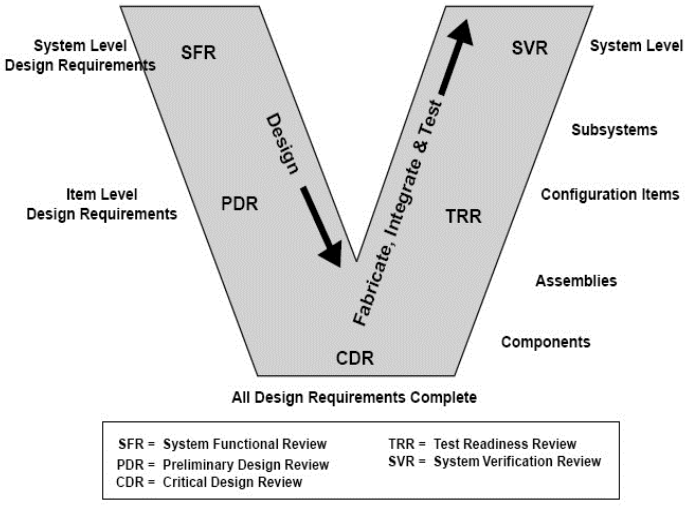



*Figure 56. Systems engineering V-Model (Buede, 2009).*

This traditional approach can also be identified in other lifecycle-based SE models such as the Vee model (Buede, 2009; INCOSE, 2015; Liu, 2015) that Figure 56 shows. This one is an evolution of the simplified waterfall model presented in Figure 58 (Forsberg and Mooz, 2003, 1992; MDD, 2007). In a Vee model, the left side refers to the definition, design, and development phase of a system. This side includes all CIs. The left side includes general requirements, design requirements, and other design development topics. The bottom of the Vee represents all implementation requirements and the beginning of the implementation process. Thus, the left side presents a full definition of the system while the right side is about fabrication, integration, testing, and verification. There are many variations in the literature about this approach (Aughenbaugh and Paredis, 2004; Buede, 2009; Fairley and Forsberg, 2020; Forsberg and Mooz, 1992; INCOSE, 2015), however its basic advancements, structure, and limitations remain. This model is strongly related to software driven projects. It also provides a context for product definition, as well as workflows within the context of project lifecycles, stakeholders, standards, and activities. Nevertheless, some aspects of the lifecycle are not covered such as decommission, services, and other support activities. While this model presents a rigid iterative approach some of its variations allow to increase its adaptability, such as: [1] incremental and iterative development (IID) and [2] the evolutionary approach (see section 3.3). These techniques (Figure 55) present multiple iterative cycles of design and delivery to obtain a faster SOI (Forsberg, 2020;

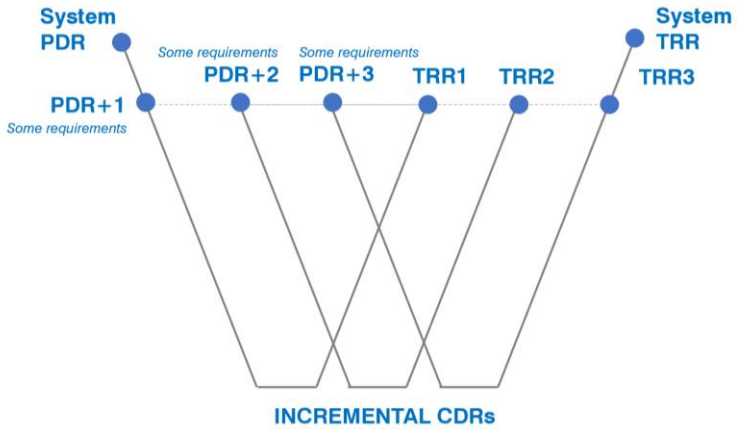



*Figure 55. Incremental and iterative development (IID) derived and based on Forsberg et al. (2005) on (INCOSE, 2015).*

Forsberg et al., 2005; INCOSE, 2015; Larman, 2004) that addresses SE process agility, adaptability, and management.

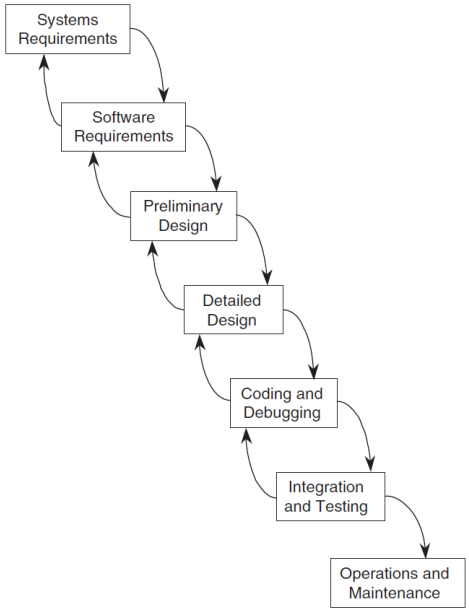



*Figure 58. Waterfall systems engineering model (Buede, 2009).*

In the 1970s the waterfall model was introduced by authors like Boehm (Buede, 2009; Liu, 2015) as an iterative linear process where different phases interact among them and under a continuous flow (Figure 58). As Liu describes, this model presents advantages (Liu, 2015) such as simplicity, frugality, and clarity in terms of structure and implementation.

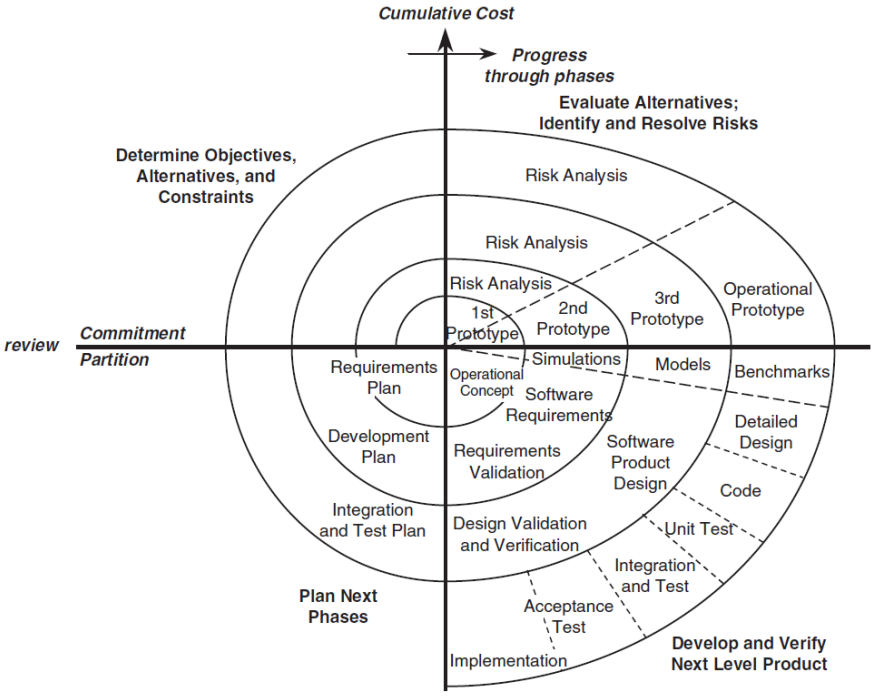



*Figure 57. Spiral systems engineering model (Liu 2015).*

However, connections among phases are not simple in practice and often they happen across non-immediate phases (Haberfellner et al., 2019). Thus, a variation from this approach is the spiral model (Kamrani and Nasr, 2010) that speeds up the SE process (Figure 57) with roots in software engineering. This approach is based on four corners: [1] objectives, [2] evaluation, [3] development, and [4] next phase planning. These covered clockwise a spiral path that defines a new iteration cycle per each pass. This SE model brings flexibility and agility, but it is still based on a linear approach just like the waterfall model. Similarly to the Vee, the spiral model also presents an incremental commitment variation (ISCM) developed by Boehm (INCOSE, 2015). This approach enables a faster risk-driven process that tackles concurrent systems engineering efforts. This method tackles both product and processes (Boehm et al., 2012) allowing a stakeholder value-based approach for any complex system development. The SIMILAR process (Bahill and Madni, 2016) is related to this approach, presenting a networked approach (Figure 60) based on the interconnected phases, such as: [1] problem statement, [2] alternatives, [3] system model, [4] integration, [5] system launch, [6] performance, and finally [7] re-evaluation.

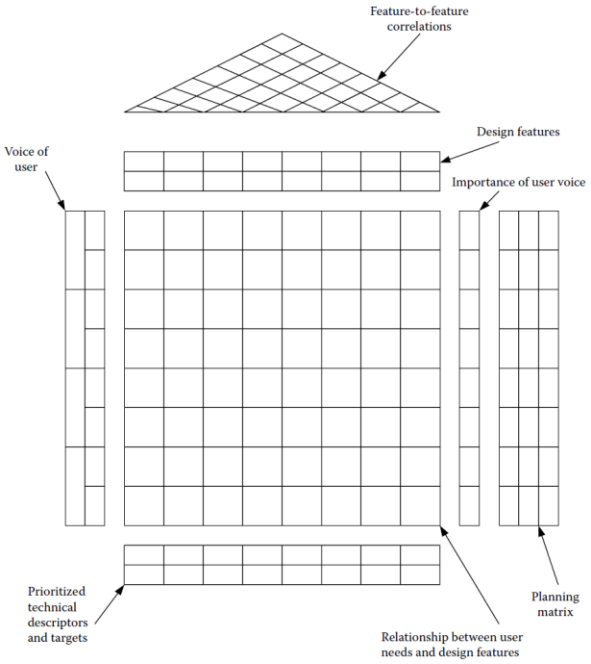



*Figure 59. House of Quality or HOQ (Liu 2005).*

The walking skeleton model (Badiru, 2019) is also an incremental model. The first step within this approach is a very basic but functional system model with only key elements acting as the '*bones*' of the system. Subsequent steps add 'muscles and skin', meaning higher levels of fidelity, new features, and complementary system models. While it started as a software technique, this approach can also be applied to hardware-based systems. A key aspect within this technique is that every phase allows in the next one to work faster by applying lessons learned. This approach presents the following steps: [1] information gathering and methodology workshop based on previous experiences, [2] reflection workshop tackling needs and methods, [3] blitz planning addressing tasks, cost, and assignments, [3] Delphi estimation by experts, [4] daily stand-ups as short efficient meetings, [5] agile interaction design as a fast-paced approach to deliver software products, [6] process miniaturization to reduce cost and learning time, [7] side-by-side programming to provide faster and more reliable results among multiple people working together, and finally [8] burn charts to assess the work that has been done.

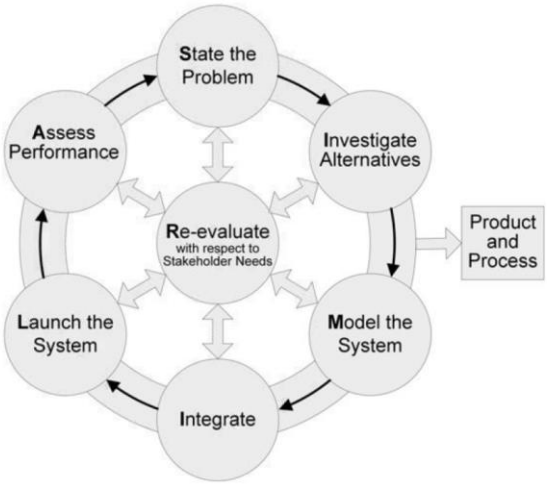



*Figure 60. SIMILAR networked process, after Bahil et al. (2016).*

The prototyping approach (Haberfellner et al., 2019) can also be considered as a very similar technique using four types of prototypes, such as: [1] proof-of-principle, [2] forms study, [3] visual, and [4] functional prototypes. These prototypes help the design process dramatically as Haberfellner et al. presented.

The DEJI SE model (Badiru, 2019) has also four main phases including: [1] design addressing agility, end goal, and stakeholders engagement, [2] evaluation including feasibility, metrics, evidence gathering, and utility assessment, [3] justification involving implementation, desirability and conclusions, and finally [4] integration including affordability, sustainability, and practicality (Figure 61). Under these frameworks, multiple tools and toolsets are incorporated (Badiru, 2019) while the model also includes quality assessment.

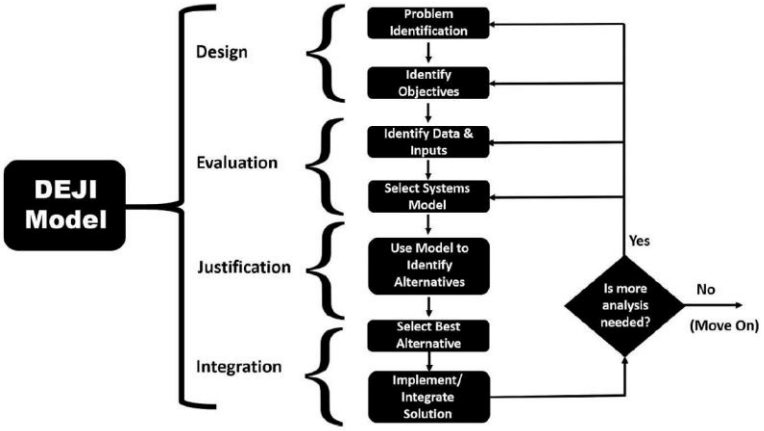



*Figure 61. DEJI systems engineering model (Badiru, 2015).*

Considering all design phases, concurrent engineering (CE) brings a new approach capable of accelerating the design process by running multiple processes in parallel so newer products can get to the market faster (Haberfellner et al., 2019). Once a concept is defined, its development is divided into simultaneous and partial phases determined by the disciplines involved in the process. This approach requires a holistic method to compress the design cycle (Salomone, 2019) which also allows a faster infusion of new technologies. This approach is not only based on models but also collaboration, behavioral dynamics, and process design tools. The goal is to follow a horizonal vee approach, where all efforts converge quickly into a product or a process. In general, CEs enable faster, cheaper, and better valued products for the customer.

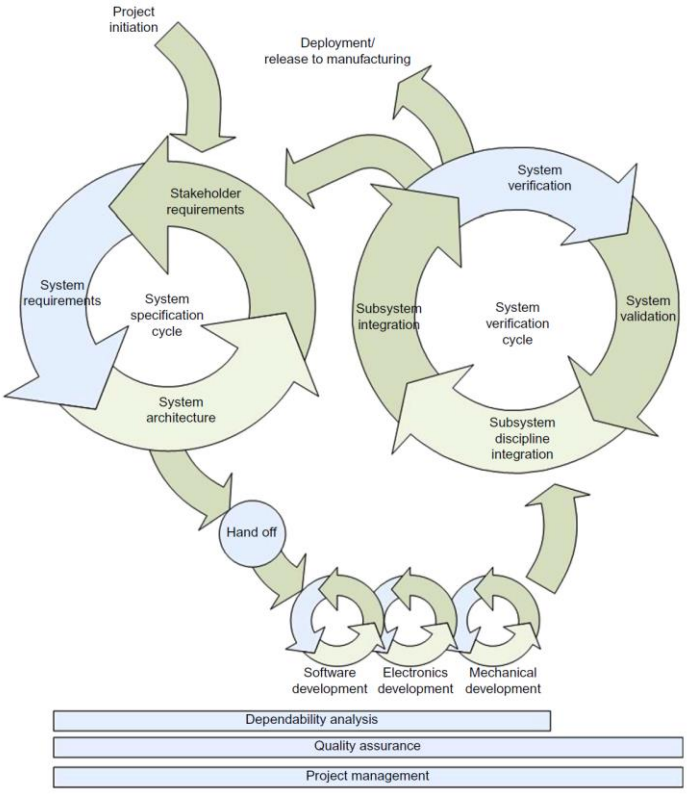



*Figure 62. Hybrid SE lifecycle per Douglas (Douglass, 2016).*

Also agile and lean SE processes are some of the first **cross-cutting** models driven by planning goals. These are used in software development because to have faster, cheaper, and leaner development processes (Haberfellner et al., 2019).

These are variations of the spiral model, including among others the incremental iterative development (IID), SCRUM, adaptive software, and extreme programming (XP) techniques (Douglass, 2016), among others. Based on the agile manifesto a set of key principles organize these SE processes. Among some of the most relevant characteristics are the following: [1] individuals and interactions are more important than tools and processes, [2] the customer collaborates, [3] it is about responding to change not to follow a plan, and [4] self-organization leads to better architectures. This process relates as much to techniques as they do to workforce and team participation (Larman, 2004). Evolutionary process will be tackled in detail in section 3.3, however as a transition from non-agile system it is worth mentioning these hybrid SE lifecycle processes (Douglass, 2016) as Figure 62 shows. Here, multiple design cycles are continuously running, often within each other, to provide faster an outcome with better quality.

Holistic methodologies and models within this category bring approaches such as system thinking (Boardman and Sauser, 2008) that understands the system as the synergy among all its parts. This field includes areas such as game theory, pattern formation, and behavioral science. Such approach presents multiple fields of application, and based on the determination of structures, boundaries, frameworks, and emergence among other key aspects leads to the creation of maps (Boardman and Sauser, 2013) and models known as *systemigrams* (Squires et al., 2010). However, while this approach captures the definition of a complex system, does not necessarily defines the process for its development.

Within the group of cross-cutting SE methods we can find several approaches building upon these models. The object-oriented system engineering process (OOSEM) is based on a series of basic objects or elements that need to be integrated into the system. This approach is characterized by [1] inheritance, so the object gets specialized by inheriting properties of the objects (Buede, 2009), and [2] information hiding, so each object works as 'black-box' that does not know how others objects work. These processes include: [1] architecture development, [2] behavior specification, [3] codesign process and transformation, [4] synthesis, and [5] product development (Morris et al., 2012).

The object-oriented analysis and design (OOAD) is another agile approach that embraces change across the lifecycle of a project (Badiru, 2019). This model groups data, processes, and systems that are turned into objects being managed by one executive person. Each system engineer manages and perfects each object, while a systems manager puts everything together to create the final system solution. This is a people-driven process relaying on the excellence of the workforce. However, managing control within this approach in complicated for larger teams and complex efforts.

Function-based system engineering (FBSE) focuses on system architecture functions including activities, actions, tasks, etc. (INCOSE, 2015). This approach is about what needs to be done instead of the process to enable it. Thus, it creates a functional map of system. Then, those functions are performed by multiple elements (e.g., hardware, software, people, etc.). There are several steps in this iterative networked method including: [1] setting top-level functions and performance requirements, [2] definition of lower-level functions, [3] necessity-based evaluation of lower-level functions, [4] cycle iteration, [5] division of functions into sub-functions, [6] decomposition of requirements, [7] evaluation of alternative decompositions, and [8] identification of all interfaces. This process produces: [1] diagrams such as input-process-output, behavior, control flow, data flow, entity relationship, and functional flow block, [2] models, and [3] simulation results. Tools used by this approach include analysis, modeling, simulation, prototyping, and requirement traceability (INCOSE, 2015).

Similarly, integrated product development (IDP) is a process-oriented approach that considers the full system lifecycle. It creates a continuous integration of the team through requirements, manufacturing, verification, and support (INCOSE, 2015) using integrated product development teams (IPDTs) as Figure 63 shows. In essence, this approach is based on: [1] decentralization of the process, [2] a better connection between the beginning and the end (manufacturing) of the process, [3] interface control, and [4] a concurrent engineering approach. This approach tackles both design and manufacturing processes to implement the system. Within this approach, cross-functional teams are created to tackle all products and services used in a system quickly and independently through systems engineering and integration teams (SEIT), product integration teams (PIT), and product development teams (PDT). The key aspect of

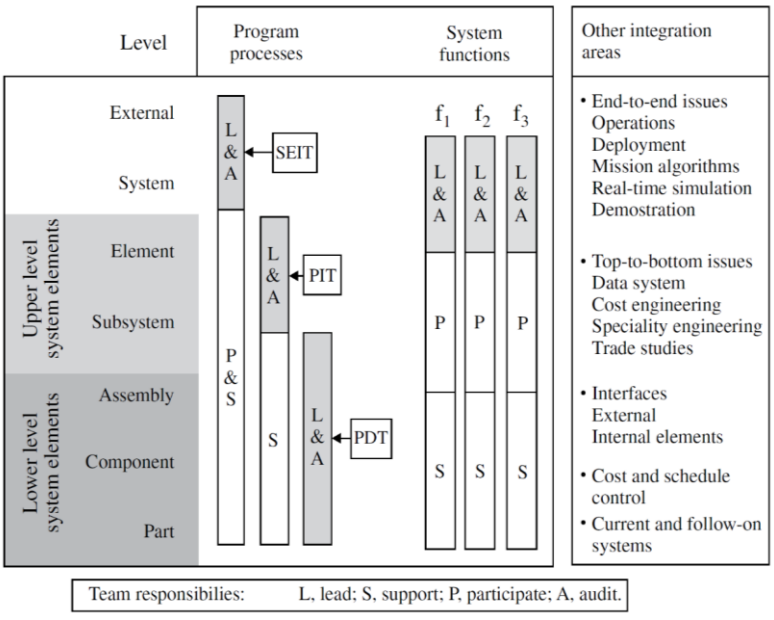



*Figure 63. Integrated product development (IDP), after INCOSE (2015).*

this approach is to use all disciplines and engage all teams from the very beginning, while system-level requirements are passed down to all subsystems and components. Regarding those teams there are several steps involved such as [1] define IPDT, [2] delegate responsibilities, [3] staffing, [4] team operating system, [5] planning and start point, [6] training, [7] team vision and definition, [8] job expansion, [9] routine process and continuous improvement, [10] progress monitoring, [11] team evolution through the project, and finally [12] documentation.

Model-based systems engineering (MBSE) methods use models to create a methodology and a framework (set of models) tackling all lifecycle phases of a system development (Badiru, 2019). These methods include system requirements, design, analysis, verification, and validation (INCOSE, 2015). This approach differs from a document-based approach significantly since it uses active models rather than passive documents. These models consists of requirements, design concepts, test cases, verification plans, trade studies, and relationships between them (Haberfellner et al., 2019). Therefore, they present a multidisciplinary standpoint that is applicable to any agile method. Some authors define this as a flexible "thought" process (Long and Scott, 2011) bringing adaptability, efficiency, agility, and single-source-of-truth into the SE process (Douglass, 2016).

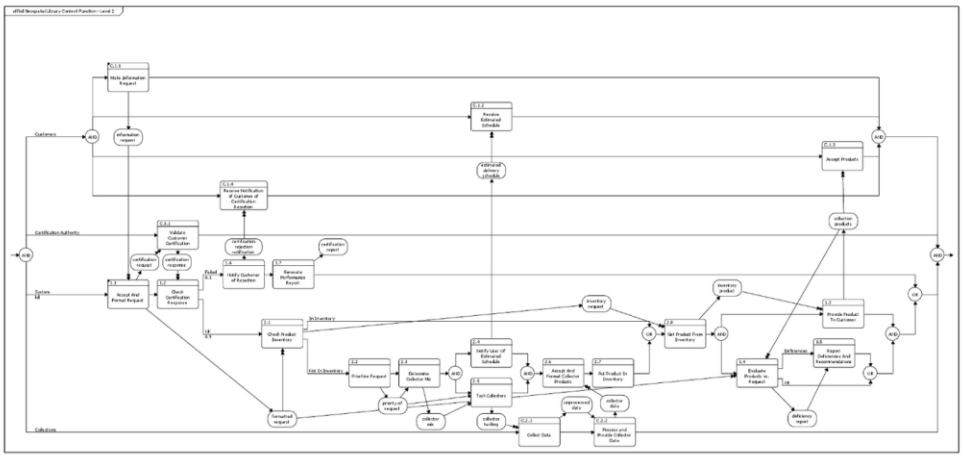


*Figure 64. Example of a MBSE diagram (Long and Scott, 2011).*

The use of models allows to have a common language for different information sources enabling a comprehensible solution that can be verified and an effort that can be reuse. These sources are captured within models as well as any complex relationship among them. There are multiple frameworks and methodologies within MBSE, but they all layer the effort into subsequent cycles of definition which are detailed by the domain. Any change at the domain level ripples throughout the whole model. These models [1] ensure rigor and repeatability, [2] promote quality, [3] reduce risk, and [4] finally enhance communications and synchronization across disciplines, people, efforts, and models (Borky and Bradley, 2018). The MBSE approach is connected to vee models sharing how the approach to tackle the system lifecycle. In general, this methodology includes the following steps: [1] concept development and analysis, [2] requirement capture, analysis, allocation, and traceability, [3] detail design (non-geometrical), [4] integration and test, [5] verification and validation, and finally [6] operations and support. The MBSE approach allows to create a template from any work effort formalizing the SE practice and expanding its reach across all system development activities and its complexity management (Badiru, 2019). The next sections review and study multiple MBSE framework and languages.

Similarly, dynamic and fuzzy systems introduce decision support systems or DSS (Badiru, 2019) capable of dealing with uncertainty and fuzziness. This is based upon probabilistic techniques such as fuzzy-stochastic methods and other related toolsets (Pedrycz and Gomide, 2007). The need for extreme complexity management as well as the emergence of artificial intelligence (AI) and machine learning techniques presents a new context full of potential opportunities for ruled-based SE and other process control methodologies (Nedjah and Mourelle, 2005a). Table 15 presents a detailed summary of tools, languages, and frameworks. The next sections also elaborate the overarching characteristics of these SE techniques from a high-level standpoint since they are used throughout some of the previous theories, process, and methods.

### 3.2.4.3. Tools

Basic tools used in the practice of system engineering can be organized in the following categories:

- **Documents**. The use of physical documents (e.g., notebooks, notes, files, etc.) and lately digital files (e.g., Excel, text edited, etc.) has become the backbone of the SE practice. These documents mainly capture quantifiable parameters (e.g., requirements) using text, formulas, and tables that can be edited. However, these documents are not necessarily interconnected so relationships among them might not be captured. This technique requires a system document strategy to be implemented upfront (Grady, 1995; Wasson, 2005), especially for complex systems. Examples of these are: requirement documents, interface control documents, SEMP, SEMS, SEDS, WBS, TMS, etc.
- **Diagrams**. The application of diagrams as a graphical description capable of capturing parameters and relationships has been critical as well. The use of these not only become the base for system descriptions, but they also enable effective descriptions of workflows and schedules. Among some relevant diagrams we can identify:
  - **Block diagrams** (Figure 66) are used to represent system functions or parts across many disciplines (Karayanakis, 1995).
  - **Flowcharts** illustrate processes and workflows, presenting both elements and relationships. These are at the core of many SE practices, tools, and languages. These are also defined by standards such as ANSI, ISO, MIL, etc. (Nakatsu, 2010)
  - **Signal-flow graphs (*SFG*)** are used to represent variables (nodes) and connectors or branches (equations, functions, etc.) among multiple system components (Levine, 1996).
  - **Functional flow block diagrams (*FFBD*)** are classical SE tools since the 1950s (Figure 53), which illustrate functional operational and system structures, sequences, inputs, outputs, and relationships (Liu, 2015; Mdd, 2008). These diagrams allow to create logical symbols (Booleans) and contextual references.
  - **Data-flows diagram (*BDFD, 70s*)** is a diagram that represents how data flows through a process or a system, capturing both inputs and outputs. These diagrams are developed by systems engineers after questioning stakeholders, users, and other systems description efforts (Shelly and Rosenblatt, 2009).
  - **Reliability block diagram (*RBB*)** is a diagram method created to assess component reliability, dependencies, and redundancy (Birolini, 2007) within complex systems across multiple fields.
  - **Process flow diagram (PFD)** shown in Figure 67 (Ohare, 2015), is used within SE processes and other physics and chemistry-based engineering system descriptions (Turton et al., 2008).
  - **N2 chart** (Figure 69) is a matrix-shaped diagram addressing functions and interfaces among systems (Batson, 1986).
  - **GANTT chart** (Figure 68) represents a project schedule with milestones, tasks, and dependencies against time (2010s). This is a classic tool used today in project management (DuBrin, 2011; Malyszkz, 2011) across multiple industries and industrial fields.
  - **Event chain diagram** is a complementary Gantt chart allowing to visualize relationships among events. These are also relevant tools in risk assessment and system analysis (Hulett, 2016).
  - **Control flow diagram (CFD)** is used to describe control flow and signals involved in multiple processes as part of a classic SE approach. They are applied to change, configuration, process, performance, and quality flows within a system, among others areas (Hatley et al., 2013).

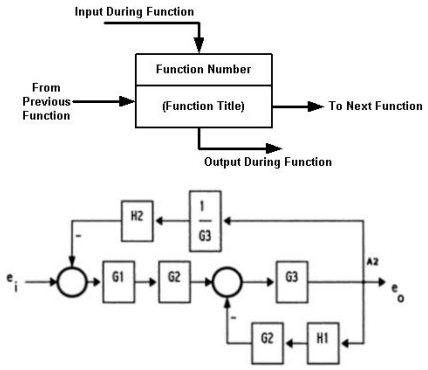



*Figure 66. Block diagram, after Karayanakis (1995).*

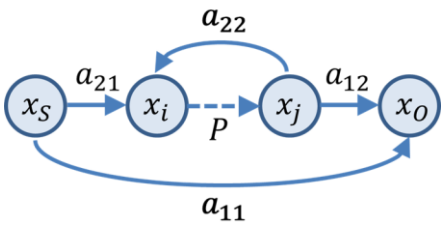



*Figure 67. Process flow diagram, after Ohare (2015).*

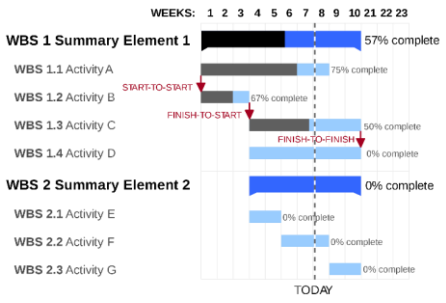



*Figure 68. Gantt chart, after Malyszkz (2011).*

- **Program evaluation and review technique (PERT)** shown in Figure 70 is a statistical diagram developed during the 2010s. It is used to evaluate and design multiple tasks within a project under the measurement of time. These diagrams express relationships, time, and milestones. They are also often used in conjunction with critical path methods (CPM) techniques (Nicholas, 2004).
- **Use case diagrams** (Figure 71) are critical for some SE languages. They represent an activity performed by a system in response or by request of a user (Satzinger et al., 2008). These diagrams also imply the use of definitions such as scenario, exceptions, actors, trigger events, stakeholders, as well as preconditions and post conditions, among others.
- **Sequence diagrams** (Figure 72) show similarly interaction among systems under a timeline (Windle and Abreo, 2003). Vertical lifelines present sequences, horizontal lines describe coexistent elements, and arrows represent communication between elements. These last two diagrams are the base of some SE languages such as UML, which will be explained more in details in the next section.

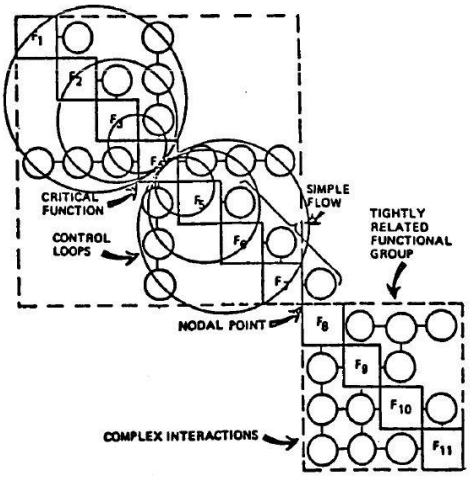



*Figure 69. N2 Diagram, after Batson (1986).*

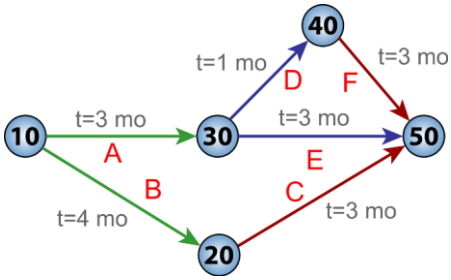



*Figure 70. PERT diagram, after Kemp (2015).*

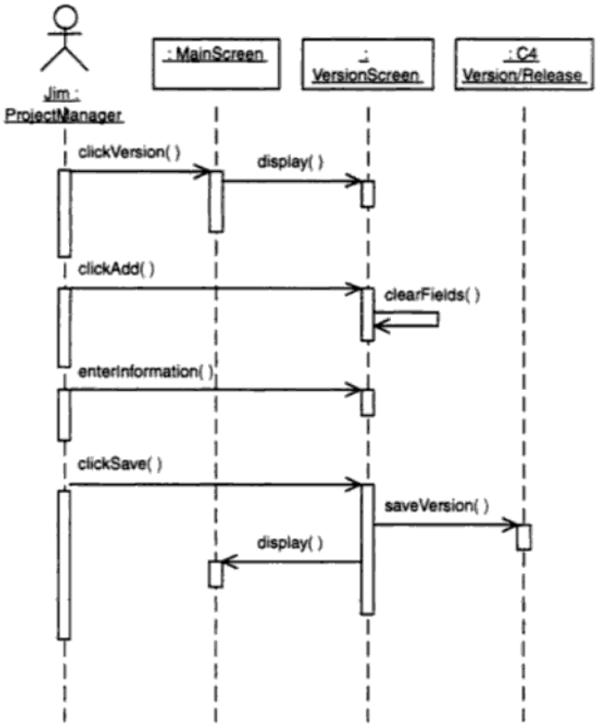



*Figure 72. Sequence diagram, after Windle (2003).*

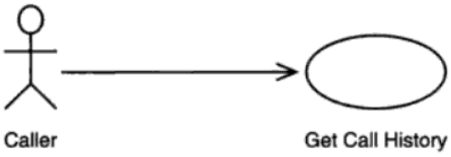



*Figure 71. Use case diagram, after Satzinger (2008).*

- **Matrixes**. These have been another classic tool in the historical toolset of SE. The use of tables has multiple applications. They are capable to tackle both qualifiable and quantifiable parameters. These can be physical (paper) or digital. Software tools like spreadsheet packages are widely used in SE and are purely based on matrix spreadsheet operations. Beyond their generic use there are a few relevant types in the practice of SE, such as:
  - **Design structure matrix (DSM)** also known as dependency structure matrix was developed in the 1960s. It is a table-based representation of system components and dependencies (Madani et al., 2014).
  - **Pugh matrix, coupling matrix, or decision-matrix method (PM)** allows to compare multiple candidates based on specific criteria to select the most optimum solution (Burge, 2009; INCOSE, 2015). This method also allows to weight multiple options and find the best alternative as the example in Figure 74 (Muller et al., 2011) shows.
  - **House of Quality (HOQ)** is a part of the quality function development method (QFD). It was created in Japan to transform qualitative needs into quantitative parameters with multiple application across industrial and business sectors (Madu, 2006). Relationships between rows and columns are codded with symbols (strong, moderate, weak, and very weak). It works as a SE communication device (Liu, 2015) allowing also to turn user desires into requirements (Figure 76, Cask05, 2006).
  - **Requirement verification and traceability matrix (RVTM)** is used to trace requirements across the lifecycle of a system as well as during testing (Phillips, 2004; Wasson, 2005). The use of codes allows to trace back the Vee model. Other aspects of the process can also be tracked such as accountability, analyses, inspections, etc. (Figure 75).
  - **Risk assessment matrix** is a standard tool to assess risk in multiple SE environments such as NASA, ISO, DoD (Figure 77). While this approach can present challenges in terms of resolution and allocations, is widely used across industries (Popov et al., 2016).

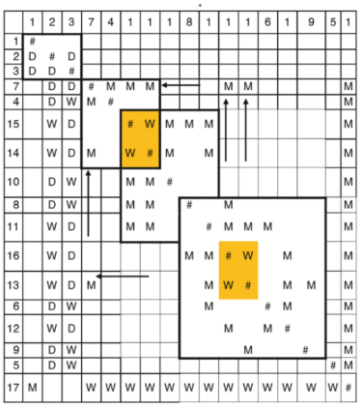


*Figure 73. DSM Example, after Madani et al. (2014).*

| Evaluation Criteria | | Score | Concepts 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| Time to connect | | | | | | |
| | Need for ROV | | - | + | + | + |
| | Design | | - | + | + | + |
| Robustness | | | | | | |
| | Connector Design | | - | S | S | + |
| | Number of parts | | - | - | + | + |
| | Handle roll-off | | + | - | S | + |
| | Influence other | | + | S | - | S |
| Redundancy | | | | | | |
| | Design | | + | - | - | S |
| | Interchangeability | | + | - | - | - |
| Cost | | | | | | |
| | HW Cost | | - | - | - | - |
| | Manufacturing Cost | | S | S | - | S |
| | Engineering Cost | | + | - | S | - |
| | Service Cost | | - | + | + | + |
| Maturity | | | - | - | S | + |
| | | Σ - | 7 | 7 | 5 | 3 |
| | | Σ S | 1 | 3 | 4 | 3 |
| | | Σ + | 5 | 3 | 4 | 7 |
| | | Position | 3 | 4 | 2 | 1 |

*Figure 74. Pugh matrix example, after Miller et al. (2011).*

Besides these classic tools there are other general toolset techniques:

- **Graphs** showing mathematical and statistical information.
- **Codes** used to run scripts and other data management tools.
- **Maps** are used for information illustration and decision-making processes such as mind maps and systemigrams.
- **Analyses** of different nature are also a key SE tool to study regression, reliability, feasibility, FEM, risk, etc.
- **Simulations** also used for testing (quantification) and demonstrations (qualification) purposes (INCOSE, 2015).

| Reqmt. ID | Requirements Statement | Allocated to | Verification Level | Method of Verification | | | | Traces Vertically to |
|---|---|---|---|---|---|---|---|---|
| | | | | Inspect | Analysis | Demo | Test | |
| SYS_136 | **3.1.1 Capability A** The system shall … | Subsystem 123 | Subsystem | X | | | | 3.1 Capability XXXX |
| SYS_137 | **3.1.1 Capability A1** The system shall (Capability A1). | Assembly A1 | Assembly | X | X | | | 3.1.1 Capability A |
| SYS_138 | **3.1.1.2 Capability A2** The system shall (Capability A2). | Assembly A2 | Subsystem | | X | | | 3.1.1 Capability A |
| SYS_139 | **3.1.1.3 Capability A3** The system shall (Capability A3). | Assembly A3 | Assembly | | | X | | 3.1.1 Capability A |
| SYS_140 | **3.1.1.4 Capability A4** The system shall (Capability A4). | | Assembly | | | | X | 3.1.1 Capability A |
| RVTM columns Applicability | RTM & RVM | RVTM | RVM | RVM | RVM | RVM | RVM | RTM |

*Figure 75. Requirement verification and traceability matrix (RVTM), after Wasson (2005).*

**CONSEQUENCES**

| LIKELIHOOD | Unlikely | Negligible | Marginal | Critical | Catastrophic |
|---|---|---|---|---|---|
| Certain | High | High | Extreme | Extreme | Extreme |
| Likely | Moderate | High | High | Extreme | Extreme |
| Possible | Low | Moderate | High | High | Extreme |
| Unlikely | Low | Low | Moderate | Moderate | High |
| Rare | Low | Low | Low | Low | Moderate |

*Figure 77. Risk assessment matrix*

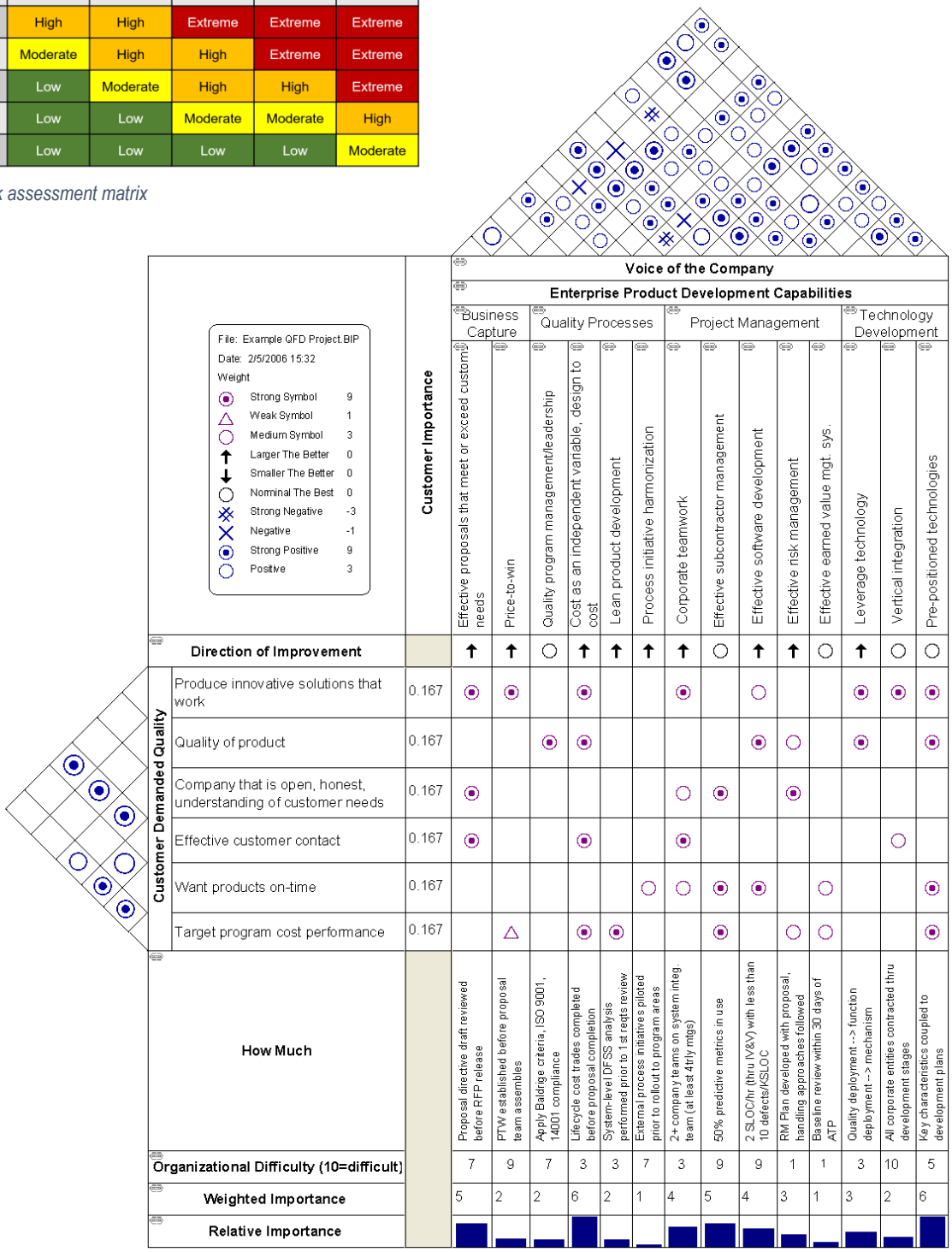



*Figure 76. House of quality, after Cask (2006).*

#### *3.2.4.4. Systems Engineering Languages*

Complementing the use of previous tools there are a series of SE modeling languages, which can be understood as a series of graphical, mathematical, and procedural structured codes. These languages enable to analyze, design, and implement systems through multiple SE workflows. Their origins are found in multiple fields going from software to mathematics. Today, there is a growing infusion of these practices which is changing the SE practice and creating new possibilities to explore components, parameters, and interrelationships, as well as to broaden them with data-driven perspectives. Several categories can be identified across the vast number of techniques and languages including:

- **Modeling languages**. Objected and function-oriented SE practices, as well as object modeling techniques (OMT) present the foundation for the use of modeling languages as an evolution of classical FFBDs. These present some common capabilities (Borky and Bradley, 2018) such as: [1] abstraction to work with common characteristics among elements, [2] encapsulation to compartmentalize systems, subsystems, and architectures, [3] modularity to reuse elements, [4] generalization and inheritance to create instantiations with a hierarchical structure, [5] aggregation and composition to build new elements upon previous solutions, [6] interfaces, and [7] polymorphism. Thus, an element can perform differently depending on its uses and needs. These languages provide the means to have a single source of truth (Douglass, 2016) and an easier integration of multiple data sources, easier maintenance, management, and verification. Among the most relevant languages supporting MBSE and SE practice we could find:
  - **Structured analysis and design technique (SADT)** evolved from the FFBS approach (Ramos et al., 2012) as a graphical language to describe hierarchically systems and functions.
  - **Integration definition for functional modeling (IDEF)** is a family of systems and software modeling languages, including IDEF0 (functional), IDEF1(information), IDEF1X (data), IDEF3 (processes), IDEF4 (object-oriented design), and IDEF5 (ontologies) (Haberfellner et al., 2019; Mayer, 2009). Here, the function node represents inputs, outputs, control, and mechanism calls (Williams et al., 2010) as a support tool for function-driven SE practices and workflows (Buede, 2009).
  - **Universal systems language (USL)** was developed after the NASA Apollo program (Hamilton technologies). It is based on axioms and includes a heavy inherent error-testing approach and an ontology (Hamilton and Hackler, 2009). It is based upon principles to look at the system from an asynchronous, distributed, and event-driven perspective. This language uses among others functions maps (FMaps), type maps (TMaps), and control applications as part of the workflow.
  - **Unified modeling language (UML)** is a multi-purpose modeling language oriented towards software engineering. It uses graphic diagrams and specified data objects, program entities, attributes, and relationships (Haberfellner et al., 2019). UML presents three types of diagrams (Buede, 2009): [1] structural that includes class, components, composite structure, deployment, objects, package, and profile, [2] behavioral activity, state machine, and use case, and finally [3] interaction including collaboration-communication, interaction overview, sequence, and timing.
  - **Lifecycle modeling language (LML)** is an open-standard and user-friendly SE modeling language (Hettema, 2013). It covers the full lifecycle including concept, use, support, and

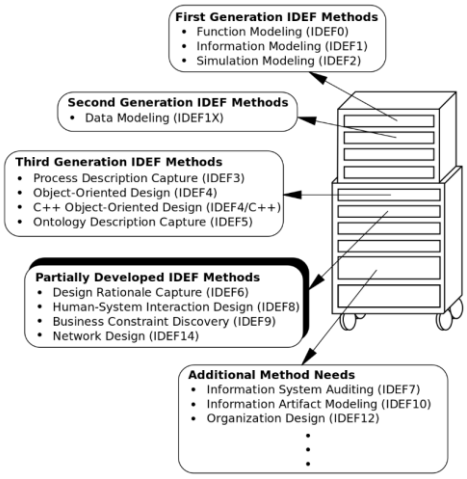



*Figure 78. IDEF Methods, after Mayer (2009).*

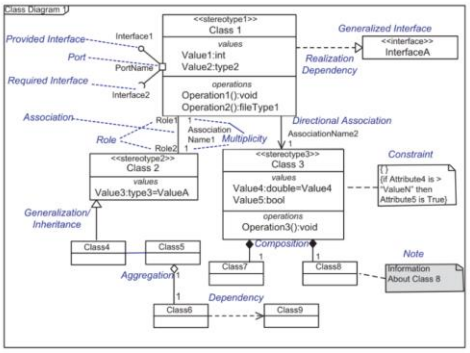



*Figure 79. UML class diagram, after Borky and Bradley (2018).*

retirement phases. It is based on classes (entities), relations (relationships), and properties (attributes), which are called ERA. It also includes risk, cost, schedule, and performance (form, function, metric, interface) as basic ontology aspects.

- **Systems modeling language (SysML)** is a standardized graphical systems engineering modeling language based upon UML (Haberfellner et al., 2019), see Figure 80. The system model presents the following families of elements / diagrams (Friedenthal et al., 2008; GFAB, 2010): [1] structure (block definition, internal block, package), [2] requirements, [3] behavior (activity, sequence, state machine, use case), [4] parametrics. This subset of the UML language is more oriented towards systems, and includes more flexible semantics, better allocation tables, and management principles and tools. The model becomes the single-source-of-truth (Estefan, 2008). There are multiple releases such as the OMG SysML™.
- **Drakon** is a visual programing modeling language developed in Russia in the 1990s. DRAKON 27 flowcharts are understood as letters which can be used to graphically create words and sentences using a syntax. Then, this graphics syntax can be customized using textual syntaxes from other languages such as C+, ASM, Java, etc. see Figure 81 (Ivannikov, 1995).

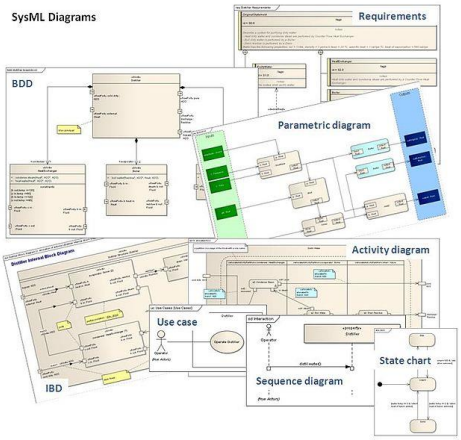



*Figure 80. SysML diagrams, after GFAB (2010).*

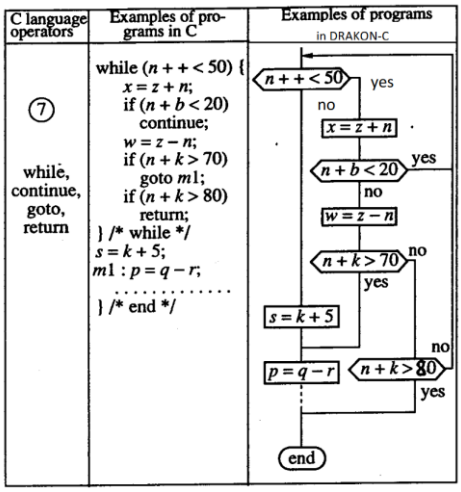



*Figure 81 DRAKON-C example, after Ivannikov (1995).*

- **Mathematical** and coding languages provide nowadays systems engineering, programming, and MBSE support enabling the capability to run scripts, analysis, and study tools that complement more traditional workflows. We can identify several of them:
  - **Algebraic modeling languages (AML)** is a family of high-level mathematical languages allowing to solve complex large-scale mathematical problems (Kallrath, 2004). This include AMPL, GAMS, AIMMS, etc. The AML approach allows to manage multiple solvers (algorithms) tackling different aspect of the problem-solving workflow.
  - **MATLAB**™ is a multi-paradigm and object-oriented functional language developed in the 1970s. It is widely distributed among multiple engineering practices worldwide (Dukkipati, 2008).
  - **Wolfram Mathematica**™ is a language supporting for machine learning, image processing, geometry, data science, statistical analysis, and neural networks, among other traditional engineering fields (Magrab, 2014).
  - **Other languages** include Maple, GNU Octave, Scilab, FreeMAT, Julia, etc.
- **Coding languages.** Generic coding languages are also widely distributed today outside purely computer programing workflows. These include object-oriented, imperative, declarative, concurrent, visual, multimedia-based, web-based, event-based, and integrative languages (Bansal, 2013). Good examples among some of these languages supporting SE practices are the following:
  - **Visual Basic** for general applications (Badiru, 2013) (Alves et al., 2009).
  - **C++** is used for trade studies, telecommunications SE, etc. (Thompson, 2020).
  - **Python** infusion and use is growing (e.g., operations, trade studies, and engineering optimization) (Allbee, 2018).
  - **SQL** is used among other fields in data management (Baba et al., 2001).
  - **PHP** is applied to enterprise SE (SeE) among other areas (Gorod et al., 2014).
  - **JavaScript** has information SE and blockchain applications among others (Matulevičius and Dijkman, 2018).

### 3.2.4.5. Systems Engineering Frameworks and Methodologies

The combination of dedicated tools and languages within a technical context constitutes a SE framework within this literature review. These frameworks not only present a series of integrated tools and languages within a software suite or tool collection, but they also imply a specific set of methodologies and sometimes even entire fields of application. The following groups of SE tool frameworks are identified as key categories:

- **Software** systems engineering development frameworks:
  - **Computer-aided software engineering (CASE)** is a set of tools developed by IBM from the 1960s to the 1990s to support the multiple phases of system development lifecycle (SDLC). There are (Valacich et al., 2017) the following: [1] project identification and selection (diagrams and matrix tools), [2] project start and planning (repository), [3] analysis (diagrams), [4] logical and physical design (document generators), [5] implementation (code generators), and [6] maintenance. This framework includes and integrates workbenches, environments (language centered processes), and tools. (Berdonosov and Redkolis, 2010; Shelly and Rosenblatt, 2009).

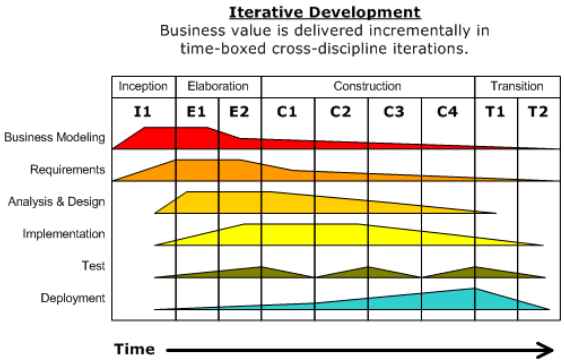



*Figure 82. RUP Iterative development, after Dutchguilder (2007).*

  - **Rational unified process (RUP)** is an iterative object-oriented software development framework, which was also created by IBM Rational software (Valacich et al., 2017). It presets four stages: [1] inception, [2] elaboration, [3] construction, and [4] construction (Figure 82). Some areas of best practices include iterative development, requirement management, components use, visual modeling, quality verification, and change control (Kruchten, 2004, Dutchguilder, 2007), among many others.
- **Systems engineering architecture frameworks** present a reference environment to standardize system definitions, as well as to provide support for enterprise architectures. These frameworks are used worldwide and they include among others (Friedenthal et al., 2008; INCOSE, 2015) the following:
  - **The open group architecture framework or TOGAF** (INCOSE, 2015) includes supporting tools for information technology enterprise architectures, becoming the industry standard its main goal (Dickerson and Mavris, 2016).
  - **Federal enterprise architecture framework (FEAF)** is a federal enterprise reference architecture to integrate businesses and technologies (Kappelman, 2009). Figure 83 shows the full suite of tools under FEAF with the consolidated reference model (CRM) at its core (CIO Council, 2013).

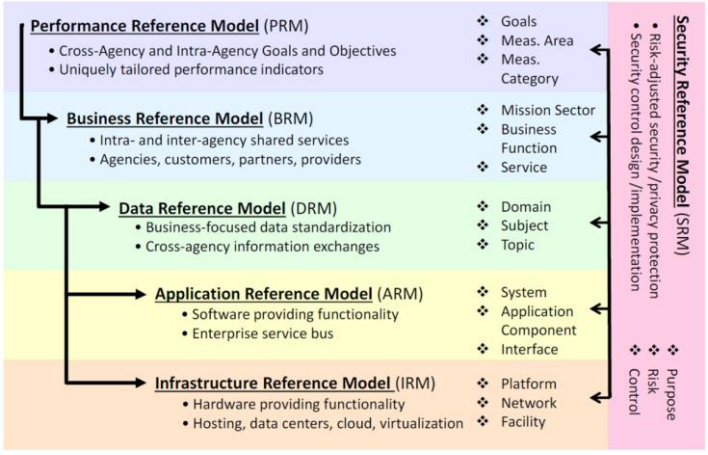



*Figure 83. FEAF Consolidated reference model, after CIO (2003).*

- **Department of defense architecture network (DoDAF)** is a foundational visualization framework for large SE developments (Liu, 2015). Furthermore, it also provides a model-driven analysis and simulation framework for systems engineering practices across many technical fields (Mittal and Martín, 2018).

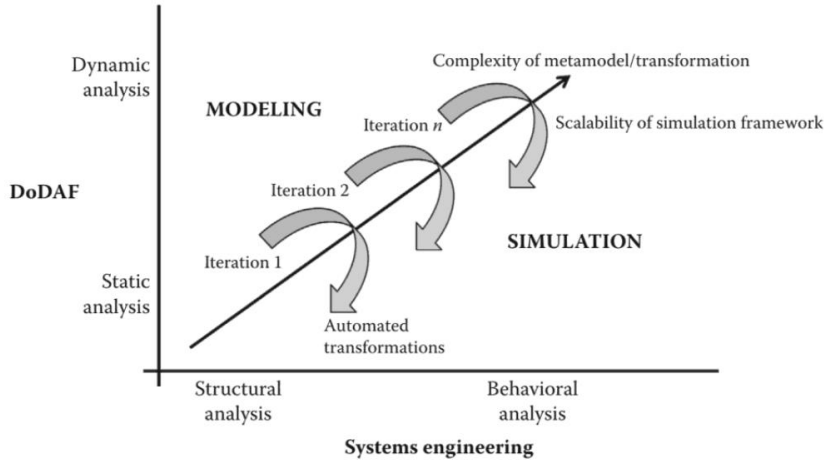



*Figure 84. Modeling, simulation, and systems engineering within DoDAF, after Mittal (2018).*

- **MoDAF** is the British ministry of defense architecture framework (Dickerson and Mavris, 2016) that includes seven key areas: technical standards, strategic, operations, service, system, and acquisition (Babers, 2015).
- **Zachman Framework** is an enterprise ontology based on identification, definition, representation, specification, configuration, and instantiation. It presents a set of rules (Zachman, 1987) that are simplified in Figure 85 (Zuech, 2002) with the key objective of providing an organizational scheme for artifacts and systems.

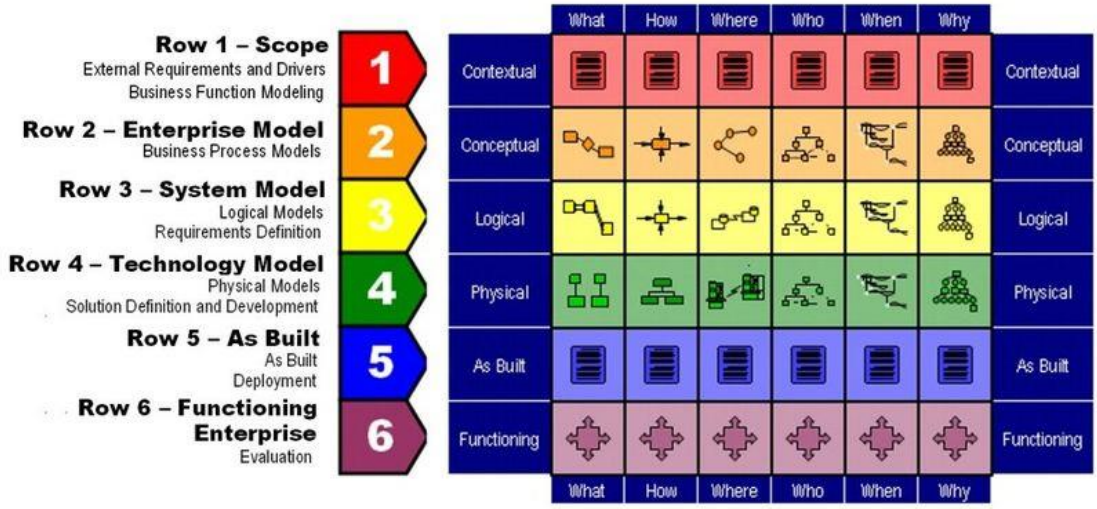



*Figure 85. Simplification Zachman Enterprise Framework, after Zuech (2002).*

- **MBSE**. There are several SE frameworks within MBSE state-of-the-art practice, including these ones:
  - **Harmony SE** (IBM Telelogic). It is a model-driven development environment. It uses Vee models, OMG SysML™, and a basic flow including: [1] requirements analysis, [2] system function analysis (identification, states, modes, physical architecture), and [3] architecture design and synthesis (Estefan, 2008; Ramos et al., 2012). Telelogic Tau and Telelogic Rhapsody are the main support tools for this framework.

- **OOSEM** (INCOSE) presents an object and scenario-driven Vee environment, using SysML™ and OMG tools (Ramos et al., 2012). Its development process (Figure 86) includes: [1] stakeholder need analysis, [2] system requirement analysis, [3] logical architecture definition, and [4] physical architecture synthesis of candidates (INCOSE, 2015).

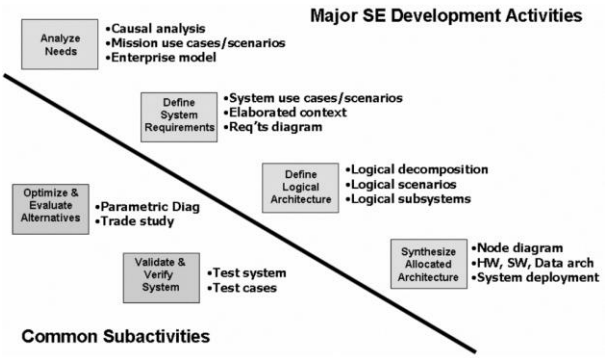



*Figure 86. OOSEM Activities and artifacts, after Stefan (2008).*

- **RUP SE** (IBM Rational) is a model-driven ICSM systems development framework for SE (INCOSE, 2015). This approach is based on object-oriented spiral models and it uses both UML and SysML™ (Ramos et al., 2012), while it emphasizes the business model side (Brusa et al., 2017). Its lifecycle approach presents four stages: [1] inception, [2] elaboration, [3] construction, and [4] transition. However, new roles, workflows, and artifacts are introduced within this MBSE approach when compared to the purely software-driven RUP approach (Estefan, 2008).
- **Architecture analysis and design integrated approach (ARCADIA)** is an MBSE development framework for software and hardware architectures developed by Thales. It is based on three activities: [1] need analysis and modeling, [2] architecture and validation, and [3] requirements engineering (Brusa et al., 2017).
- **Alstom advanced system architecture program (ASAP)** is a top-down SE application with multiple views: [1] operational, [2] functional, and [3] constructional. This approach includes an evolution of the system (Fanmuy et al., 2016) since the object of information can be duplicated and manage more easily.
- **Vtech MBSE** (Vtech corporation) uses Vtech CORE™ environment as a SE design repository across stakeholders and domains. These include: [1] requirements analysis, [2] behavior/functional analysis, [3] architecture synthesis, and [4] verification and validation (Brusa et al., 2017). Figure 87 shows more details. It also uses the system definition language (SDL), which is based on elements, relationships, structure, entities, attribute of relationships, and attributes using a patented 'onion' SE model. Multiple layers in the framework this approach enable SE activities to increase concurrently and incrementally all levels of definition (Estefan, 2008).

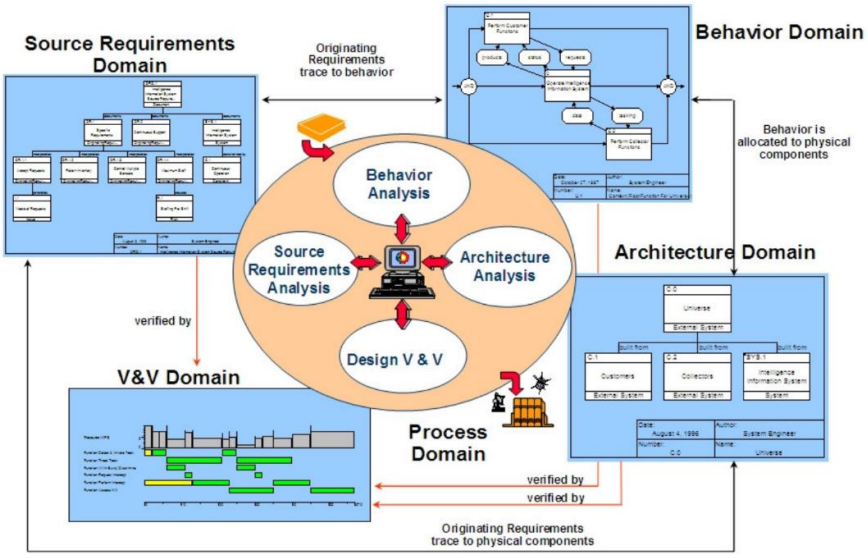



*Figure 87. Vitech MBSE domains and activities, after Stefan (2008).*

- **Object process methodology (OPM)** was developed by Professor Dov Dori as a holistic modeling language and methodology based on an ontology of objects (things that exists) and related processes transforming patterns of those objects. This method is used for natural and artificial systems based on: [1] function (what they do), [2] structure (how they are constructed), and [3] behavior (how they change over time) (Dori, 2016). This approach defines the system development, lifecycle, and evolution including maintenance and usage based upon: [1] requirement specification, [2] analysis and development, and [3] implementation (Brusa et al., 2017). The OPM uses simple graphics or OPD (object-process diagrams) as well as natural languages sentences or OPL (object-process language) (Dori, 2016). OPM uses OPCAT software tools (Ramos et al., 2012). See Figure 88 for a simple example of both OPD and OPL (Estefan, 2008).

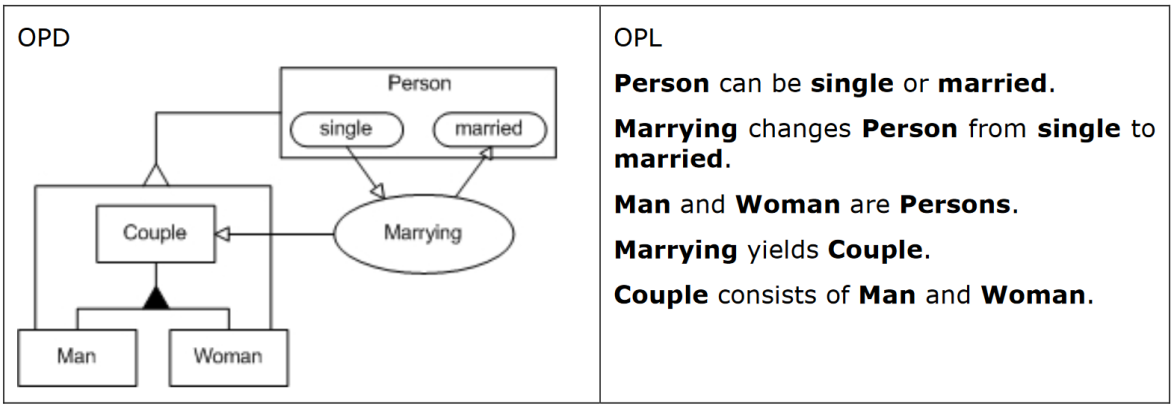



*Figure 88. Example of simple OPD and OPL modeling examples in OPM, after Stefan (2008).*

- **Architecture and engineering tools.** Within these disciplines, certain families of tools present specific workflows, which while they are not fully SE in nature. These can be used in the practice of SE as well as in the development of complex system development lifecycles. Among the most widely used toolsets are the following ones:
  - **Building information modeling (BIM)** provides a framework to plan, create, manage, and modify digital representations of buildings tackling functions, properties, phases, and designs across the lifecycle of construction. This tool integrates requirements analysis, design, construction, operations, cost, sustainability, and recycling (Smith and Tardif, 2009). This is especially relevant in the creation of system definitions and documentation delivery that including manuals, model views, etc. (Eastman et al., 2011). BIM creates a digital representation of the building in a 3D environment, including multiple discipline perspectives into the model (e.g., architecture design, structures, HVAC, plumbing, etc.). There are expansions of this workflow into SE and MBSE realms (Polit Casillas and Howe, 2013) such as: [1] 4D BIM (connecting 3D components with scheduling), [2] 5D BIM (adding cost information), and [3] 6D BIM (adding the dimension of operations, and maintenance).
  - **Product lifecycle management (PLM)** aims towards connecting all information with regards of products and enterprises questions (Elangovan, 2020) such as documents, workforce, and relationships with enterprise resource planning (ERP). PLM serves multiple phases and steps across the lifecycle of a system such as: [1] systems engineering (requirements, variations, reliability), [2] product portfolio, [3] product design, [4] manufacturing, and [5] product data management (Tyulin and Chursin, 2020). Figure 89 shows the PLM lifecycle.
  - **Rational dynamic object-oriented requirements system (DOORS™)** is a requirement system developed by IBM Telelogic. It was created aiming requirement optimization, communications, verification and collaboration (Stjepandić et al., 2015). This allows the use and generation of UML models linking requirements and enabling a collaborative platform between all stakeholders such as system engineers, vendors, analysts, etc.
  - **Microsoft Office Suite** is also widely distributed across the SE and general engineering practice. Spreadsheets, schedules, databases, etc. are used by individuals and organizations across the world, both discreetly and using interconnected models to address requirements, reliability, trade-space studies, cost, etc. (Badiru, 2013).

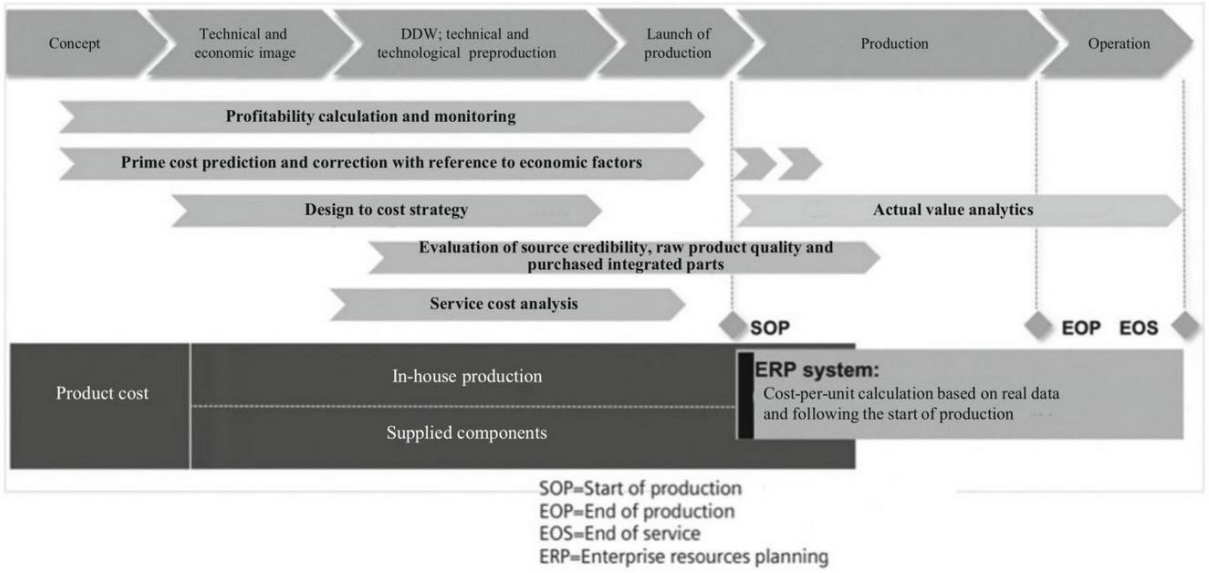



*Figure 89. PLM use across the system and product lifecycle, after Tyulin and Chursin (2020).*

#### *3.2.4.6. Systems Engineering Literature Review Matrix*

Table 15 presents a similar summary to the one in section 3.1.3.3 regarding state-of-the-art practice systems engineering approaches that were elaborated in previous points. Key characteristics addressed by this study include:

- **Foundation**. This is a short summary description or keywords of basic principles and characteristics.
- **Main function or task.** Does the systems engineering approach concentrate on analysis (ANSY), design (DES), implementation (IMP), or all of them at once?
- **System design phase.** What lifecycle phases are addressed by this approach? Basic design phases are numbered as it follow: [1] planning, [2] problem study, [3] concept design, [4] embodiment design, [5] detailed design, [6] analysis, [7] optimization, [8] testing and validation, [9] documentation, [10] implementation, [11] delivery, [12] marketing, [13] operations, [14] decommission, and [15] recycling of products and processes (Seider et al., 2016) (Haik et al., 2010). See Figure 45 for color codes, detail level, and structure.
- **Geometrical or abstract information**. This refers to the capability of the SE method to [1] manage, author, and edit geometrical information (GEO) such as volumes, shapes, sections, tolerances, and other graphical constructs, [2] handle abstract information (ABS) such as analytical parameters, and finally [3] address system interfaces (INT).
- **Qualitative / quantitative (Qt./Ql.)**. Can the method be used to quantify and qualify multiple parameters?
- **Scope**. Can the SE method handle only point-design solutions (PDS), families of point-design solutions (FDPS), development process (DEV), continuous designs (CONT), or a combination (COMB) of all of them?
- **Adaptability**. This addresses if the SE method is flexible (FLE), networked (NET), strict linear (LI), iterative (ITE), waterfall (WA), or used spiral (SPI) methodologies. Figure 46 shows graphically these types of methods.
- **Perspective.** Is the SE method based upon discrete disciplinary standpoints (DD) or synergetic multidisciplinary approaches (SA)? In other words, is the method based on a 'divide-and-conquer' approach discretizing disciplines and subsystems? Or on the hand, can it tackle multidisciplinary perspectives?
- **Optimization**. Does the approach allow a parametric optimization of the system or just its parameters?
- **Tool platform**. What type of tool or technique does the SE approach enable or support? This can include: [1] mathematical models, [2] drawings, [3] CAD/PLM, [4] graphs, [5] Eng. models, [5] documents/text, and [6] schedule.
- **Reference**. This is a summary list of relevant technical references and professional practice inputs reviewed during this research and thesis dissertation.

| Theory/Method/Tool | Foundation | Function | Phase | Geo. | Qt./Ql. | Scope | Adapt. | Pers. | Opt. | Tools | References |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***SE1 - System Engineering Theories and Standards*** | | | | | | | | | | | |
| **SE1-1 General** | Historical SE techniques and theories. | | 1-15 | | | | | | | | |
| **NAVY PERT** | (1958) Evaluation program for evaluation and review technique for manufacturing scheduling method based on: [1] activity, [2] costs, and [3] time. | ANSY IMP | 1,6,7,10,11 | ABS | QT | PDS FPDS DEV | ITE | DD | Yes | Math Graphs | (Liu, 2015) |
| **Hall's** | (1962) This is a SE method with 5 phases: [1] systems and program planning, [2] exploration planning, [3] development planning, [4] development, & [5] current Eng. | ANSY DES | 1-6, 10, 13 | ABS | QT | PDS | N/A | DD | No | Math | (Liu, 2015) (Buede, 2009) |
| **NASA SE** | (1960-20s) It is based on: [a] mission, [b] operations objectives, [c] mission success, [d] requirements, [e] constraints (functional, logical, behavioral & design), [f] trade studies, [g] design studies, and [h] product breakdown structure. | ANSY DES IMP | 1-15 | ABS | QT | PDS DEV COMB | ITE NET | DD | Yes | Math Graphs Models | (Johnson, 2006) (NASA, 2007) (Hitchins, 2008) |
| **Contemporary SE** | Complex system engineering based on signal, data, materials, and energy. Three phases: [a] concept development, [b] engineering development, and [c] post-development (DoD, ISO, NASA, IEC) | ANSY DES IMP | 1-15 | ABS | QT | PDS DEV COMB | ITE NET | DD | Yes | Math Graphs Models | (Kossiakoff et al., 2020) (Lapham et al., 2014) |
| **Systems Thinking** | SE based on deep analysis: [a] system dynamics, [b] action research & soft systems, and [c] pattern discovery. | ANSY DES IMP | 1-10,13 | ABS | QT | PDS DEV COMB | ITE NET | DD | Yes | Math Graphs Models | (Senge, 2010) (INCOSE, 2015) |
| **SoSE** | It is a system of systems engineering methods for extreme complex systems. Keys include synergism, self-government, reconfiguration, symbiosis, and modularity. Potential biological guiding principles. | ANSY DES IMP | 1-15 | ABS | QT | FPDS DEV | NET FLEX | DD | Yes | Math Graphs Models | (Badiru, 2019) (Jamshidi, 2011) (Sauser et al., 2010) |
| **SE1-2 Standards** | Systems engineering standards and models | | 1-14 | | | | | | | | (Badiru, 2019) |
| **MIL-STD 499B** | (70s) It is a SE standard based on: [1] system performance parameters (operational needs), [2] technical efforts (development, manufacturing, verification, deployments, operations), [3] system configuration, [4] WBS (cost, schedule), and [5] information. | ANSY DES IMP | 1-12 | ABS | QT | PDS DEV | ITE | DD | Yes | Math Graphs Models | (Buede, 2009) (Liu, 2015) |
| **ANSI/EIA-632** | (90s) It addresses potential consequences of fundamental processes such as: [1] acquisition, [2] technical management, [3] system design, [4] product realization, and [5] technical evaluation. | ANSY DES IMP | 1-13 | ABS | QT | PDS DEV COMB | ITE | DD | Yes | Math Graphs Models | (Buede, 2009) (Valerdi and Wheaton, 2015) |
| **ISO/IEC/IEEE** | (2002) ISO/IEC/IEEE 15288 includes systems engineering, software engineering, and systems lifecycle (24765, 29148, 42010, 15289, 15939, 16085, 24748-4, etc.) | ANSY DES IMP | 1-13 | ABS | QT | PDS DEV COMB | ITE | DD | Yes | Math Graphs Models | (INCOSE, 2015) (Buede, 2009) |
| **COSYSMO** | (2003) It includes SE lifecycle phases, processes, and models: [1] conceptualize, [2] development, [3] operational test and evaluation, [4] transition to evaluation, [5] operations, maintenance, enhancement, and [6] replacement and dismantling. | ANSY DES IMP | 1-14 | ABS | QT | PDS DEV | ITE | DD | Yes | Math Graphs Models | (Valerdi et al., 2003) (Badiru, 2019) |
| **CMMI** | (2002) It is a process-based protocol, using process maturity levels such as: [1] initial, [2] managed, [3] defined, [4] quantitatively managed, and [5] optimizing. | ANSY IMP | 1-10 | ABS | QT | DEV | ITE | DD | Yes | Models | (CMMI Product Team, 2018) (Humphrey, 1988) (Kamrani and Nasr, 2010) |
| **XMI** | Metadata interchange is an OMG metamodeling standard for exchange information via XML. | ANSY DES IMP | 1-14 | ABS | QT | DEV | NET | DD | No | Models | (OMG, 2015) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **MOF** | Meta-object facility is an OMG model-driven engineering standard for CORBA architectures. | ANSY DES IMP | 1-14 | ABS | QT | DEV | NET | DD | No | Models | (OMG, 2019) (Gaševic et al., 2006) |
| ***SE2 - System Engineering Models and Paradigms*** | | | | | | | | | | | |
| **SE2-1 Document-based** | Document-based models and techniques | | 1-10 | | | | | | | | |
| **SEMP** | SE management plan includes process planning, requirement analysis, functional analysis, synthesis, control analysis, technologies, risks, integration efforts, activities, schedule, and metrics. | IMP | 1-2, 10 | ABS | QT | DEV | ITE | DD | No | Document Schedule Workflow | (Martin, 1996) (Kamrani and Nasr, 2010) (Liu, 2015) (INCOSE, 2015) |
| **SEMS** | SE master schedule is an event-based document based on milestones, relationships, and selected criteria. | IMP | 1-2, 10-11 | ABS | QT | DEV | ITE | DD | No | Document Schedule | (Liu, 2015) (Martin, 1996) (Kamrani and Nasr, 2010) |
| **SEDS** | Systems engineering detailed schedule. Calendar-based task schedule. | IMP | 1-2, 10 | ABS | QT | DEV | ITE | DD | No | Document Schedule | (Martin, 1996) (Kamrani and Nasr, 2010) |
| **WBS** | Work breakdown structure allows to schedule and track efforts, tasks, resources (hardware, software, data, infrastructure), items and services. It is Gantt chart based. | IMP | 1-2, 10 | ABS | QT | DEV | ITE | DD | No | Document Schedule | (Martin, 1996) (Kamrani and Nasr, 2010) |
| **Requirements Doc.** | These are capturing document addressing systems objectives and thresholds. | ANSY DES | 1-4 | ABS | QT | PDS FPDS | LIN | DD | No | Document | (Martin, 1996) |
| **TPM** | Technical performance measurement is a progress assessment document. It is used for risk mitigation. | IMP | 8-10 | ABS | QT | DEV | ITE | DD | No | Document | (Martin, 1996) (Kamrani and Nasr, 2010) |
| **FFBD** | Functional flow block diagram is a multilevel, step-by-step graphical document showing the sequence of operations. Elements: functional graphic block, function (identification, connection, and flow) directions, changes, and stopping criteria. | ANSY IMP | 1-11 | ABS | QT | DEV | Net | DD | No | Document Diagram | (Badiru, 2019) (Liu, 2015) (Defense Acquisition University, 2005) |
| **INCOSE SEFT** | (1990s) It is an evidence-based SE competency framework: [1] management, [2] professional, [3] core principles, [4] technical competencies. It presents individual and organizational applications. | ANSY IMP | 1-11 | ABS | QT/QL | DEV | N/A | DD | N/A | Models | (Liu, 2015) (Badiru, 2019) (INCOSE, 2015) (Buede, 2009) |
| **SE2-2 Lifecycle-based** | Lifecycle-based SE methodologies | | 1-15 | | | | | | | | |
| **TDSE** | It is a top-down SE (multiple levels) including analysis, definition, and verification. | ANSY DES IMP | 2-8 | ABS | QT | PDS DEV | ITE | DD | No | Workflow Models | (Buede, 2009) |
| **Vee** | It has several sides: [right side] design (requirements), [bottom] implementation, and [left side] fabrication. There are variations: incremental and iterative development (IID), Vee model XT (extreme tailoring), etc. | ANSY DES IMP | 1-11 | ABS | QT | PDS DEV COMB | ITE | DD | Yes | Workflow Models | (Forsberg and Mooz, 1992) (Fairley and Forsberg, 2020) (INCOSE, 2015) (Aughenbaugh & Paredis, 2004) |
| **Waterfall** | It is an iterative sequential process with loops between subsequent design phases. It is simple, frugal, and not very adaptable. | ANSY DES IMP | 1-15 | ABS | QT | PDS DEV COMB | WA | DD | No | Workflow Models | (Liu, 2015) (INCOSE, 2015) (Buede, 2009) (Haberfellner et al., 2019) |
| **Spiral** | It is based on [1] objectives, [2] evaluation, [3] development, and [4] planning of next phase, covered with subsequent passes. | ANSY DES IMP | 1-14 | ABS | QT | PDS FPDS | SPI | DD | No | Workflow Models | (Liu, 2015) (INCOSE, 2015) (Buede, 2009) (Kamrani and Azimi, 2010) (Boehm, 1988) |
| **ICSM** | Incremental commitment spiral model is a | ANSY | 1-14 | ABS | QT | PDS | SPI | DD | No | Workflow Models | (INCOSE, 2015) (Haberfellner et al., |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | valued-based, concurrent, and fast methods based on IID tackling product & processes. | DES IMP | | | | DEV COMB | | | | | 2019) (Boehm et al., 2014) |
| **SIMILAR** | It is wheel-structured with networked steps: [1] problem statement, [2] alternatives, [3] system model, [4] integration, [5] system launch, [6] performance, [7] re-evaluation. | ANSY DES IMP | 1-14 | ABS | QT | PDS DEV COMB | NET | DD | No | Workflow Models | (Haberfellner et al., 2019) (Bahill and Madni, 2017) |
| **DEJI** | It includes [1] design, [2] evaluation, [3] justification, and [4] integration. It also addresses quality system integration. | ANSY DES IMP | 1-15 | ABS | QT/QL | PDS FPDS | FLEX | DD | No | Document Workflow Models | (Badiru, 2019) |
| **Walking Skeleton** | It is based on [1] information gathering, [2] reflection workshop, [3] blitz planning, [3] delphi estimation, [4] daily stand-ups, [5] agile design, [6] process miniature, [7] side-by-side programming, [8] burn charts. | DES IMP | 1-14 | ABS | QT | PDS FPDS COMB | ITE NET FLEX | DD | Yes | Code Workflow Models CAD | (Badiru, 2019) |
| **Concurrent SE** | It is a simultaneous and past-pace SE process that is based on parallel and incremental partial design cycles. | ANSY DES IMP | 1-15 | ABS GEO | QT/QL | PDS FPDS DEV | NET | SA | Yes | Document Workflow Models CAD | (Backhouse and Brookes, 1996b) (Salomone, 2019) (Haberfellner et al., 2019) |
| **SE2-3 Cross-cutting** | Multidisciplinary methodologies and models | | 1-5 | | | | | | | | |
| **Agile SE** | This includes simultaneous fast-paced and method-based SE techniques with interconnected cycle under the agile manifesto. Many tools and techniques: XP, SCRUM, FDD, etc. | ANSY DES IMP | 1-15 | ABS GEO | QT/QL | PDS FPDS DEV | NET | SA | Yes | Document Workflow Models CAD | (Haberfellner et al., 2019) (Douglass, 2016) (Larman, 2004) (Huang et al., 2012) |
| **System Thinking** | It is a complex system description based upon multidisciplinary synergies. Presents a graphical representation of relationships. | ANSY DES | 1-7, 12-13 | ABS | QT/QL | PDS FPDS DEV COMB | FLEX NET | SA | No | Document Models | (Boardman and Sauser, 2013) (Boardman and Sauser, 2008) |
| **OOSE** | Object-oriented SE. Basic elements are integrated into a system. Specialization is based on inheritance and 'black-box' models and components. | ANSY DES IMP | 1-11 | ABS | QT | PDS FPDS DEV COMB | FLEX NET | DD | No | Document Workflow Models | (Buede, 2009) (INCOSE, 2015) (Morris et al., 2012) |
| **OOAD** | Object-oriented analysis and design. Agile SE model. Data, processes, and systems are turned into objects, and managed by one person. System manager puts everything together. People-driven. Small teams. | ANSY DES IMP | 1-11 | ABS | QT | PDS FPDS DEV | NET | DD | No | Document Workflow Models | (Badiru, 2019) (Ramnath and Dathan, 2010) |
| **FBSE** | Function-oriented SE includes: [1] top-level functions & performance requirements, [2] lower-level function definition, [3] lower-level function evaluation, [4] iterations, [5] sub-functions, [6] sub-requirements, [7] alternatives, and [8] interfaces. | ANSY DES IMP | 1-11 | ABS | QT | PDS FPDS DEV | FLEX NET | DD | Yes | Diagram Simul. Models | (INCOSE, 2015) |
| **IDP** | Integrated product development (IDP) is a process-oriented full lifecycle approach. It is based on a continuous integration of cross-functional teams. Its characteristics include: [1] decentralized, [2] design-to-manufacturing integration, [3] interface control, [4] concurrent, [5] fast-pace agile, and [6] multidisciplinary. | ANSY DES IMP | 1-15 | ABS | QT | PDS FPDS DEV COMB | FLEX NET | SA | No | Document Workflow Models | (INCOSE, 2015) |
| **MBSE** | Model-based SE is a multidisciplinary, full-lifecycle, and agile methods. It is based on interconnected elements (e.g., requirements, concepts, test cases, verification plans, trade studies, etc.). It enables multiple improvements: [1] better rigor and repeatability, [2] more quality, [3] risk reduction, and [4] enhanced communications across disciplines and people. It has multiple workflows, tools, and methodologies. | ANSY DES IMP | 1-15 | ABS | QT | PDS FPDS DEV COMB | FLEX NET | DD | Yes | Docu. Workflow Models | (INCOSE, 2015) (Haberfellner et al., 2019) (Badiru, 2019) (Friedenthal et al., 2008) (Fernandez and Hernandez, 2019) (Borky & Bradley, 2018) (Long & Scott, 2011) (Ramos et al., 2012) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Fuzzy SE** | This is a ruled-based SE method based on fussy probabilistic logic for process control. | ANSY DES IMP | 1-10 | ABS | QT | PDS DEV | FLEX | DD | Yes | Math Model Graph | (Pedrycz and Gomide, 2007) (Badiru, 2019) (Nedjah and Mourelle, 2005a) |
| ***SE3 - System Engineering Tools*** | | | | | | | | | | | |
| **Documents** | These are text-based documents (physical & digital) containing parameters, relationships, and components. Documents can be reviewed and edited but they are not linked. | ANSY DES IMP | 1-15 | ABS | QT | PDS DEV | LIN | SA | No | N/A | (Grady, 1995) (Wasson, 2005) (Liu, 2015) |
| **Diagrams** | They are graphics-based representation describing: [1] system components and actors, [2] relationships, [3] flows, [4] functions, and [5] timelines. They can be physical or digital. They are complemented with alphanumeric parameters. Among some of the most relevant are block diagrams, flowcharts, SFG, FFBD, BFFB, RBB, PFD, N2, GANTT, event chain diagrams, CFD, PERT, use case diagrams, sequence diagrams, etc. | ANSY DES IMP | 1-15 | ABS | QT/QL | PDS DEV | LIN | DD | No | N/A | (Karayanakis, 1995) (Nakatsu, 2010) (Levine, 1982) (Liu, 2015) (Shelly and Rosenblatt, 2009) (Turton et al., 2008) (Nicholas, 2004) (Hatley et al., 2013) (Satzinger et al., 2008) (Windle and Abreo, 2003) |
| **Matrixes** | They tackle quantifiable and qualifiable parameters. They are digital or physical. There are many standard techniques and custom applications, such as: DSM, PM, HOQ, RVTM, risk assessment, etc. | ANSY DES IMP | 1-15 | ABS | QT/QL | FPDS DEV | N/A | SA | Yes | N/A | (Madani et al., 2014) (Burqe, 2009) (Muller et al., 2011) (Madu, 2006) (Liu, 2015) (Popov et al., 2016) |
| **Graphs** | These are graphical mathematical and statistical tools. | ANSY DES IMP | 1-15 | ABS | QT/QL | FPDS DEV | N/A | SA | Yes | N/A | ( |
| **Codes** | These include scripts, data management techniques, and tools. | ANSY DES IMP | 1-8, 13-15 | ABS | QT | FPDS DEV | LIN | DD | Yes | N/A | (Parnell et al., 2011) (INCOSE, 2015) |
| **Maps** | They include information illustrations, communication, decision making, etc. Examples are mind maps and systemigrams. | ANSY DES | 1-8, 13-15 | ABS | QL | PDS DEV | LIN | SA | No | N/A | (Haberfellner et al., 2019) (Squires et al., 2010) |
| **Analysis** | These are mathematical, physics, or multi-physics in nature. Applications include regression, feasibility, FEM, FEA, risk, etc. | ANSY DES IMP | 1-8, 13-15 | ABS | QT | PDS DEV | LIN | DD | Yes | N/A | (Badiru, 2019) |
| **Simulations** | Testing (parameter quantification) and demonstrations (result qualification). | ANSY DES | 2-15 | ABS | QT | FPDS DEV | LIN | DD | Yes | N/A | (INCOSE, 2015) |
| ***SE4 - System Engineering Languages*** | | | | | | | | | | | |
| **IDEF** | Integration definition for functional modeling is modeling family of languages, including: IDEF0 (functional), IDEF1(information), IDEF1X (data), IDEF3 (processes), IDEF4 (object-oriented dee.), & IDEF5 (ontologies). | DES IMP | 1-15 | ABS | QT | DEV | NET | DD | No | N/A | (Buede, 2009) (Haberfellner et al., 2019) |
| **USL** | Universal systems language is based on axioms, error-testing principles ontologies. It presents functions maps (FMaps) and type maps (TMaps). It has control applications. | ANSY DES IMP | 1-15 | ABS | QT | DEV | NET | DD | No | N/A | (Hamilton and Hackler, 2009) |
| **UML** | Unified modeling language is a multi-purpose graphic modeling language. It has three types of diagrams: [1] structural, [2] behavioral, and [3] interaction. | ANSY DES | 1-15 | ABS | QT | DEV | NET | DD | No | N/A | (Haberfellner et al., 2019) (Borky & Bradley, 2018) |
| **LML** | Lifecycle modeling language is an open-standard user-friendly SE modeling language. Full lifecycle: concept, use, support, and retirement. ERA: Classes (entity), relations (relationship), and properties (attribute). It includes risk, cost, | ANSY DES IMP | 1-15 | ABS GEO INT | QT | DEV | NET | DD | No | N/A | (Hettema, 2013) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | schedule, and performance (form, function, metric, and interface) in its ontology. | | | | | | | | | | |
| **SysML** | Systems modeling language is a standardized graphical SE modeling language based upon UML. Elements / diagrams include: [1] structure (block definition, internal block, package), [2] requirements, [3] behavior (activity, sequence, state machine, use case), and [4] parametrics. Release by OMG SysML™. | ANSY DES IMP | 1-11,13 | ABS | QT | DEV | NET | DD | No | N/A | (Friedenthal et al., 2008) (Buede, 2009) (Estefan, 2008) (INCOSE, 2015) |
| **Drakon** | It is a visual programing modeling language using flowcharts as letters, which can be used to graphically create words and sentences using a syntax. Textual syntaxes from other languages can be added. | ANSY DES IMP | 1-11,13 | ABS | QT | DEV | NET LIN | DD | No | N/A | (Ivannikov, 1995) (Schwarzbach et al., 2015) |
| **Math Languages** | Mathematical coding languages provide SE and MBSE support to run scripts, provide analysis tools, etc. Examples of them are AML, MATLAB™, Mathematica™, etc. | ANSY DES | 1-11,13 | ABS | QT | DEV | NET | DD | Yes | N/A | (Kallrath, 2004) (Magrab, 2014) |
| **Coding Languages** | Generic coding languages are used for SE and MBSE purposes. These include object-oriented, imperative, declarative, concurrent, visual, multimedia-based, web-based, event-based, and integrative languages. | ANSY DES IMP | 1-15 | ABS | QT | DEV | NET | DD | Yes | N/A | (Bansal, 2013) (Friedman and Wand, 2008) |
| ***SE5 - System Engineering Frameworks*** | | | | | | | | | | | |
| **CASE** | *Software*. It is a computer-aided toolset for SDLC including: [1] project identification and selection, [2] project start and planning, [3] analysis, [4] logical and physical design, [5] implementation, and [6] maintenance. | ANSY DES IMP | 1-11,13 | ABS | QT | DEV | NET PDS | DD | No | N/A | (Valacich et al., 2017) (Berdonosov and Redkolis, 2010) (Shelly and Rosenblatt, 2009) |
| **RUP** | *Software*. Rational unified process is an iterative object-oriented software Dev. framework covering [1] inception, [2] elaboration, [3] construction, [4] construction. | ANSY DES IMP | 1-11,13 | ABS | QT | DEV | NET PDS | DD | No | N/A | (Kruchten, 2004) (Valacich et al., 2017) |
| **TOGAF** | *Enterprise Framework*. The open group architecture framework includes tools for information technology enterprise Arch. | ANSY DES IMP | 1-6, 9-11, 13 | ABS | QT | DEV | NET | DD | No | N/A | (INCOSE, 2015) (Dickerson and Mavris, 2016) |
| **FEAF** | *Enterprise FW*. Federal enterprise architecture framework is for business and technology integration. | ANSY DES IMP | 1-6, 9-11, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Kappelman, 2009) |
| **DoDAF** | *Enterprise FW*. Department of defense architecture network is a large SE visualization framework and model-driven analysis and simulation framework. | ANSY DES IMP | 1-6, 9-11, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Mittal and Martín, 2018) (Liu, 2015) |
| **MoDAF** | *Enterprise FW*. British ministry of defense architecture framework, includes seven key views: technical standards, strategic, operations, service, system, and acquisition. | ANSY DES IMP | 1-6, 9-11, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Dickerson and Mavris, 2016) (Babers, 2015) |
| **Zachman FW** | *Enterprise FW*. Enterprise ontology is based on identification, definition, representation, specification, configuration, and instantiation. | ANSY DES IMP | 1-6, 9-11, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Zachman, 1987) (Zuech, 2002) |
| **Harmony SE** | *SE*. This model-driven development environment is based on: [1] requirements analysis, [2] system function analysis (identification, states, modes, physical arch.), and [3] architecture design and synthesis. | ANSY DES IMP | 1-10, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Ramos et al., 2012) (Estefan, 2008) |
| **OOSEM** | *SE*. Object and scenario-driven is a Vee environment process based on: [1] Stakeholder needs analysis, [2] system requirement analysis, [3] logical architecture | ANSY DES IMP | 1-10, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Ramos et al., 2012) (INCOSE, 2015) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | definition, [4] physical architecture synthesis. | | | | | | | | | | |
| **RUP SE** | *SE*. (IBM Rational) It is a Model-driven ICSM system development framework based on an spiral model. Its lifecycle presents four stages: [1] inception, [2] elaboration, [3] construction, and [4] transition. | ANSY DES IMP | 1-10, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Ramos et al., 2012) (INCOSE, 2015) (Estefan, 2008) |
| **ARCADIA** | *SE*. Architecture analysis and design integrated approach is a SE development framework for software and hardware. It is based on three activities: [1] need analysis and modeling, [2] architecture and validation, [3] requirements engineering. | ANSY DES IMP | 1-10, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Brusa et al., 2017) |
| **Alstom ASAP** | *SE*. Advanced system architecture program is a top-down SE application with multiple views including: [1] operational, [2] functional, and [3] constructional. | ANSY DES IMP | 1-10, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Fanmuy et al., 2016) |
| **Vtech MBSE** | Vtech corporation concurrent environment tackles [1] requirements analysis, [2] behavior/functional analysis, [3] architecture synthesis, and [4] verification and validation. It is based on elements, relationships, structure, entities, attribute of relationships, and attributes, using an 'onion' SE model. | ANSY DES IMP | 1-10, 13 | ABS | QT | DEV | NET | DD | No | N/A | (Estefan, 2008) (Brusa et al., 2017) |
| **OPM** | *SE*. This is a holistic modeling language and methodology based on objects and processes for natural and artificial systems. It is based on: [1] function (what they do), [2] structure (how they are constructing), and [3] behavior (how the change over time). OPM tackles: [1] requirement specification, [2] analysis & development, and [3] implementation using OPD (object-process diagrams) & OPL (object-process language). | ANSY DES IMP | 1-14 | ABS | QT | DEV | NET | DD | No | N/A | (Dori, 2016) (Estefan, 2008) (Brusa et al., 2017) |
| **BIM** | *Arch*. Building information modeling provides a framework to plan, create, manage, and modify digital representation of buildings tackling functions, properties, phases, and designs across the lifecycle. There are several levels: [1] 4D BIM (3D components + scheduling), [2] 5D BIM (adding cost related information) and [3] 6D BIM (adding the dimension of operations, maintenance, etc.) It also tackles sustainability and recycling. | ANSY DES IMP | 1-15 | ABS GEO | QT QL | FPDS PDS DEV COMB | NET | SA | Yes | N/A | (Eastman et al., 2011) (Smith and Tardif, 2009) |
| **PLM** | *Eng*. Product lifecycle management connects information about people and the lifecycle of a product or an enterprise. PLM serves multiple phases across the system: [1] systems engineering (requirements, variations, reliability), [2] product portfolio, [3] product design, [4] manufacturing, and [5] product data management. | ANSY DES IMP | 1-10 | ABS GEO | QT QL | FPDS PDS DEV COMB | NET | SA | Yes | N/A | (Elangovan, 2020) (Tyulin and Chursin, 2020) |
| **DOORS** | *Eng*. Dynamic object-oriented requirements system is a requirement system developed for requirement optimization, communications, verification, and collaboration. It links requirements and stakeholders collaboratively. | ANSY DES | 1-9 | ABS | QT | FPDS PDS DEV COMB | NET | DD | No | N/A | (Stjepandić et al., 2015) |
| **Microsoft Office** | It is widely distributed. Spreadsheets, schedules, databases, etc. are used by individuals and organizations across the world, both discreetly and interconnected. | ANSY DES IMP | | ABS | QT | FPDS PDS DEV COMB | NET | DD | No | N/A | (Badiru, 2013) |

*Table 15. Systems engineering methods, theories, and tools.*

### 3.2.5. Conclusion

Based on this literature review of state-of-the-art systems engineering theories, standards, models, languages, and frameworks several conclusions and gaps can be drawn from the perspective of hardware-based systems development:

- **Geometry**. While SE methodologies are developed to handle complex systems, they do not have a clear way to deal with geometrical information or to connect with non-SE geometrical design tools such as CAD or BIM. Managing abstract analytic information is at the core of many of these approaches since they evolved from software development techniques, but they do not handle physical relationships natively. Frameworks such as BIM, PLM, and concurrent CAD present platforms capable of handling analytical parameters, but do not handle complex relationships among them. Concurrent (Salomone, 2019) and agile SE (Douglass, 2016) approaches tackle the management of geometrical information but do not present a specific framework or toolset to represent them. On the other hand, LML (Hettema, 2013) tackles some geometrical aspects in the language approach, but it does not seem to present a clear workflow or interface either with complex geometry-driven frameworks. The development of complex hardware-based system architectures needs a multidisciplinary combination of quantifiable (analytical) and qualifiable parameters (including geometry), as well as relationships among them that evolve over time during multiple design cycles.
- **Continuity**. Most of all addressed methods present quite a compartmentalized approach between phases and steps across the lifecycle of a system or enterprise. The development of complex systems often requires multiple and iterative design cycles. Among all these methods two approaches present a unique approach tackling the SE development as a continuous process: [1] the skeleton SE method (Badiru, 2019) and [2] IID approaches such as ICSM (INCOSE, 2015). However, both approaches do tackle specifically geometrical information. SE techniques tend to specifically disregard whether the scope of the method is a single-point solution or families of solutions beyond a single instantiation of key parameters. Among them, OPM (Dori, 2016) does take into account the evolution of the system from the beginning, enabling the use of adaptable language and diagrams that describe such system change.
- **True full cycle**. Similarly, most SE methodologies do not consider some phases of the lifecycle such as communication, marketing, recycling, and decommission. SE practices consider more design and analysis aspects than implementation topics. They address implementation management, but not necessarily detailed areas such as manufacturing, testing, analysis, etc. While languages and tools can be used in principle across the full lifecycle of a system, model, theory, and framework they need to include these last phases as part of their teleology, otherwise they would not be really integrated in their workflow. Thus, closing the design loop is key so an SE process could be reinforced, feedbacked, and improved with the information coming from the last phases of a project development.
- **Synergy**. Besides generic SE frameworks such as BIM and reviewed toolsets (documents, matrixes, graphs), all system engineering methodologies present a very discipline-oriented approach. The type of challenges and problems these methods tackled are multidisciplinary in nature, but their approach tends to divide the problem by discipline. Concurrent and agile techniques are closer to a more multidisciplinary and holistic approach than the rest since multiple disciplinary problems are addressed faster, and in more detail, than other techniques. Along those lines, the Vtech MBSE framework presents an 'onion' model similar to the walking skeleton and IID methods. Among these, multiple design and SE cycles occur much faster to subsequently increase the level of definition. A fast-paced approach increases interactions and brings a more synergetic approach just by temporal proximity in the lifecycle.
- **Optimization**. While SE techniques can be used in optimization activities and efforts, these methods and theories do not necessarily embrace the optimization as a specific part of the theoretical workflow. Optimization is often perceived as a task, rather than a part of the development lifecycle process itself. As a result, optimization tends to be addressed during the design phase only providing feedback to reassess or modify analysis and implementation later on.
- **Flexibility**. Finally, there is a gap that connects all previous points. All these methods present a clear and defined workflow that tackles analysis, design, and implementation separately. While the practice of SE and DE often requires multiple in-between steps across phases, such theoretical frameworks present a significantly rigid structure. However, all these methods do allow to have more flexible workflows while the use of related toolsets also enable a more flexible utilization. As such, all of them with the exception of some design framework such as BIM, tackle efficiently quantifiable and non-geometrical parameters.

## 3.3. Evolutionary Principles: Nature, Engineering, and Design

### 3.3.1. Approach

The concept of evolution as well as subsequent evolutionary mechanisms have outgrown the purely biological realm and influenced already many technical fields such as software design, bio-engineering, and system engineering, among other disciplines (Hingston et al., 2008). While this is indeed a growing and under-development field, this literature review section addresses concepts, mechanism, principles, and applications that not only are a foundational base for this research. Furthermore, these also highlight key gaps along this new paradigm especially regarding hardware-based system architectures. Biological evolution is one of nature's mechanisms to deal with change and entropy, and it is also the approach behind how complexity emerges within biological systems in general (Ray, 1994).

In physics, evolution is the approach of a system to its thermodynamic equilibrium defined by an increase in entropy (second law of thermodynamic). On the other hand, from a biological perspective it means an increase in the complexity of its structure and internal connections. This contradiction is reconciled by the scale at which this is applied while it also brings the notion of stability (Chakrabarti and Ghosh, 2011). In essence, evolution is the engine of complexity (Mayfield, 2013) and mastering its methodology has served of inspiration in multiple technical fields, such as software development to increase, manage, and harness the inherent complexity of a system. As Braha, Minai, and Bar-Yam exposed is their complex engineered systems book, when it comes to complex systems current paradigms of goal-oriented reductionist analysis and centralized control are not capable of handling very large or very complex systems (Braha et al., 2007). At the same time, evolution-driven principles of adaptability, self-organization, resilience, and scalability, among others serve very well upcoming systems engineering and design challenges in an era of increasing complexity and more global scarcity.

Section 3.3.2 presents an organized literature review of all these topics across multiple technical fields. This is based on a series of key overarching evolutionary principles (EVPs) as shown in Table 16. Furthermore, Table 17 presents a detailed summary of each method, principle, and technique addressed in the literature review. This last table also presents the same format and scale than the DE and SE review tables to make any further comparison and reference easier.

| ID Code | Field of study | Theories | Principles | Models | Concepts | Applications | Description | Driven by | Applied Domain |
|---|---|---|---|---|---|---|---|---|---|
| Evo1 | **Natural Evolution** | X | X | | | | This includes modern biology theories, evolutive mechanisms, and principles. | *Nature* | *Biology* |
| Evo2 | **Computer Science** | | | X | X | | This is about methodologies and theories based on the previous point and applied to a modern use of computer science tackling complex engineering challenges. | *Data* | *Optimization* |
| Evo3 | **Software Design** | | | X | | X | It is a subset within the previous computer science point that is applied to fast-paced software development techniques. | *Process* | *Agility* |
| Evo4 | **Systems Engineering** | | | X | | X | This field includes all the above points to tackle more efficient systems engineering methods across the system lifecycle. | *Workflow* | *Efficiency* |
| Evo5 | **Hardware Design** | | | | X | | Examples and concepts using this approach are included in this category. | *Object* | *Optimization* |

*Table 16. Fields and scope of evolutionary methods.*

## 3.3.2. Literature Review Summary

These sections elaborate and summarize some of the most relevant principles across key fields, as well as their relevance towards their application, usage, and relationship to hardware-based system design engineering.

### *3.3.2.1. Biological Evolution (Evo1)*

Within the scientific study of natural evolution, several key principles and concepts could be understood as very relevant in the context of this research, independently of their biological nature. They provide approaches validated by nature which have inspired multiple new methodologies specially in the computer science and software domain. Among some of the most relevant principles in this realm we could find the following:

- **Natural selection** is a key evolutionary mechanism, based on how certain phenotypes (physical observable characteristics of an organism) increase their statistical survival above other less fitted traits, ensuring their reproduction and continuation as Drawing and Wallace described (Herron and Freeman, 2013). But selection is also an organized route in itself against complexity (Bell, 1996). While entropy at large continues to increase, evolution tends towards a greater level and order. Evolution only happens through natural selection if there are enough genetic variations. At the most basic level, natural selection is based on inherited variations within species that allow them to have a survival advantage (Zeigler, 2014). This mechanism also considers destructive processes such as predation, competition for resources, climate changes, and diseases, among others. Figure 90 presents a graphical representation of this process (Wykis, 2007). This basic evolution mechanism describes and predicts life evolution on Earth (Dobzhansky, 1973) and it was later connected to genetics by the *modern synthesis* theory. ***Relevance.*** Natural selection raises two concepts that relate to both DE and SE such as [1] the optimization of the solution is performed by multiple variations that are being tested against environmental requirements, and [2] previous solutions serve as leverage towards a new generation that become heritage. ***Application.*** This principle today is relevant at different levels because the evolution of products and services can be understood as small trending variations among multiple brands and customer feedback evolving towards an established design within their business ecosystem. In broader terms and with some caveats the concept of evolution also applies to technology development as Ziman pointed out (Ziman, 2003). Furthermore, the concept of finding more fitted solutions (or approximations) based on modifications within a given context is in essence the foundational principle of genetic algorithms developed in the 1990s for software programing (Forrest, 1993; Koza, 1994).

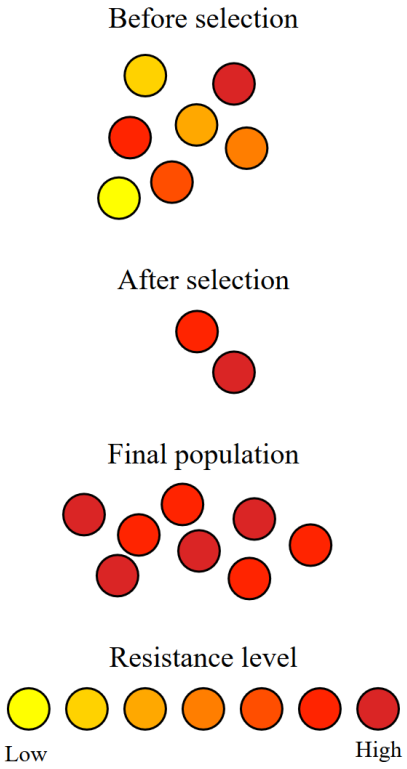



*Figure 90. Natural selection applied to antibiotic resistance, after Wykis (2007).*

- **Evolutionary developmental biology ('evo-devo') deep homology and pleiotropy**. Evolution is about descendants with modifications and represents changes at both genetic and organism levels, enabling permanent changes in all species (phylogeny). On the other hand, development is about how an organism is produced (ontogeny) within its own individual time scale (Hall, 2012a). The evo-devo approach studies and compares those ontogeny processes to infer phylogenetic relationships (Arthur, 2002). In essence, ontogeny produces phylogenetic change and evo-devo studies how embryonic changes during one generation relate to evolutionary changes at the species level, and in any stage of the life cycle (Hall, 2012b). With the development of molecular genetics, evo-devo also studies the relationship between physical traits (phenotype) and genetics (genotype). Among many mechanisms discovered within this approach, deep homology describes how certain genes (*hox genes*) control the development

process (growth and differentiation) within a species, and across many species (Hall and Olson, 2003). Figure 91 show this concept graphically (Hueber et al., 2010). Eco-eco-devo incorporates the notion of ecology into this retro-alimented cycle. Furthermore, across the smaller portion of genes controlling the development, one gene affects multiple and unrelatedly phenotypic characteristics. This process is called *pleiotropy* (Miglani, 2010). ***Relevance***. At a high level, this approach brings a couple of important points in the development of any complex system, such as [1] the development of the system affects the system itself and possible futures generations and [2] a small control mechanism guiding the development process affects multiple subsystems. Regardless of the biological origin of this approach, these principles can apply to both system design and systems engineering developments. The notion that the process intrinsically affects point designs as well as whole families of designs, provides a much broader and richer perspective with multiple design consequences. ***Application.*** Evo-devo principles are being explored for software and hardware developments especially toward optimization and computer generation of solutions without human involvement, with applications in urbanism and architecture for instance (Richards et al., 2012).

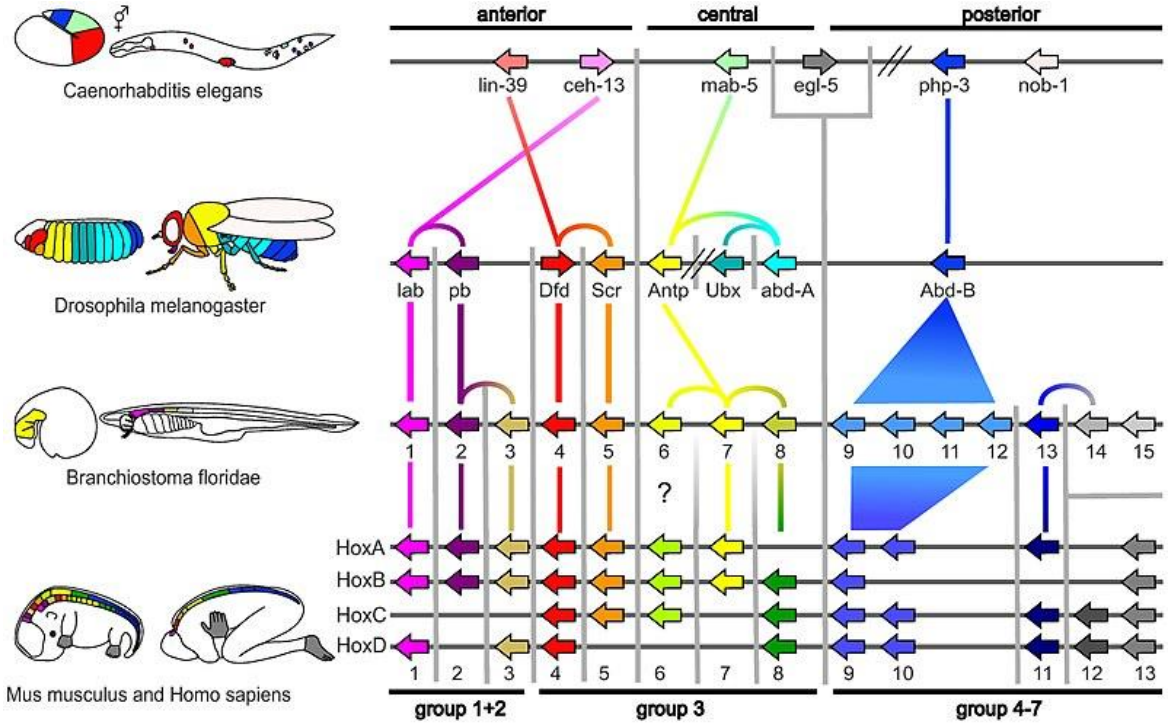



*Figure 91. Hox genes across species after Hueber et al. (2010).*

- **Speciation**. In such evolutionary path, one species could evolve into multiple other species over time (speciation), thus becoming the common ancestor of those (Dieckmann et al., 2004). Populations that are isolated from each other can evolved into new species driven by adaptive mechanisms such ecological, reproductive (Dieckmann et al., 2004), and artificial divergence. These mechanisms include: [1] geographical separation, [2] small population entering an isolated niche, [3] entering a new connected niche, [4] and a population going through genetic changes (Butlin et al., 2009; Karonen, 2006). ***Relevance***. While implications of this simple concept have multiple consequences and theories in the literature, the essence is that the creation of diversity (and therefore all ties between individuals) is part of the evolutionary process. ***Application.*** Software techniques such as digital forensics explore these principles (Cooper, 2005).

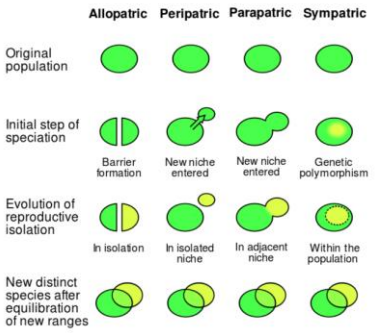



*Figure 92. Speciation mechanisms after Karonen (2006).*

- **Mutations** are changes in DNA molecules leading to the genetic variation of species. These genetic changes may lead to phenotype changes, and these to an evolutionary advantage in the natural selection process (Nei, 2013). ***Relevance.*** Change is at the core of evolution, and nature show us that smaller variations lead over time to greater complexity and more efficiency in any environmental context. Regardless of the biological or artificial nature of a system, this mechanism applies to the notion of continuity and the approach towards what design methodologies are aimed. ***Application.*** This principle can be found across software testing techniques (King and Offutt, 1991), structural design optimization (Burns, 2002), and genetic algorithms (Srinivas and Patnaik, 1994), among others.
- **Coevolution** happens when two or more (guild) species affect each other evolution through natural selection (Thompson, 2005), and it is one of the most powerful mechanism on the Earth ecosystem. Multiple paths lead to coevolution including predator-prey, host-parasite, or mutualism (e.g., flowers and insects), and they imply some specialization of species involved (Thompson, 2009) while bringing a geographical standpoint. ***Relevance.*** Coevolution goes beyond biology, affecting from systems engineering and computer science to culture and human diversity (Durham, 1991). The notion that unrelated systems can help, support, and thrive upon each other, is at the core of any complex system as well as it could be within systems engineering efforts. Thus, the duality of specialization and co-evolution applies to all kinds of systems, natural or otherwise. ***Application.*** This mechanism is found across many technical fields such as: [1] computer science to develop coevolutionary algorithms for artificial intelligence and machine learning (Potter and Jong, 2000), [2] cosmology and astronomy (Ho, 2004), [3] manufacturing, applied for instance to the coordinated development of products, processes, and production systems (Tolio et al., 2010), [4] architecture design, for instance within biomimetic architecture (Mazzoleni, 2013), [5] management such as the coevolutionary NKCS model (Allen et al., 2011), [6] sociology (Durham, 1991), and [7] technology (Lee, 2020), among others.

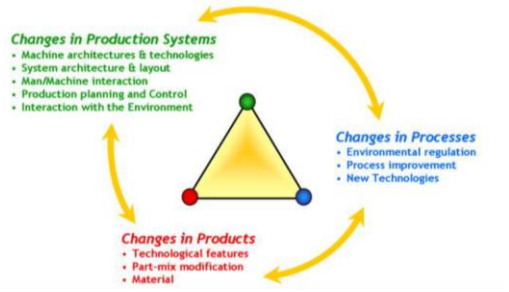



*Figure 93. Co-evolution paradigm after Tolio et al. (2010).*

- **Adaptation**. Furthermore, the aforementioned dynamic evolutionary process leads consequently organisms to become more fitted within their environment and potentially evolve as species if phenotypic changes become hereditary. Adaptation and biological fitness are therefore related, since the last one relates to the genotype and the ability of an organism to pass genes they carry (Werf et al., 2008). ***Relevance***. Physical self-organization and evolutionary adaptation are also interconnected (Vijver et al., 2013a) since a higher level of evolutionary adaptability (based on variations and selection) requires a higher level of self-organization among components and their interactions. In general and beyond biology, the adaptation of a system is related to its capability to react against environmental changes influencing the designer to forecast such situations (Levi and Kernbach, 2010). These changes could happen because: [1] a new situation, [2] a new functionality need, [3] a modified behavior, and [4] an optimization of system parameters. ***Application.*** This principle is widely used across multiple fields within design principles and an optimization strategy such as: [1] robotics (Levi and Kernbach, 2010), [2] architecture (Košir, 2019), and [3] artificial intelligence (Holland et al., 1992), among others.
- **Evolvability** is the capacity of a developmental system for its adaptive evolution (Minelli, 2018). The key is to provide phenotypic (physical) variations that become heritable, so they are maintained over time leading to the evolution of the specie through natural selection. This concept is also related to the robustness of an organism and the persistence of certain traits under external perturbations (Wagner, 2013). Robustness against mutation is therefore as important as the capacity of the system to evolve. Thus, biological systems are capable of changing and tolerating change at multiple levels. ***Relevance***. This is a critical aspect to understand how complicated systems can withstand a lot of change (Wagner, 2013), and it can be key to understand heritage as an unchanged but proven solution or species trait. ***Application.*** As well as previous concepts shown, this principle is used in multiple fields such as [1] robotics, optimizing towards arbitrary behavior and creating novel system functions (Long et al., 2018), and [2] system architecture, reducing cost in large systems by managing changes in smaller parts through defeaturing, abstraction, duplication reduction, etc. (America et al., 2010).

- **Self-organization** is the principle by which natural systems become structured through their own internal processes and unlike technological systems which achieve such state through external commands (Yates, 2012). These internal processes lead to patterns at a system-level (Schweisguth and Corson, 2019; Schweitzer, 1997), because of interactions at lower system levels. This requires enough available energy although they appear not to have a clear directive sometimes. Mechanisms leading to self-organization include: [1] positive and negative feedbacks increasing or reducing the magnitude of a perturbance in the system, [2] exploitation and exploration, [3] multiple interactions, and [4] energy versus entropy or disorder (Camazine et al., 2003). A key aspect of this mechanism is the access to information, since organization arises from interactions among individuals that include signals and cues obtained from each other as a work in progress or stigmergy. Self-organized systems are dynamic, flexible, and they present emergent properties through local interactions within a more global order. They are also non-linear, with an organizational hierarchy, and inherently complex, while staying far from the thermodynamic equilibrium. Multiple theories are also behind this principle such as: [1] dissipative structure theories based on a matter/energy exchange balance, [2] synergetic principles based on the coordination or synergy of mechanism between internal components, and [3] catastrophe theories based upon "the long-run and stable equilibrium that can be identified with the minimum of a smooth well-defined function". In other words, it is a transition towards a steady state through unsteady states. **Relevance**. However self-organization not only applies at the organism level, but also at molecular, mineralogy, thermodynamic, behavioral, social, economic, urban, information science, and cultural (Vijver et al., 2013a; Yates, 2012) levels. As Ashby defined in its "Principles of the self-organizing system", any deterministic dynamic system will evolve towards an equilibrium state (attractor), leaving all non-attractor states behind so its evolution will constrain it into the attractor itself (Ashby, 1991; Zhang, 2015). In essence, the multiple literature about self-organization reinforces the idea of an intrinsic self-order for any complex system within a contextual environment. ***Application.*** Self-organization principles and theories are used and applied across multiple fields such as: [1] biology (Camazine et al., 2003), [2] chemistry (e.g., molecular self-assembly Whitesides et al., 1991), [3] systems control (Gershenson, 2007) , [4] cybernetics (Ashby, 1991), [5] thermodynamics (Nagarajan and Ruckenstein, 1991), [6] computer science (Winfree, 2006), [7] socio-economics (Witt, 1997), and [8] linguistics (Zhang and Park, 2008), to name a few.

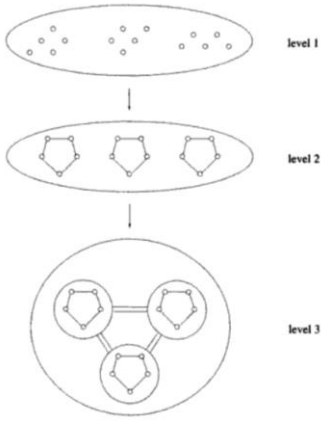


*Figure 94. Emergence of self-organization after Schweitzer (1997).*

- **Evolutionary ecology** is in between ecology and evolutionary biology (Mayhew, 2006) perspectives. This field studies variations (genotype differences) "within individuals, among individuals, among populations, and among species" (Fox et al., 2001), considering the relationship with the physical environment and the effects of performance, behavior, longevity, and fertility. Thus, this implies that is key to understand when a phenotype trait within an organism is caused by its genotype and how much this will drive its natural selection. This approach requires a perspective that takes into account simultaneously time scale, complexity, size, and space (see Figure 95, Pianka, 2011), as well as the complex interactions among species, populations, and individuals. The term system ecology refers to both massive complexity and subtle interactions. However, the limitation in predicting such responses with models that are limited by state-conditioned data is based on the difficulty of inferring new behaviors in any new state going beyond any available data (Pianka, 2011). ***Relevance.*** This field provides a unique perspective regarding complexity because not only it addresses subtle and complex

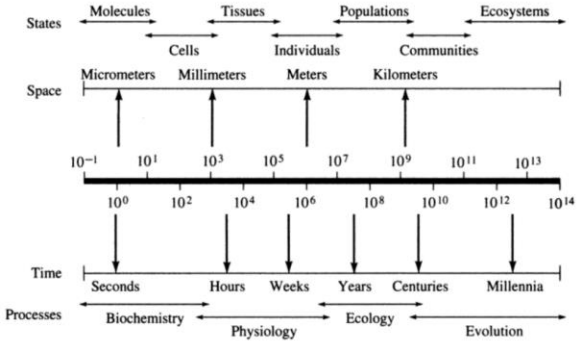


*Figure 95. Time-scale scaling of biological phenomena after Pianka (2011).*

interactions between all components of an ecology, but also relates to different scales within such ecology. Independently of the biological origin, such perspective could also be very applicable to the development of very complex artificial ecosystems, such aerospace, energy, etc. ***Application***. Again there are multiple fields where this approach is being used or based upon, such as: [1] biology, [2] social sciences (Smith, 2017), [3] information processing (Dukas, 1998), [4] technology (Solée et al., 2013), and [5] urbanism (Rivkin et al., 2019).

- **Biological network**. Biological systems at multiple scales (e.g. protein-protein interactions, between-species interactions, food webs, etc.) are better captured by network representations as Figure 96 (Kepes, 2007) shows. In essence, modeling processes in bioscience can be summarized by the following phases: [1] conceptualization, [2] mathematical formalization, and [3] management or optimization (Marin-Sanguino et al., 2019) as seen in Figure 97. Bioinformatics turn such complex systems into building blocks or nodes that interact among them while representing biological units based on graph theory such as genes, molecules, cells, and organisms (Proulx et al., 2005). At the core of this mathematical approach there are several parameters, which are critical to understand the network topology such as: [1] degree distribution or variation in the connectivity to the nearest neighbors P(k) (Kepes, 2007), [2] clustering coefficient representing the ratio of connections or small-world properties of a graphs, [3] assortativity coefficient as a measure of how many edges in a network tend to connect similar nodes (Boccaletti, 2010), [4] *eigenvector* (Guzzi and Roy, 2020) that measures the influence of a node in a network, [5] hierarchy, [6] motifs, and [7] betweenness centrality that defines how central a graph is based on the shortest possible path (Freeman, 1977). ***Relevance.*** This approach applies beyond biological studies and presents a unique way to look at complex systems as networked elements under multiple inherent relationships that are defined by their nature. Complexity of natural systems does not reside only in the number of components (e.g., cells) but also key relationships among them and at multiple levels. ***Application.*** This approach is used across multiple fields in biosciences tackling scales such as molecule and protein interactions, systems (e.g., immune, or neural systems), and ecologies (e.g., system of systems).

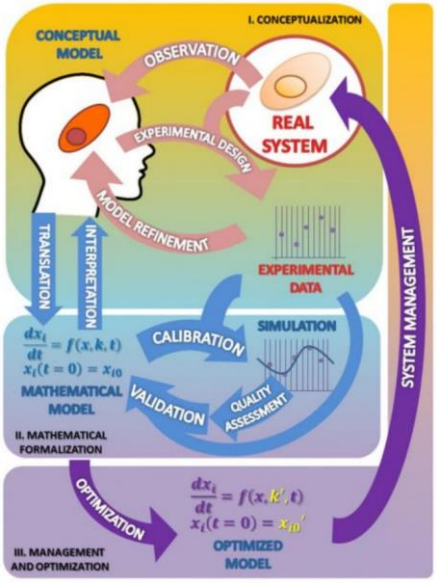



*Figure 97. Modeling in biosciences after Marin-Sanguine et al. (2019).*

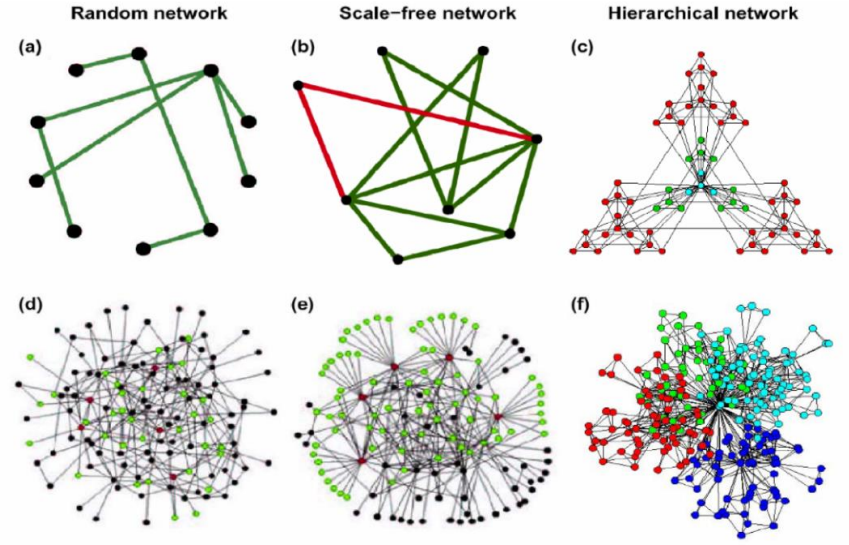



*Figure 96. Graphics presentation of three networks models after Kepes (2007).*

### *3.3.2.2. Evolutionary Computer Science (Evo2)*

The use of evolutionary computational (EC) principles inspired by these biological mechanisms has been widely used since the 1960s across multiple technical fields, with an emphasis in both optimization and artificial intelligence (Jong, 2006). The width of these fields is very extensive and is out of the scope of this research, however it is key to understand some basic principles behind. This subset of computer science is based upon algorithms, models, programming, and strategies (Dumitrescu et al., 2000) with many potential applications to planning, design, simulation, optimization, identification, control, machine learning, scheduling, strategy acquisition, and classification, among others within this realm (Baeck et al., 2018; Bentley, 1999). Evolution is a good problem solver, and has many commonalities with human design (Bentley, 1999).

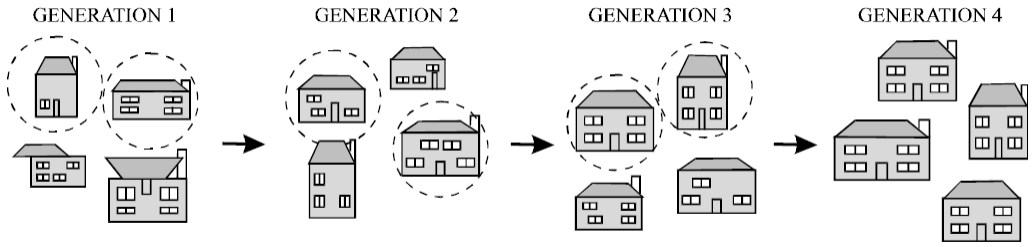



*Figure 98. Generation of evolving house designs (population of 4) after Bentley (1999).*

Interactive evolutionary computation (IEC) is a subset of EC that optimizes a system based upon a subjective human evaluation (Figure 99, Takagi, 2001) with multiple applications in animation, 3D computer graphics, industrial design, speech processing, etc. This approach merges the quantifiable parameter space with the qualifiable psychological space. Similarly, human-based evolutionary computation (HBEC) relies on humans to manage candidate solutions either through a centralized approach (e.g., web server) or a distributed way (e.g., information sharing among people) (Ohnishi et al., 2017; Tan et al., 2017). These could be applied to both selection and evolutionary methods across multiple technical fields.

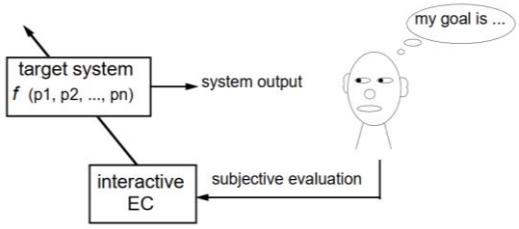



*Figure 99. IEC basic scheme, after Takagi (2001).*

These methods address evolution as an optimization process (Baeck et al., 2018). Generally, they are based upon producing multiple initial solutions which are iteratively and stochastically refined through every following generation until a fitted solution is obtained. This is based upon natural selection principles (Eiben and Smith, 2007). EC is about searching within a space of possible solutions using evolutionary algorithms that combine several techniques (Ashlock, 2006).

**Evolutionary algorithms (EA)** are a subset of EC, and in general they are all based in the natural selection principle. A preliminary algorithmic population is genetically created using a collective learning process for such population, then through a fitness function a parent group is selected to breed a new descendant population that based on multiple randomized processes (Baeck et al., 2018) until the solution is approximated well enough. Among the most relevant EA are **genetic algorithms** (GA), which were developed by John Holland in 1962 (Dumitrescu et al., 2000). These are also based on the survival of the fittest principle with the purpose of designing robust adaptive systems. This metaheuristic process presents an evolution of candidate solutions (chromosomes or fixed-length binary strings) through different search operations such as crossover, mutations, and inversion. Figure 100 shows an example of a genetic algorithm. Within this context, genotypes can be understood as solutions seeds, which map phenotypes or solutions. Figure 102 also presents the general architecture of GA algorithms (Bentley, 1999) that can be summarized in three basic steps: [1] generation of an initial population, [2] selection of a portion of that population to seed the next generation, and [3] development of a next generation from that portion through crossover (recombination of parental genetic information to create a new offspring) and mutation (changing some algorithm chromosomes). In general, the GA method needs a genetic representation of the solution and a fitness function to evaluate it. Advanced genetic algorithms include among others (Bentley, 1999): [1] steady-state (offspring

replaced after generation according to fitness), [2] parallel (multiple processes in parallel), [3] distributed, [4] niching & speciation (segregating population into different species), [5] messy, [6] hybrid (GAs combined with search algorithms), [7] structured, [8] 'genetic engineering', and [9] multi-objective (Takahashi et al., 2011), among many others.

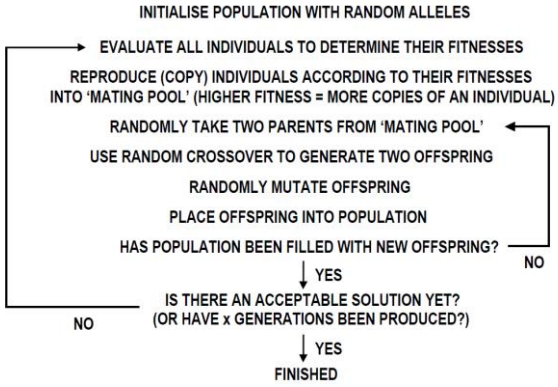



*Figure 100.Example of a genetic algorithm after Bentley (1999).*

GA are heuristic in nature, so within an optimization problem they are designed to find a 'good-enough' solution (Vose, 1999). Nevertheless the GA method presents limitations when compared to other methods (Sivanandam and Deepa, 2007) such as: [1] identification of a proper fitness functions, [2] premature convergence, [3] parameter selection, [4] gradients are not possible, [5] local optimization is complicated, [6] they required a coupled search technique, [7] dynamic data sets are complicated to tackle, [8] the criteria to terminate the process is often not clear, and [9] scalability to deal with more complexity if often complicated as well, among many others others.

**Adaptive genetic algorithms** (AGA) present a variation of AGs where key parameters such as population and mutation change at the same time the algorithm is running allowing changes 'on-the-fly' (Pearson et al., 2012). These variations present the following steps (Sivanandam and Deepa, 2007): [1] initialization, [2] genetic operators (selection, crossover, mutation), [3] local search (iterative), [4] heuristic for adaptive regulation, and [5] stop conditions.

Other related evolutionary computing techniques include **meta-heuristic optimization** techniques based upon search methods in the decision space to find optimal solutions (Bozorg-Haddad et al., 2017). These techniques can be summarized in both trial-and-error and sampling, which could be grid, random, and targeted. Among multiple examples these AGAs are highlighted: [1] ant-colony optimization that uses graphs and artificial ants behaviors as a solver mechanism for optimization (Dorigo et al., 2004), [2] cultural algorithms expanding generic GAs with a domain-specific belief space that conditions the search space as shown in Figure 101 (Reynolds, 2018), [3] memetic algorithms (MA) extending GA with domain-specific local search (individual) capabilities (Neri et al., 2011), [4] stochastic optimization using random search variables such as Monte Carlo (Schneider and Kirkpatrick, 2007), [5] particle swarm optimization (PSO) that iteratively improves convergence candidates (continuous non-linear functions) by using participles, coordinates, and speed within a solution space (Clerc, 2013; Erdogmus, 2018), [6] grammatical evolution (GE) that evolves solutions based on a user-driven grammar (O'Neill and Ryan, 2012), [7] dual-phase evolution (DPE) that promotes self-organization in large scale systems using both graphs and networks (Green et al., 2013). The system goes through multiple phases globally and locally processes affecting connections and components in each

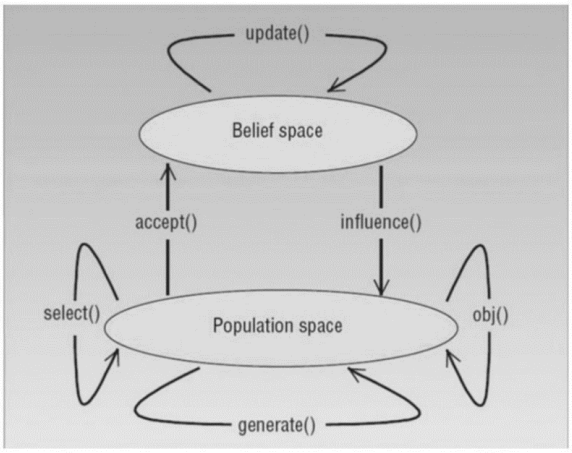



*Figure 101. Cultural algorithms components after Reynolds (2018).*

phase for both natural and artificial systems, and finally [8] differential evolution (DE) that uses differences among individuals as a fast linear operator (Feoktistov, 2007).

On the other hand, **evolutionary programming (EP)** simulates evolution through behavioral relationships instead of a genetic descendance (Fogel and Fogel, 1995). Intelligent behavior is simulated within EP methods through symbols. The machine creates an output symbol, which is the prediction of what the next input symbol will be. Then a payoff function evaluates the quality of the prediction (Dumitrescu et al., 2000). Applications of this technique can be found in medicine, geology, and economics, among many more fields.

**Evolutionary strategies (ES)** or e*volutionstrategie* is an optimization technique created by Bienert, Rechenberg and Schwefel (Bentley, 1999) to tackle hardware systems at first (e.g., pipe bent optimization). In ES methods, there is also no distinction between phenotype and genotype. The child solution is created by randomly mutating parameters of the parent (Rechenberg, 1989). Then it is evaluated by its fitness, leading to another solution until the objective is met. This 1+1 approach presents a problem of stagnation at the local answer (Beyer, 2013) and also a slow convergence to a solution.

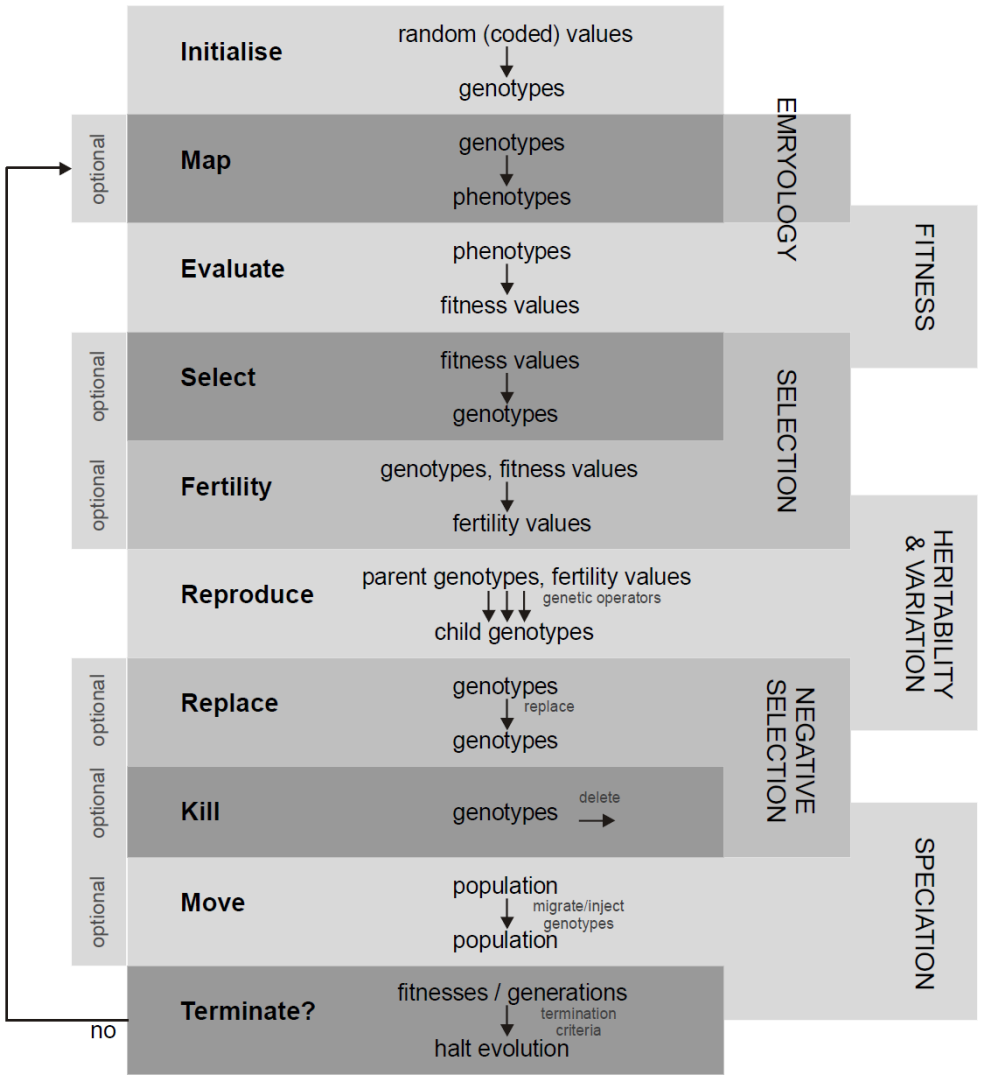



*Figure 102. General architecture of evolutionary algorithms after Bentley (1999).*

### 3.3.2.3. Evolutionary Software Design (Evo3)

The use of evolutionary techniques has influenced computer science and its application like software design. With goals such as efficiency, speed, and complexity several techniques can be highlighted as examples of evolutionary principles that are applied to the development of this field.

**Genetic Programming** (GP) was developed by Koza in the 1990s as a software development technique (see 3.3.2.3). GP is based on the same natural evolution principles as GA, and applies them to find the fittest computer program (Koza, 1994). Figure 103 shows a flowchart representation of the genetic programming. Computer programs are then genetically programmed, and through genetic recombination (crossover) there are mated in a Darwinian process to obtain the fittest solution or its best approximation. Unlike GA, this approach does not make a difference between search space and solution space, so genotypes and phenotypes are the same (Bentley, 1999) altering the solutions directly. The GP method uses hierarchical tree representations to show operations.

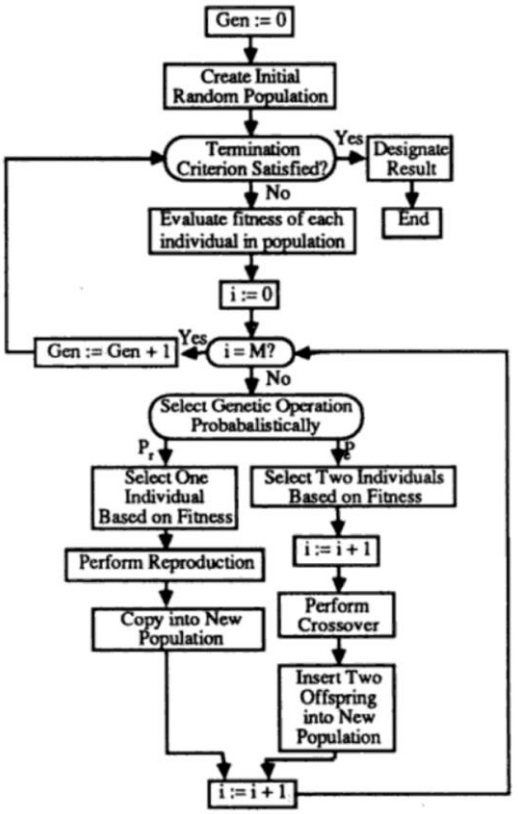



*Figure 103. Flowchart of the GP approach after Koza (1994).*

**Agile techniques** were introduced within SE practices and originally developed for software developments in the beginning of the century. These are based on self-organization principles, bringing speed and flexibility towards changing requirements (Eckstein, 2013). Thus, these methods present an adaptive approach, prioritizing code ("genotype") over documentation or product ("phenotype') and always with a goal-oriented path. These techniques can be applied to large projects (Stober and Hansmann, 2009). Several key characteristics summarize some basic relationship with EC practices such as: [1] agile software projects are meant to grow and evolve constantly until certain size (complexity) makes them not viable any longer, [2] practices are tailored or fitted to specific problems by teams, and [3] the practice is defined by a bottoms-up approach filling gaps within workforce teams and techniques. Among some of the most relevant agile software techniques are the following:

- *Scrum* is an agile practice and software framework development with its origins in product development. It presents a heavy emphasis in both processes and small teams management (Ockerman and Reindl, 2019). Scrum methods are based on: [1] transparency (every team member is aware of all aspects and goal-oriented), [2] inspection (short and frequent meetings – scrums - are used for updates and reviews), and [3] adaptation (fast inspection allows to quickly change strategy, plans, and behaviors in order to achieve goals efficiently and with more quality) (Cohn, 2010; Ockerman and Reindl, 2019). Figure 105 (Mitchell, 2015) shows the scrum cycle using multiple fast sprints.
- *Extreme programming* (XP) is oriented towards quality improvement and customer needs adaptability. This is based on both coding and testing, while integrating customers in the process. It encourages simplicity, feedbacks (system, clients, team), and embraces change (Beck and Andres, 2004). XP presents a process with fast small releases and a continuous integration (teams are always synced) using feedback techniques such as pair programming where two programmers work in the same code simultaneously to reduce errors and increase speed (Zannier et al., 2004).
- *Test driven development* (TDD) is also based on very short redevelopment cycles with requirements being validated through tests cases (Astels, 2003; Beck, 2003). TDD presents a rapid cycle of testing, coding, and refactoring (Shore et al., 2008) that provides proven code every few minutes. Adding features is done in pairs as well and in small increments, reducing defects and improving resilience. Thus, TDD follows an incremental evolutionary approach.
- *Lean software development* (Lean SD) evolved from lean manufacturing and presents a solid framework based upon some key principles such as: [1] eliminate waste, [2] amplify learning (set-based development), [3] decide as late as

possible (concurrent development), [4] deliver as fast as possible (pull systems), [5] empower the team, and [6] a holistic optimization of the whole (Poppendieck and Poppendieck, 2003). Lean SD also involves short iterations and automation (Hibbs et al., 2009) with a continuous approach towards the integration and development of the product.

- *Adaptive software development* (ASD) is a change-driven technique (Highsmith and Highsmith, 2002) that embraces uncertainty within both process and technical ecosystems. This approach intertwines concepts, developments, and management models (Highsmith, 2013) for the development of a complex adaptive system (CAS) presenting high speed, high change, and high uncertainty. ASD sees the project team of an organization as a living organism and applies concepts to it inspired by nature, for a much faster approach than waterfall and other evolutionary engineering paradigms. The adaptability in this method is based upon leadership and collaborations rather than control and command. At the core of this approach is the principle that a 'complex behavior' implies 'simple rules and rich relationships' (Highsmith, 2013). ASD is presented as an iterative design lifecycle (plan, build, and revise) approach, as well as iterative development lifecycle (learn, collaborate, and speculate). ASD has three critical mission artifacts: [1] the project vision (charter), describing objectives, specifications, etc., [2] the project data sheet which is as single page summary used to focus team members, managers, and customers, and finally [3] the product specification outline that presents features, functions, data, and operations, among others high-level product definition documents. This adaptive approach also influences project management by addressing the core question of how to proceed when the solution is not known either partially or completely (Wysocki, 2010). Thus, ASD and adaptive project framework (APF) are centered around the goal but not the solution, so they can accommodate change as a continuous aspect towards developing products and services (Yu et al., 2019). APF manages the 'scope triangle' (time, cost, and resource availability) through multiple cycles until the goal is fulfilled. It is also a client-driven approach that enables and uses continuous increments and questioning.

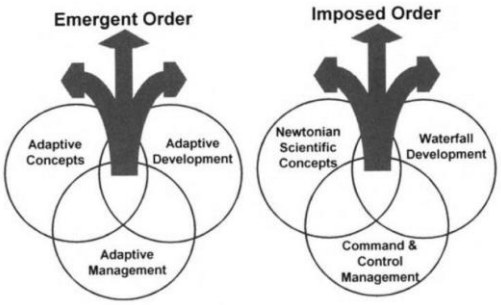



*Figure 104. Differences between adaptive and traditional models, after 2013.*

Finally, **evolutionary software architecture** (Ford et al., 2017) is a set of tools and frameworks to create incremental and guided software developments. This approach presents three phases: [1] incremental change, [2] fitness functions, and [3] appropriate coupling. Like other techniques, the evolutionary approach involves finding a fitted system for an ever-changing environment. This method emphasizes the overall adaptability of the system beyond its components.

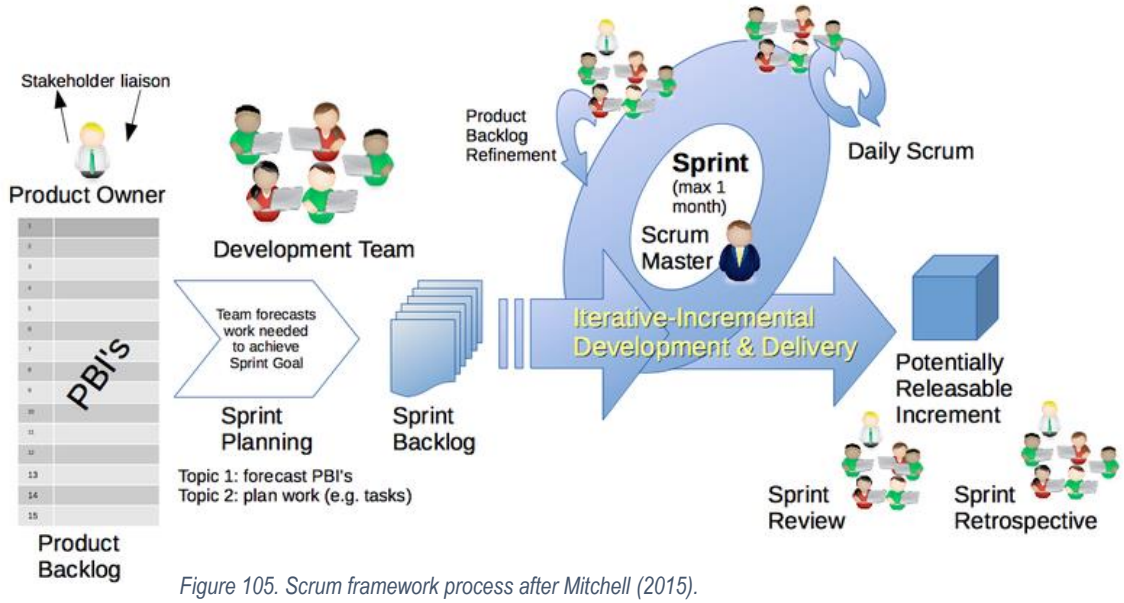



*Figure 105. Scrum framework process after Mitchell (2015).*

#### *3.3.2.4. Evolutionary Systems and Computer Engineering (Evo5)*

The notion of evolutionary-based systems engineering methods was already introduced in section 3.2. However, this section presents details regarding several of these general methodologies, as well as their connection to natural evolution principles. Among some of the most relevant evolutionary systems engineering methods are the following:

- **Evolutionary system architecture** (Jamshidi, 2011) tackles dynamic and not fully formed system of systems (SoS). Within this approach, the evolution of a SoS is developed through mechanisms such as [1] self-evolution, when sub-systems interfaces change, [2] joint evolution, when multiple integrated systems take part on the effort, and [3] emergent evolution, when a new system excesses the capabilities of its subsystems. ES architecture considers several aspects such as [1] business, [2] operations, [3] technology, and [4] interfaces, among others. These are all within structured layers and evolution environments addressing the system complexity (Chen and Han, 2002). This approach follows these steps (Jamshidi, 2011): [1] identify evolution requirements, [2] identify technology options, [3] generate an architecture to fill gaps, and [4] evolve such generated architecture though an architecting process. According to Jamshidi, this approach presents a unique framework to be reinforced by artificial intelligence and other EC techniques including multimethodology analysis capabilities (Grösser, 2012) to obtain better solutions.
- **Incremental and iterative development (IID)** was already presented in section 3.2. Nevertheless, it can be applied to both SE and software development domains (Blokdyk, 2017; Larman and Basili, 2003). As previously stated, IID is based upon multiple iterative cycles that provide incremental and smaller improvements until the goal is fulfilled (Isaias and Issa, 2014). Each cycle serves as feedback input for the next cycle, allowing revisions and improvements at each step. Once more, this approach presents a layer-structured framework taking into account perspectives from team members, customers, and managers (Bittner and Spence, 2006).
- **Evolutionary development (ED)** is common in research and development (R&D) environments (Forsberg, 2020). Each cycle is used as input for the next one and its output can present an unknown nature. This approach includes the user perspective in the development framework and prototypes as way to validate and collect information for the decision process (Hirschheim et al., 1995). This method involves both software and hardware, and implies a collaborative and experimental learning capability using prototypes in the context of ED.
- **Evolutionary System Development Prototyping** (Budde et al., 2012) is an approach based on the evolutionary development of software systems and it involves the production of multiple early working versions as an experimentation source. It was developed by Reinhard Budde, Karlheinz Kautz, and others. This approach brings a rapid engineering standpoint to the process, as well as to other development activities such as project start, information system modeling, software design, software construction, installation, and organizational integration. This approach also includes verification and validation of models through prototyping across the full system lifecycle.
- **Evolutionary System Model (ESM)** is a layered approach for evolutionary complex computation systems, especially for business environments (Henderson, 2012). There are six layers within this approach: [1] technology (e.g., including hardware, networks, operative systems, etc.), [2] domain machine (e.g., computer services), [3] domains (e.g., ontology) as fixed points of an evolutionary system, [4] enterprise (e.g., business practices and organizational structures), [5] process (e.g., individual process and functions), and [6] executive (e.g., human interface).
- **Evolutionary systems engineering (ESE)** presents a systems engineering process in three phases (Hitchins, 2003): [1] basic capabilities including operations and system familiarization, [2] additional capabilities, and [3] delivery. This is especially relevant for lean volume supply productions as well as acquisition and procurement efforts. It presents a very different approach that other systems methods for mass development and production.
- **Evolutionary SoS development** (Rainey and Tolk, 2015) is a systems engineering method where a SoS is never fully formed or complete. Furthermore time, structure, function, and purpose are also developed through evolutionary methods. SoS emerges from existing systems and its evolution implies changes over time (e.g., Arpanet).
- **Feature-driven development (FDD)** is an agile system engineering method made of collections of workflows and techniques that us based on features and roles with an added value such as chief architect (Haberfellner et al., 2019). This method presents a two-weeks fast-paced five steps including: [1] overall model development, [2] feature list considering actions, results, and objectives, [3] feature planning, [4] feature design, and [5] feature construction.
- **Complex adaptive systems (CAS)** is a method based on the notion that complex natural and social systems present

stable states outside an equilibrium (Carmichael and Hadžikadić, 2019) with multiple and correlated feedbacks among all agents within such systems. The CAS is often not addressed at the system-level and requires the study of the agent behavior. Thus, as a SE framework, CAS is an agent-based approach (Miller and Page, 2009) that presents: [1] multiple levels of feedback among agents and components of such complex systems, [2] emergent properties and self-organization, and [3] non-linear dynamic behaviors (Miller and Page, 2009) similarly to those of natural organism. Feedbacks within CAS imply 'that the output of a system at a time *t* has influenced the input of a system at a time *t+1'*. Intelligent complex adaptive systems (Ang and Yin, 2008) applies a multidisciplinary approach towards the simulation of multi-agent systems and organizational studies to understand the behavior of synergetic complex systems. In summary, this approach presents a perspective that looks at a system as a: [1] complex construct due to dynamic behavioral interactions among parts and components, and an [2] adaptable element since a system that can evolve and self-organize itself is based on events within its environment. Applications of this approach are used in economics, agent-based modeling (ABM), strategic management, etc. (Yin and Ang, 2008). Adaptive systems extract and give energy to their environment (Gros, 2015) reflecting the importance within this approach to consider contextual relationships. Thus, CAS systems learn and adapt as they interact (Holland et al., 1992).

#### *3.3.2.5. Evolutionary Hardware Design (Evo4)*

Finally, evolutionary techniques have been proposed and keep being applied to a growing number of hardware design topics and design theories towards physical systems. Among some of the most relevant techniques are the following:

**Evolutionary design** combines evolutionary biology techniques such as biomimetics and comparative studies, computer science (EC), and design (CAD) since the 1990s (Bentley, 1999). Real examples exist in multiple fields, as Figure 111 shows. There are four pillars in this approach:

- *Evolutionary design optimization* uses EC to optimize existing hardware designs making use of parametrics and adaptive CAD (Holland et al., 1992), but it does not generate a brand new design (Kalyanmoy, 2008). Related approaches involve concept design, detail design, evaluation, and iterative redesign (Bentley and Wakefield, 1996). See Figure 108a for a visual example.
- *Creative evolutionary design* is based upon the creation of evolutionary designs starting from scratch using two perspectives:
  - *Conceptual evolutionary design* provides a high-level design framework with simpler representations showing basic building blocks (Figure 108b) for the system. In this phase, basic genetic algorithms are used to assess system phenotypes.
  - *Generative evolutionary design* (genetic design) uses computer methods based upon GA to create 3D representations of phenotypes. As Figure 108c shows, this approach goes from unshaped components to detailed and valid designs based on genotype generations and fitting evaluations of the system.
- *Evolutionary art* involves the generation of images using EC (e.g., digital trees, vegetation, moving crowds, etc.) with great success in the fields of animation, design, etc. (McCormack, 2008).
- *Evolutionary artificial life form* is a method based upon the creation of artificial behaviors, problem solving approaches, and communication strategies, among other tools using EC techniques. In essence, this has the objective to simulate natural and complex behaviors virtually within a digital environment and in combination

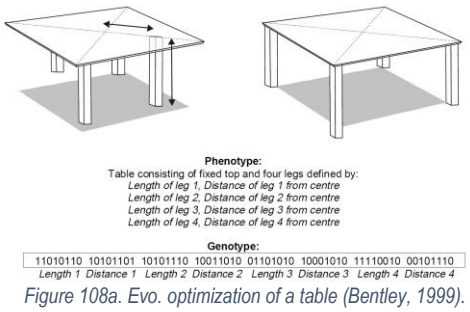



*Figure 108a. Evo. optimization of a table (Bentley, 1999).*

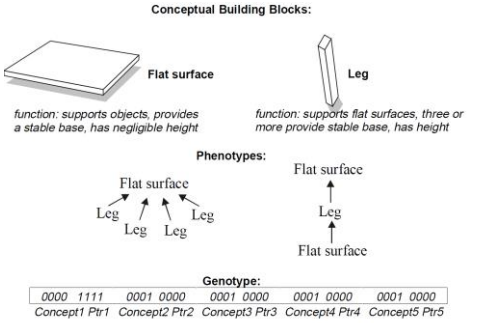



*Figure 108b. Conceptual evo. design (Bentley, 1999).*

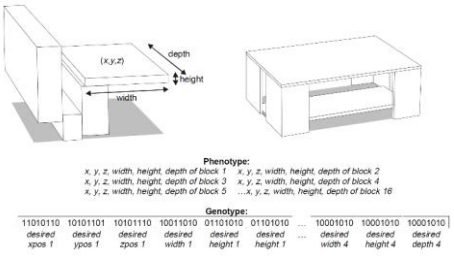



*Figure 108c. Generative evolutive design of a table (Bentley, 1999).*

of a series of fitness functions to validate multiple complex facets.

- **Induction design** is a method of evolutionary design applied to architecture and urban development (Watanabe, 2002). This approach reinforces the need of diversity towards system design resilience.

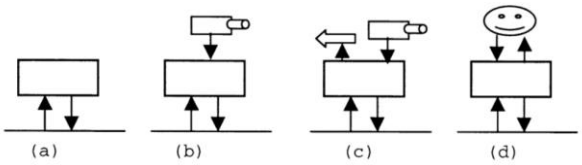



*Figure 109. Different agents after Shen (2019).*

- **Multi-agent system** (Shen, 2019) tackles multiple aspects of systems implementation from an agent-based standpoint. This integrates collaborative, concurrent, planning, and other advanced manufacturing techniques, such as holonic manufacturing across engineering fields. Agent based models (ABMs) are computer models simulating the interaction between agents with design consequences to the overarching system. Ants are a good natural example of the inspiration behind agent-based models (Wilensky and Rand, 2015). Multiagent systems bring a similar approach considering intelligent agents (Weiss, 1999). Holonic manufacturing systems or HMS (Gräßler and Pöhler, 2017) within this multi-agent approach use "autonomous and cooperative building blocks of manufacturing systems to transform, transport, store, and validate physical objects" (Shen, 2019). The agents within this method include specialists, sensors, action agents (e.g., robots), and interactive agents with humans (e.g., assistants). Figure 109 shows a general agent diagram. These agents process information (messages), make decisions, execute, and record information transactions as part of the methodology. Thus, this method [1] studies learning mechanisms among agents, [2] establishes architectures accordingly to optimize their implementation, [3] tackles other phases and challenges such as coordination, concurrency, allocation, conflict resolution, and knowledge ontologies (Monostori et al., 2006).

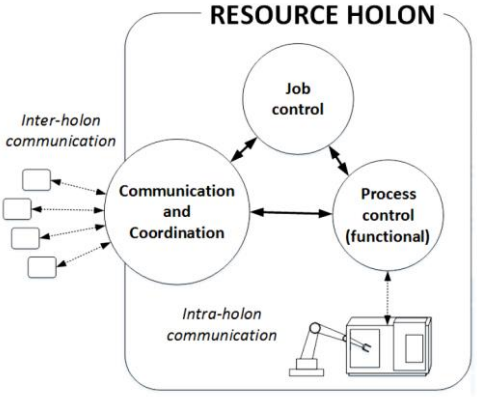



*Figure 110. Holon after Gräßler, et al. (2017).*

- **Evolutionary robotics** (ER) uses EC to develop and optimize both hardware designs and controllers for autonomous robotics (Nolfi et al., 2000). Under this method, the control system of a robot becomes an artificial chromosome, so EC techniques are used to improve its fitness. This applies as well to the body morphology (structure), sensors, motor properties, and design layouts (Vargas et al., 2014). ER is based on the fitness of the robot behavior which is generated from genomes. Here genetic operators are used like in other EC techniques to select and produce the next system generation. This method is applied to hardware and robotic systems allowing to explore new and unconventional designs based on large numbers of variables, tune parameters, design optimizations, and system robustness through its fitness (Vargas et al., 2014). Besides the fitness-based evolution, another approach within ER is a novelty-based evolution (Silva et al., 2016). Such approach is an evolution based on behavioral diversity, that avoids early convergence issues and enables an open-ended system evolution addressing both hardware and system behavior (Evans and Back, 2011) simultaneously.

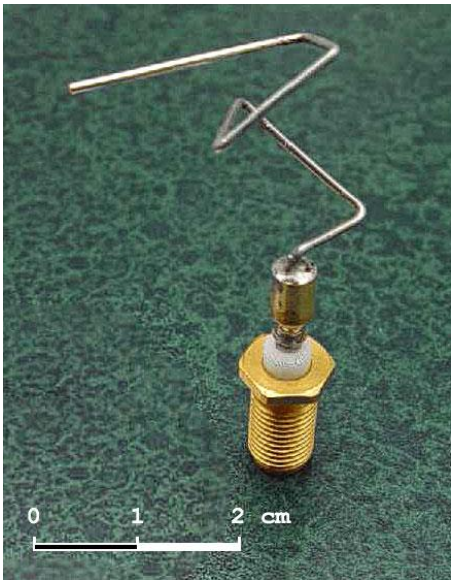



*Figure 111. ST5 antenna designed using GA techniques (NASA, 2006 - public domain).*

- **Evolutionary machine design** (Nedjah and Mourelle, 2005b) exemplifies a growing trend addressing how evolvable hardware and genetic programming can be combined to improve and optimize both hardware designs and behavioral capabilities of hardware-based systems. Applications of this can be found in adaptable circuits (Koza

et al., 2005), neural network controlled robots (Bekey and Goldberg, 2012), and SOA hardware systems across fields.

### *3.3.2.6. Evolutionary Principles and Models Review Matrix*

Table 17 presents a summarized review of evolutionary key principles (EVP) across different fields of study. A brief description and basic characteristics regarding their approach and use are also provided, while an evaluation of critical aspects related to the design of hardware-based complex system architectures is considered based upon these points:

- **Foundation**. This is a summary description of the evolutionary principle (EVP) and its characteristics. The following prefixes describe the category as [1] theory (TH), [2] principle (PRI), [3] technique (TEC), [4] model (MOD), [5] method (MET), and [6] mechanism (MECH).
- **Main function or task.** Is the EVP approach concentrated on analysis (ANSY), design (DES), implementation (IMP), or development (DVP)?
- **System design phase.** What phases are addressed by this approach? Basic design phases are numbered as it follows: [1] planning, [2] problem study, [3] concept design, [4] embodiment design, [5] detailed design, [6] analysis, [7] optimization, [8] testing and validation, [9] documentation, [10] implementation, [11] delivery, [12] marketing, [13] operations, [14] decommission, [15] recycling of products and processes (Seider et al., 2016) (Haik et al., 2010). See Figure 45 to identify color codes and structure level.
- **Information type.** Does the EVP tackle geometrical information (GEO), such as volumes, shapes, sections, tolerances, and other graphical constructs)? Can it handle abstract information (ABS) such as analytical or genetic parameters? Can it handle interfaces (INT)?
- **Qualitative / quantitative (Qt./Ql.)**. Does the EVP tackle qualification and quantification parameters?
- **Scope**. Can the SE method handle point-design solutions (PDS), families of point-design solutions (FDPS), development process (DEV), continuous designs (CONT), or a combination (COMB) of all of them?
- **Adaptability**. Does the EVP present an approach that is considered flexible (FLE), networked (NET), strict linear (LI), iterative (ITE), waterfall (WA), spiral (SPI) or natural (NAT) methodologies (see Figure 46)?
- **Perspective.** Is the EVP based upon a discrete disciplinary standpoint (DD) or a synergetic multidisciplinary approach (SA)? This question studies again whether it is based on a 'divide-and-conquer' approach discretizing disciplines and subsystems, or on the hand it can tackle multidisciplinary perspectives all at once.
- **Optimization**. Does the approach allow any optimization of the system, solution, or process?
- **Tool platform**. What type of tool and technique does the EVP enable or support? This could include: [1] computer models, [2] drawings, [3] CAD, [4] graphs, [5] physical prototypes, [6] documents, [7] schedules, and [8] math models.
- **Reference**. This is a summary list of relevant technical references and professional practice inputs reviewed during this research.

| EVP | Foundation / *Application* | Function | Phase | Geo. | Qt./Ql. | Scope | Adapt. | Pers. | Opt. | Tools | References |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Evo1 - Biological Evolution*** | | | | | | | | | | | |
| **Driving Principles** | | | | | | | | | | | |
| **Natural Selection** | *MECH*. It is based on the survival of the fittest phenotype principle. This mechanism increases system complexity and provides an adaption approach against Env. changes. *Application: genetic algorithms* | DES IMP | 1-15 | ABS GEO | QT QL | FPDS | NAT | SA | Yes | N/A | (Bell, 1996) (Herron and Freeman, 2013) (Zeigler, 2014) (Forrest, 1993) (Koza, 1994) |
| **Evo-Devo Deep Homology and Pleiotropy** | *MECH*. The development of an individual (ontogeny) produces changes at the species level (phylogeny). Deep homology describes how certain genes (hox genes) control the development of an individual as well as across species. The development of a natural system affects species and control mechanisms of its development including multiple subsystems. *Application: software and hardware* | ANSY DES IMP | 1,5-7,10 | ABS GEO | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Arthur, 2002) (Hall, 2012b) (Hall and Olson, 2003) (Richards et al., 2012) (Miglani, 2010) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *computer optimization.* | | | | | | | | | | |
| **Speciation** | *MECH.* Species could evolve into other species over time (speciation) becoming the common ancestor (descent). Its mechanisms include: [1] geographical separation, [2] small population entering an isolated niche, [3] entering a new connected niche, [4] and a population going through genetic changes. The creation of diversity is part of the evolutionary process.<br>*Application: software*. | ANSY DES IMP | 1,5-7,10 | ABS GEO | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Dieckmann et al., 2004)<br>(Butlin et al., 2009)<br>(Cooper, 2005) |
| **Mutations** | *MECH.* Mutations are changes in DNA molecules leading to phenotype changes and the genetic variation of species. Smaller variations on system components lead to more complexity and efficiency in the environmental context over time.<br>*Application: software testing, structural optimization, genetic algorithms, etc.* | ANSY DES IMP | 1,5-7,10 | ABS GEO | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Nei, 2013)<br>techniques (King and Offutt, 1991)<br>(Burns, 2002)<br>(Srinivas and Patnaik, 1994) |
| **Coevolution** | *MECH.* Two or more (guild) species affect each other evolution through natural selection. Related paths include predator-prey, host-parasite, or mutualism. Thus, unrelated systems can help, support, and thrive upon each other.<br>*Applications: computer science, cosmology and astronomy, manufacturing, architecture, management, sociology, technology.* | ANSY DES IMP | 1,5-7,10 | ABS GEO | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Thompson, 2009)<br>(Thompson, 2005)<br>(Durham, 1991)<br>(Potter and Jong, 2000)<br>(Ho, 2004)<br>(Tolio et al., 2010)<br>(Mazzoleni, 2013)<br>(Allen et al., 2011)<br>(Durham, 1991)<br>(Lee, 2020) |
| **Adaptation** | *PRIN.* The dynamic evolutionary process leads for an organism to become fitter to its environment and potentially evolve as species if phenotypic changes become hereditary. Adaptation and biological fitness are related. A higher level of evolutionary adaptability requires a higher level of self-organization among components and their interactions. These happen because of [1] new situations, [2] new functionality needs, [3] a modified behavior, and [4] an optimization of system parameters.<br>*Applications: robotics, architecture, artificial intelligence, machine leaning, etc.* | ANSY DES IMP | 2,4-7,10 | ABS | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Werf et al., 2008)<br>(Levi and Kernbach, 2010)<br>(Košir, 2019)<br>(Holland et al., 1992) |
| **Evolvability** | *PRIN.* It is the capacity for adaptive evolution of a developmental system. This is related to the robustness of an organism, and the persistence of certain traits under perturbations. This is a critical aspect to understand how complicated systems can withstand a lot of change.<br>*Application: robotics, systems architecture, design, etc.* | ANSY IMP | 2,6, 7,10 | ABS | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Minelli, 2018)<br>(Wagner, 2013)<br>(Long et al., 2018)<br>(America et al., 2010) |
| **Self-organization** | *PRIN.* Self-organization is the principle by which natural systems become structured through their own internal processes, and without external commands. Self-organization mechanisms include: [1] positive and negative feedback, [2] exploitation and exploration, [3] multiple interactions, and [4] energy versus entropy or disorder. A key aspect is the access to information (e.g., interactions).<br>*Application: biology, chemistry, systems control, cybernetics, thermodynamics,* | ANSY IMP | 2,6, 7,10 | ABS | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Yates, 2012)<br>(Schweisguth and Corson, 2019)<br>(Camazine et al., 2003)<br>(Ashby, 1991)<br>(Zhang, 2015)<br>(Whitesides et al., 1991)<br>(Gershenson, 2007)<br>(Nagarajan and Ruckenstein, 1991) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *computer science, socioeconomics, linguistics, etc.* | | | | | | | | | | (Winfree, 2006) (Witt, 1997) (Zhang & Park, 2008) (Vijver et al., 2013a) |
| **Evolutionary Ecology** | *MOD*. It studies the variation (genotype differences) within individuals and among individuals, populations, and species. It considers the effects of the physical environment. It is based upon the scale of time, complexity, size, space, and interactions between species, populations, and individuals. *Application: biology, social sciences, information processing, technology, urbanism, etc.* | ANSY IMP | 2,6,7,10 | ABS | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Fox et al., 2001) (Pianka, 2011) (Smith, 2017) (Dukas, 1998) (Solée et al., 2013) (Rivkin et al., 2019) |
| **Biological Network** | PRIN. Biological systems at multiple scales are captured by network representations using nodes for different biological entities (e.g., cells, molecules, organism). This approach looks at complex systems as networked elements with relationships among parts. *Application: biosciences, genetics, ecology...* | ANSY DES IMP | 1,5-7,10 | ABS GEO | QT QL | FPDS DEV | NAT | SA | Yes | N/A | (Kepes, 2007) (Marin-Sanguino et al., 2019) (Proulx et al., 2005) (Boccaletti, 2010) (Guzzi and Roy, 2020) (Freeman, 1977) |
| ***Evo2 - Computer Science*** | | | | | | | | | | | |
| **Methods** | | Function | Phase | Geo. | Qt./Ql. | Scope | Adapt. | Pers. | Opt. | Tools | |
| **Genetic Algorithms (GA)** | *MET*. Genetic algorithms are a metaheuristic process. A preliminary algorithmic population is genetically created, then through a fitness function a parent group is selected to breed a new descendant population based on multiple randomized processes until the solution is approximated enough. Basic steps are: [1] generation of initial population, [2] seed selection, [3] next generation by crossover (recombination of parental genetic information) and mutation (changing algorithm chromosomes). Types of GA are: [1] steady state, [2] parallel, [3] distributed, [4] niching & speciation, [5] messy, [6] hybrid, [7] structured, [8] genetic engineering, [9] multi-objective, etc. GA limitations include proper fitness function, premature convergence, parameter selection, gradients, local optimization, and scalability, among others. | ANSY DES | 2-4, 6,7 | ABS | QT | FPDS DEV | NET | DD | Yes | Model | (Baeck et al., 2018) (Dumitrescu et al., 2000) (Holland, 1962) (Bentley, 1999) (Takahashi et al., 2011) (Vose, 1999) (Sivanandam and Deepa, 2007) |
| **Adaptive Genetic Algorithms (AGA)** | *MET*. Adaptive genetic algorithms present a variation of AG where key parameters (e.g., population, mutation, etc.) change at the same time the algorithm is running, which allows changes 'on-the-fly'. Key steps are: [1] initialization, [2] genetic operators (selection, crossover, mutation), [3] local search (iterative), [4] heuristic for adaptive regulation, [5] stop conditions. | ANSY DES | 2-4, 6,7 | ABS | QT | FPDS DEV | NET | DD SA | Yes | Model | (Pearson et al., 2012) (Sivanandam and Deepa, 2007) |
| **Evolutionary Programming (EP)** | *MET*. Evolutionary programming (EP) simulates evolution through behavioral relationships instead of genetic descendance. Intelligent behavior is simulated within EP through symbols. Payoff functions evaluates the prediction. *Applications*: medicine, geology, and | ANSY DES | 2-4, 6,7 | ABS | QT | FPDS DEV | NET | DD | Yes | Model | (Dumitrescu et al., 2000) (Fogel and Fogel, 1995) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | economics, among many others | | | | | | | | | | |
| **Evolutionary Strategies (ES)** | *MET*. It is a system optimization with no distinction between phenotype and genotype. Child solutions are created by randomly mutating parental parameters, evaluating by its fitness, and enabling the next child solution until objectives are met. | ANSY DES | 2-4, 6,7 | ABS | QT | FPDS DEV | NET | DD | Yes | Model | (Bentley, 1999) (Rechenberg, 1989) (Beyer, 2013) |
| ***Evo3 - Software Design*** | | | | | | | | | | | |
| **Techniques** | | Function | Phase | Geo. | Qt./Ql. | Scope | Adapt. | Pers. | Opt. | Tools | |
| **Genetic Programming (GP)** | *MET*. It is based on the same natural evolution principles as GA. Computer program are genetically programmed, and through genetic recombination (crossover) they are mated in a Darwinian process to obtain the fittest solution (or approximation). This approach does not make a difference between search space and solution space, so genotypes and phenotypes are the same. | ANSY DES | 2-4,6,7 | ABS | QT | FPDS DEV | NET | DD | Yes | Model Math | (Koza, 1994) (Bentley, 1999) |
| **Scrum** | *TEC*. It is an agile practice and software framework development done in multiple iterative fast-paced cycles. Its principles are transparency, inspection, and adaptation. | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | PDS DEV | NET | DD | Yes | Model Math | (Ockerman and Reindl, 2019) (Cohn, 2010) |
| **Extreme Programming (XP)** | *TEC*. It is oriented towards quality improvement and adaptability to customer needs based on both coding and testing, while integrating customers in the process. XP uses small releases, continuous integration (team members are always sync) and feedback methods (pair programming). | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | PDS DEV | NET | DD | Yes | Model Math | (Beck and Andres, 2004) (Zannier et al., 2004) |
| **Test Driven Development (TDD)** | *TEC*. It is based on very short redevelopment cycles, with requirements being validated through tests cases. TDD presents a rapid cycle of testing, coding, and refactoring. Adding features is done in pairs as well and in small increments. | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | FPDS DEV | NET | DD | Yes | Model Math | (Astels, 2003) (Shore et al., 2008) |
| **Lean Software Development (Lean SD)** | *TEC*. It is a framework based upon: [1] eliminate waste, [2] amplify learning (set-based development), [3] decide as late as possible (concurrent development), [4] deliver as fast as possible (pull systems), [5] empower the team, and [6] holistic optimization of the whole. It uses continuous short iterations and automation. | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | FPDS DEV | NET | SA | Yes | Model Math | (Poppendieck and Poppendieck, 2003) (Hibbs et al., 2009) |
| **Adaptive Software Development (ASD)** | *TEC*. It is a change-driven technique that intertwines concept, development, and management models for the development of complex adaptive systems (CAS). ASD presents an iterative life cycle (plan, build, revise) and an iterative development life cycle (learn, collaborate, speculate). It has three key artifacts: [1] project vision, [2] project data sheet, [3] product specification. The adaptive project framework (APF) within it manages the 'scope triangle' (time, cost, and resource availability) through multiple client-driven cycles until the goal is fulfilled. | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | FPDS DEV | NET | DD | Yes | Model Math | (Highsmith and Highsmith, 2002) (Koza et al., 2005) (Wysocki, 2010) (Yu et al., 2019) |
| **Evolutionary Software Architectures** | *TEC*. It is a set of tools and frameworks to create incremental software developments. It has 3 phases [1] incremental change, [2] fitness functions, and [3] appropriate coupling. This involves finding a fitted system for an ever-changing environment, | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | FPDS DEV | NET | SA | Yes | Model Math | (Ford et al., 2017) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | emphasizing the system adaptability. | | | | | | | | | | |
| ***Evo 4 - Systems Engineering*** | | | | | | | | | | | |
| **Techniques** | | Function | Phase | Geo. | Qt./Ql. | Scope | Adapt. | Pers. | Opt. | Tools | |
| **Evolutionary System Architecture (ES Arch)** | *TEC*. It tackles dynamic and not fully formed system of system (SoS) in three ways: [1] self-evolution (sub-systems interfaces change), [2] joint evolution (multiple integrated systems take part on it), and [3] emergent evolution (the new system excess the capabilities of its subsystems). Key steps are: [1] identified evolution requirements, [2] identify technology options, [3] generate architecture to fill the gaps, [4] evolutionary architecting process based on generated architecture. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD | (Jamshidi, 2011) (Chen & Han, 2002) (Grösser, 2012) |
| **Incremental Iterative Development (IID)** | *TEC*. It is applied to both SE and software development. It is based upon multiple iterative cycle, providing incremental and smaller improvements until the goal is fulfilled. Each cycle serves as feedback input for the next cycle, allowing for revisions and improvements at each step. It is layer-structured framework considering perspectives from team members, customers, and management. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS | QT | FPDS DEV | NET | SA | Yes | Model | (Blokdyk, 2017) (Larman and Basili, 2003) (Isaias & Issa, 2014) (Bittner and Spence, 2006) |
| **Evolutionary System Development Prototyping (ESDP)** | *TEC*. It is based on evolutionary software systems development producing multiple early working versions. It uses a rapid engineering process including verification, validation, and prototyping. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Budde et al., 2012) |
| **Evolutionary System Model (ESM)** | *TEC*. It is a layered approach for evolutionary complex computation systems, used in business environments. It presents six layers such as [1] technology, [2] domain machine (computer services), [3] domain (ontology), [4] enterprise (business practices and organizational structures), [5] process (individual process and functions), and [6] executive (human interface). | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Henderson, 2012) |
| **Evolutionary Systems Engineering (ESE)** | *TEC*. ESE presents 3 phases: [1] basic capabilities (operations and system familiarization), [2] additional capabilities, and [3] delivery. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Hitchins, 2003) |
| **Evolutionary SoS Development** | *TEC*. It presents a SE approach where a SoS is never complete, so time, structure, function, and purpose are developed through evolutionary methods. SoS emerges from existing systems and their evolution implies changes over time. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Rainey and Tolk, 2015) |
| **Feature-driven Development (FDD)** | *TEC*. It is an agile SE method with five fast-paced steps done in two weeks: [1] overall model development, [2] feature list considering actions, results, and objectives, [3] feature planning, [4] feature design, and [5] feature construction. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Haberfellner et al., 2019) |
| **Complex Adaptive Systems (CAS)** | *TEC*. CAS is based on the notion that complex natural and social systems present stable states outside an equilibrium, with multiple and correlated feedbacks among agents within such systems. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Carmichael and Hadžikadić, 2019) (Miller & Page, 2009) (Yin and Ang, 2008) (Gros, 2015) |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | (Holland et al., 1992) |
| **Evolutionary Development (ED)** | *TEC*. In ED each cycle is used as input for the next one, and the output is unknown. It includes a user perspective in the development framework, and it uses prototypes to validate and collect information in the decision process. It involves software and hardware and involves a collaborative and experimental learning approach using prototypes across the lifecycle. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Forsberg, 2020) |
| ***Evo 5 - Hardware Design*** | | | | | | | | | | | |
| **Methods & Theories** | | Function | Phase | Geo. | Qt./Ql. | Scope | Adapt. | Pers. | Opt. | Tools | |
| **Evolutionary Design (ED)** | *TH*. ED combines evolutionary biology, computer science (EC), and design (CAD). It presents four areas such as [1] design optimization, [2] creative evolutionary design, and [3] evolutionary art, and [4] evolutionary artificial life forms. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Bentley, 1999) (Kalyanmoy, 2008) (Bentley and Wakefield, 1996) (McCormack, 2008) |
| **Induction Design** | *MET*. This is a method of evolutionary design applied to architecture and urban development. This approach reinforces the need of diversity for design resilience. | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | FPDS CONT | NET | SA | Yes | Model Docu. CAD | (Watanabe, 2002) |
| **Multi-Agent System (ABM)** | *MOD*. It tackles multiple aspects of a system implementation from an agent-based standpoint such as collaborative work, concurrent, planning, advanced manufacturing, holonic manufacturing across engineering fields. Agent-based models (ABMs) are computer models simulating interactions among agents including specialists, sensors, action agents (e.g., robots), and interactive agents for humans (e.g., assistants). These agents process information (messages), make decisions, execute, and record information transactions. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Shen, 2019) (Wilensky and Rand, 2015) (Weiss, 1999) (Gräßler and Pöhler, 2017) (Shen, 2019) (Monostori et al., 2006) |
| **Evolutionary Robotics (ER)** | *MET*. ER uses EC to develop and optimize controllers and hardware for autonomous robotics (body morphology, sensor, motor properties, and layout). It is based on [1] the fitness of the robot behavior (genomes) and [2] genetic operators to select and breed the next generation. This method allows to: [1] explore unconventional designs, [2] tune parameters, [3] optimize designs, and [4] improve system robustness through its fitness. ER presents both fitness-based and novelty-based evolution. | ANSY DES IMP INT | 2-4, 6, 7-11 | ABS GEO | QT | PDS FPDS DEV CONT | NET | SA | Yes | Model Docu. CAD Proto. | (Vargas et al., 2014) (Nolfi et al., 2000) (Silva et al., 2016) (Evans and Back, 2011) |
| **Evolutionary Machine Design** | *MET*. It combines evolvable hardware and genetic programming to improve and optimize both hardware design and behavioral capabilities of hardware-based systems. *Applications: adaptable circuits, neural network-controlled robots, etc.* | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | FPDS DEV | NET | SA | Yes | N/A | (Nedjah and Mourelle, 2005b) (Koza et al., 2005) (Bekey and Goldberg, 2012) |
| **Agile Hardware** | This involves agile software techniques, MBSE methods, and rapid prototyping to delay design freezes. | ANSY DES IMP | 2-4, 6, 7-11 | ABS | QT | FPDS | NET | SA | Yes | N/A | (Huang et al., 2012) |

*Table 17. Evolutionary concepts, methods, and techniques across multiple fields.*

### 3.3.3. Evolutionary Methods Conclusions and Gaps

Across all these multiple topics and methods that evolve from the study of natural systems, there are some overarching principles and critical gaps relevant to this research. These gaps and conclusions regarding evolutionary domains are summarized in Table 18, including key foundational aspects towards evolutive systems design methods.

| | Family | Conclusion Points | Gaps | Foundation Points |
|---|---|---|---|---|
| **Evo1** | **Natural Evolution Process** | • Natural evolution allows continuously to increase complexity and validate solutions against multiple environmental changes.<br>• Each former generation (parental heritage) is a validated entry for a new generation. However the new generation is and can be very different.<br>• Evo-devo. The development process of a new organism (system) is as critical as its genotype (design). Such process affect organs (subsystems) in the short term and also the whole species (product family) in the long-term.<br>• Diversity is key for survival.<br>• Co-evolution. Evolution is complex and involves relationships among species, individuals, and multiple environmental responses as well.<br>• Evolution leads to adaptation through self-organization in response to its environment context and conditions.<br>• Evolutive changes bring robustness to the individual, but they need to be heritable and successful to improve a species permanently (evolvability).<br>• Self-organization works at a multidisciplinary and multifaceted level.<br>• Ecology-level interactions (individuals, species, population, and environment) affect natural evolution processes.<br>• Networks provide a model to study and represent such complex relationships, mechanisms, and systems. | • The application of evolutionary principles to other technical fields has been mainly monodisciplinary and limited in the scope of its process. For instance, addressing directly only numeric parameters when using evolutionary-based methods is a good example.<br>• Natural evolution involves both phenotype, which can be understood as hardware, and genotype that can also be understood as software. However, not many techniques and methods apply methodologies embracing both simultaneously. | • This mechanism presents a multi-complex nature.<br>• The management of complexity is a secondary effect of this mechanism.<br>• The refinement of the solution is done through multiple small and self-directed variations.<br>• Both the product (system or organism) and the process are adaptable in nature when following such natural mechanism. |
| **Evo2** | **Evolutionary Computer Science** | • Evolutionary algorithms in general and genetic algorithms-driven techniques in particular have been very successful approaches towards system design, optimization, and efficiency.<br>• EC techniques are often inspired by hardware or natural systems, but they tackle mostly genotype or data-driven aspects of the system. | • These methods only tackle software and data-driven systems, but not hardware.<br>• Evolutionary computational techniques do not handle geometry directly. They rather | • Optimization of both system and process is done synergistically and simultaneously. |

| | | | | |
|---|---|---|---|---|
| | | • EC techniques enable both optimization and complexity management. | address only abstract analytical data.<br>• These methods apply across all design and system development lifecycle phases. | |
| **Evo3** | **Evolutionary Software Design** | • These techniques are an applied subset of evolutionary computing.<br>• Agile techniques are inspired by EC methods. They use iterative methodologies for faster, smaller, and more adaptable design cycles.<br>• These methods do not tackle necessarily the type of coding but the process and workflow by which software programs are created more efficient, faster, and with better-quality.<br>• These methods embrace constant change under a fast-paced and highly adaptable approach. | • These techniques do not handle well non-analytical data.<br>• These are mainly software and data driven methods.<br>• There is no explicit reference to the use of technical heritage solutions. | • Methods within this category tackle the design workflow as part of their approach. |
| **Evo4** | **Evolutionary Systems Engineering** | • These methods address the evolution of complex system of systems (SoS) considering the full lifecycle process that includes all technology options.<br>• Prototyping is a key tool and method within this process.<br>• Models include multiple aspects regarding operations, roles, domains, validation, and verification towards the implementation of the system.<br>• These approaches address both design and development of complex adaptive systems (CAS). | • These do not present hardware-based considerations.<br>• Non-geometrical processes are not addressed by these techniques. | • Multiple lifecycle phases could be addressed by these methods.<br>• Prototyping becomes a key tool within many of these approaches. |
| **Evo5** | **Evolutionary Hardware Design** | • These methods present a path to optimize hardware-based designs based on EC principles and algorithms.<br>• They tackle geometrical information considering multiple scales, interfaces, and configurations.<br>• Agent-based models are based upon key interactions within this approach.<br>• Machine learning and robotic designs have embraced these techniques.<br>• These methodologies present a more applied approach towards physical design and systems implementation. | • Lifecycle phases are considered across methods and multiple specialized areas.<br>• Multidisciplinarity is characteristic of these methods with limited application.<br>• These tackle non-complex assemblies.<br>• There is a lack of a full and integrated method and theory. | • Geometrical information is handled by addressing interfaces and basic shapes. |

*Table 18. Summary of conclusions and gaps regarding evolutionary theories and methods.*

## 3.4. Overall Literature Review Gaps and Conclusions

The study of techniques and state-of-the-art practices in the fields of design engineering (DE) and systems engineering (SE) theories leads to several overarching conclusions that are summarized in this section. Furthermore, this literature review of general evolutionary principles and applications also provides several key conclusive and foundational points for this research directly aiming at the domain of hardware-based system architecture designs.

**Design engineering** theories and methodologies studied in section 3.1 present a general 'divide-and-conquer' approach. This implies that subsystems and especially disciplines, tend to be tackled independently from each other following a sequential aggregation process across the design lifecycle. Thus, disciplines are in essence tackled one after the next one and sometimes partially in parallel. On the other hand, very solid design methodologies such as prescriptive design embrace both analysis and synthesis from a scientific standpoint presenting in general two tendencies. They can be more on the [1] highly creative side (e.g., innovative design) with huge capabilities towards addressing complex problems but less powerful than mote detailed design techniques, or [2] they could be more rigid such as method-driven techniques (e.g., axiomatic) but with a powerful foundation towards computer-driven workflows making them less capable towards more innovative solutions with no heritage. Similarly, tools and workflows described in section 3.1 present both a foundation and a practice approach around those two opposed tendencies. Computer developments and data-driven techniques have made possible to bring analysis and design simultaneously into the design and development process, however this is still done mainly from a parametric and facilitated standpoint. In essence, design engineering in the 21st century presents a gap towards reconciliating the characteristic multidisciplinary synergy of simpler (and older) techniques such as those created in the dawn of architecture practice, with new analytical and process-driven capabilities that are nowadays enhanced by artificial intelligence and machine learning techniques. Thus, adaptability is the missing link between those apparently opposed paths.

Nevertheless, complex large systems had increasingly required tackling non-geometrical aspects since the 1950s. **Systems engineering** of complex challenges became a third and very solid branch in addition to those design approaches previously described. A close study of the literature review and leading practices conducted in section 3.2 has shown that a very fast development of this field in the last half century has led to multiple approaches, theories, workflows, and techniques. In general, from document-based beginnings to current state-of-the-art Model-based techniques, SE has looked at the system as an abstract construct (model), but there have been challenges regarding how to bring geometry into the SE process. In general, most of these reviewed techniques present a rigid methodology (partially paired with DE methods) except for iterative (IID), OPM language, and skeleton methodologies that exercise a more continuous system design approach. Furthermore, SE methods do not tend to fully recognize the full lifecycle of complex systems today ignoring often phases such as recycling, repurposing, and decommission to name a few. However, there is not a clear integration among theories and their practice. In parallel, the workflow associated to these practices tends to be rigid as well presenting not much synergy among disciplines that are tackled by SE activities or with other design processes as exposed in section 3.2.

The flexibility of workflows and methods when dealing with complex and highly adaptable system is the common denominator between design and systems engineering sides within a joint effort. This has been the starting point for previous evolutionary techniques addressed in section 3.3. Nature-inspired techniques dealing with complex design while increasing system efficiency and quality have been developed mainly in the computer science area (e.g., genetic algorithms) with some applications towards systems engineering as well.

**Evolutionary computational methods** (EC) are very capable and often fast-paced techniques for system optimization in data-driven processes. Although, these present relevant gaps towards a full methodology capable of creating hardware-based complex systems. Some of these techniques such as evolutionary design techniques and robotic application present effective approaches manage and integrate geometrical information as well. The literature review reveals that they are quantifiable evolutionary methods, but there is gap when it comes to methods capable of qualifiable evolutionary workflows using a multidisciplinary standpoint with implementation and development purposes.

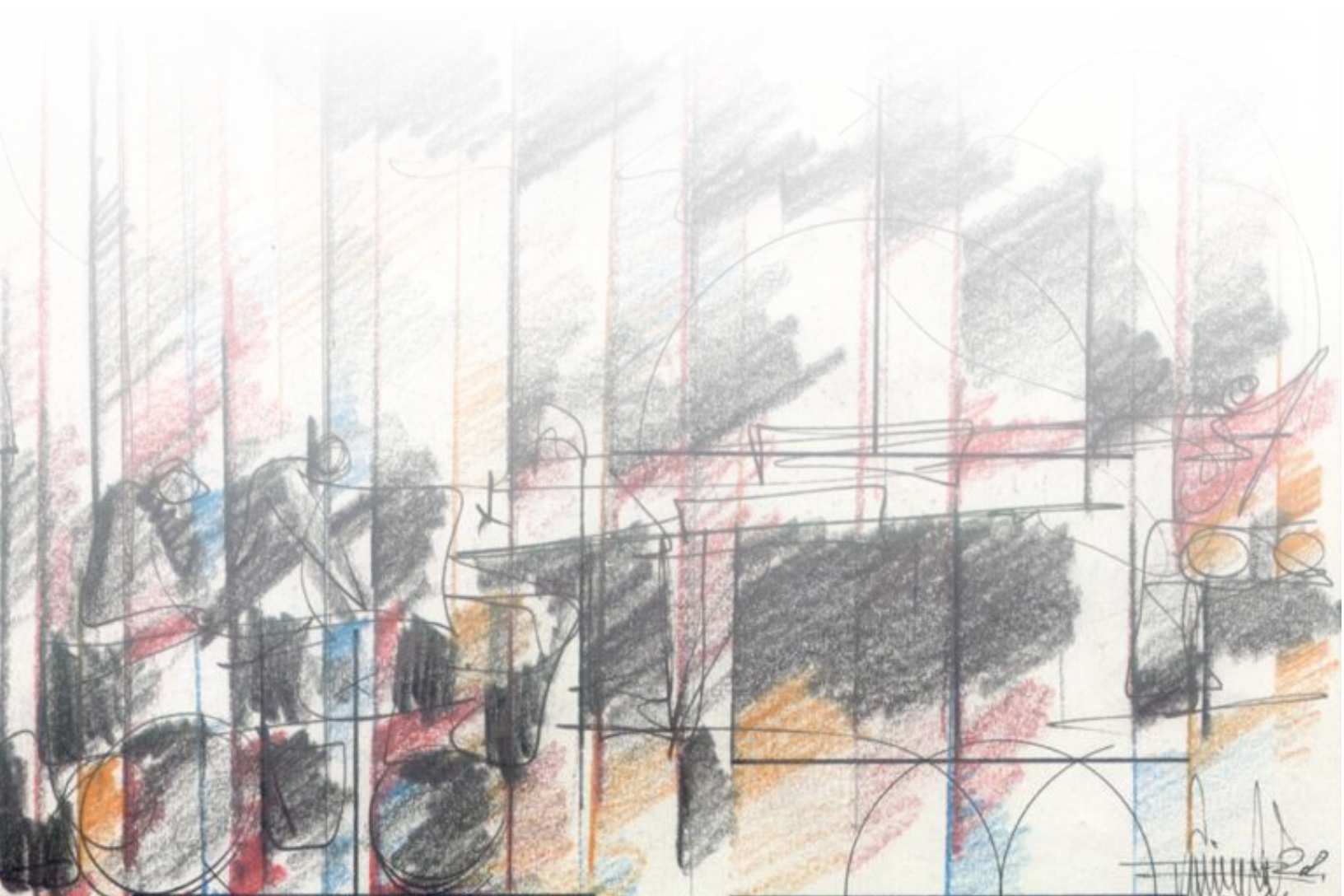
Finally, the study of basic natural principles behind natural-evolution-driven EC brings the **inspiration towards mechanisms capable of embracing continuity, flexibility, and heritage not only from a data or information perspective, but also from a hardware design standpoint.** Section 3.3 presents the potential and importance of embracing of development processes in the creation of a complex and efficient system (e.g., evo-devo).

# EVOLUTIVE ARCHITECTURES

## Literature Review

## CHAPTER 4

*"It is not the strongest of the species that survives,*
*nor the most intelligent,*
*but the one most responsive to change."*
***Charles Darwin, 1809***



# 4. Evolutive System Architectures

The context presented in section 2 defines a series of key characteristics affecting the present and future practice of both design engineering and systems engineering disciplines. Beyond such practice, system architectures as solutions are also influenced by the consequences of scarcity, agility, complexity, and heritage. This is especially critical towards the design of high-performance hardware-based systems due to the inherent system complexity (CHS), the level of performance often associated with them, as well as their multidisciplinary nature. Thus, this research is focused on two areas:

- **Evolutive system architectures** (eSAR) as a class of systems within such contexts.
- **Evolutive system architecture design** (eSARD) as a practical design methodology to model such architectures.

This evolutive design approach can certainly be applied to any system design architecture development, independently from the field of application, and it is presented as a theoretical framework. The overall objective of this is to increase system performance and efficiency beyond any existing heritage, while using an agile and system-level perspective.

The following section tackles several definitory subjects such as: [1] overall evolutive approach (section 4.1), [2] key evolutive system characteristics (section 4.2), [3] design constraints and drivers (section 4.3), [4] eSAR definition and methodology (section 4.4), [5] complexity as evolutive integration (section 0), and [6] overall conclusion (section 4.6).

## 4.1. Evolutive Approach: Inspired by Evolution and Driven by Adaptability

If general context stressors define both practice and systems outcomes, to determine the characteristics of an evolutive system it is necessary to study an approach capable of handling such influences. Those characteristics will lead to a final definition of eSARs, and subsequently to the foundation for a design methodology towards their development (chapter 5).

Any complex system, for instance an organism, is also defined by its environment. Such context, as we have seen in previous section is critical. However, such context not only affects the system itself, but also the process used to develop it. Such context for the practice of system design engineering combines both design engineering and systems engineering approaches among other perspectives when dealing with highly complex and/or large systems. Section 2 provided such context for this research and highlighted a series of key stressors that any complex system endeavor should consider from both the product and the process standpoints. Hence, these are the most relevant stressors:

- **Complexity** could be understood as the number of parts, features, and relationships among subcomponents or system behaviors. Figure 13 shows how the efficiency of a complex system could be understood as the number of resources required to enable all functions required for its completion. The origin of that complexity includes environmental, multidisciplinarity, and other sources affecting product and processes from design to implementation.
- **Heritage** brings pressure, influence, and opportunity in risk assessment, decision making process, and design principles behind any complex system design and implementation. This stressor opens a door towards mechanisms capable of making the most with previous versions, solutions, or technologies across technical fields.
- **Cultural Disruption.** New methodologies (Figure 16) and solutions do stress established methods in any given organization. They affect risk perception and influence outcomes based on environmental and cultural changes.
- **Performance**. Similarly, the need for higher performance is driven by previous points, affecting the efficiency and technical capability of a system, as well the development methodology required for its implementation.
- **Interconnection**. Complex system architectures are not only are becoming smarter nowadays but also more connected among their components and systems, as well as with their environments and frameworks of operation.
- **Innovation**. The constant need or drive towards new solutions and methods ripples through both design and development processes. This inckudes multiple cultural postures such as conservative, incremental, and radical.
- **Scarcity**. External factors such as climate, population growth, economy, competitiveness, $CO_2$ emissions, and energy availability become stressors in the development of a complex system, driving the necessity of serving more needs with less resources (Figure 7). The scarcity factor depends on both the lack of resources available as well as the uncertainty given by practices used in the process. As a result, there is a growing trend for systems to use less resources (frugal) and processes to be more adaptable so they can compensate for any uncertainty.

- **Multidisciplinarity**. One of the consequences of the inherent complexity of some systems is the need to tackle multiple disciplinary standpoints to provide feasible, reliable, competitive, and efficient solutions. However, this can also be a source of efficiency for the system, and efficiency of the process.
- **Agility** is related to the speed and leanness in the use of resources towards addressing constraints brought by previous points, as well as the system flexibility increasingly needed due to environmental changes. This is key in any present and future system design engineering efforts in highly competitive environments.

On the other hand, if a future system needs to address such environmental and contextual topics, the process of designing, optimizing, and implementing such a system should do it too. Consequently, the literature review in section 3 detailed three areas such as [1] design engineering, since a system need to be envisioned, [2] systems engineering since a system needs to be defined and described, and [3] evolutionary principles as a potential pool of techniques and inspirations towards tackling such constraints. While sections 3.1.4, 3.2.5, 3.3.3, and 3.4 provided conclusions with regards to the review, there are some key overarching gaps and areas of interest across domains, fields, and techniques worth highlighting.

- **Synergy**. Often design and systems engineering methods tackle challenges from a multidisciplinary standpoint at a high level, and from a discreet disciplinary standpoint at a lower or more detailed level.
- **Continuity**. System design methodologies tend to be driven by point-design or discreet solutions. These often do not allow fast, easy, or flexible changes to the process or the system solution.
- **Qualification**. While the quantification of parameters is widely distributed across techniques and fields, tackling non-quantifiable parameters, variables, and topics is certainly more complex and less developed. This is even more relevant when it is key for the process to handle both quantifiable and qualifiable aspects.
- **Geometry**. Systems engineering methodologies do not tackle geometrical information well, especially when handling large complex systems. They rely on partial unidirectional 'bridges' to CAD and BIM models for instance. Furthermore, systems design processes often struggle to pair complex geometries to complex system definitions or models.
- **Full cycle**. Key stressors in the context of these practices require having a fully integrated set across the complete lifecycle of a system. This includes and reinforces both initial design and final recycling or decommission phases.
- **Flexibility**. Design and systems engineering methodologies need to provide an increasing level of flexibility especially when dealing with design and context stressors. This is critical in complex systems regardless of the field.
- **Disruption** relates to the capability of the process and the needs of the solution in bringing new designs and concepts into fruition with a high-level of detail, and a remarkable differentiation with other heritage solutions at any level.
- **Fast paced**. Currently, agility and resource leanness are key traits in a system design process, not only as a response to the previous context stressors, but also because of the business and finance constraints such as time-to-market.
- **Connectivity.** Complex systems are becoming more networked in nature across subsystems, components, users, and environments. Thus, design techniques need to respond to such increasing core characteristics.

All these points are in essence interconnected from multiple perspectives, thus a feasible response towards tackling them completely requires a holistic and overarching approach. The evolutive outlook developed on this thesis addresses such response, and it has a compound nature based on both evolutionary principles and adaptive concepts.

### 4.1.1. Evolutionary Approach

The literature review in section 3.3 highlighted and summarized some of the most basic mechanisms and principles behind the implementation and development of natural and artificial evolutionary systems. Such processes include natural selection, self-organization, co-evolution, adaptation, optimization, speciation, evo-devo, eco-evo-devo, mutation, agile methods, and genetic computational techniques, among others. The natural selection process finds and develops fitted solutions for each context where a system or organism exists. Each successful solution (parent) in such an environment, provides the building code to a better solution (progeny), which will be better adapted to such changing environment. Evolution is therefore nature's solution to deal with change and therefore entropy.

Evolution is multidimensional because it works at a physical level, through the phenotype of the organism (hardware), which has been created from successfully transmitted genotypes (software code) at the information level in such process. It is also multidisciplinary as multiple and often highly complex factors such as environmental, biological, and even cultural among others affect the survival and adaptation capabilities of the organism. Thus, a self-organization principle is at the core

of such process, leading a complex system to gradually find its optimum state of equilibrium among the multiple influential design factors, environmental stressors, and opposing design forces.

Within any evolutionary process there is not just a final and singular solution, as each system is part of a species that is in constant evolution to deal with such environmental changes and stressors. Therefore, adaptation is a consequence along with the shared co-evolution of species, to reduce these effects and increase the efficiency (effort-energy ratio) as a response to the environment. Thus, both natural and artificially evolved systems, can optimize in this way their responses towards changing conditions. It is important to highlight that in evolutionary processes previous solutions or generations could be considered as heritage and are indeed validated solutions that serve as a foundation for the next generation. Hence, heritage is it not a limiting factor, but rather a solid foundation towards new and possibly disruptive solutions.

A key part of this natural process is the development of the system itself or, in other words, how different components should grow and become integrated from the early beginning of the organism (evo-devo). Such evolutionary development also considers other ecological aspects (eco-evo-devo) becoming a critical phase in the development of any complex system. When this is translated into creating artificial or man-made systems, this applies to the ideation, prototyping, and early-stage development phases. These phases are often disconnected from the end results in many design methodologies, unlike the case with natural systems where they become the initial instruction for the creation and development of any organism. Furthermore, complex systems present a networked attribute defined by the interactions between components, subsystems (organs), environment, users, and even other types of systems (species). Connectivity is therefore both a key characteristic and a strength that implements and substantiates self-organization and optimization principles within such systems.

Natural evolution addresses and manages complexity, making system optimization simple, efficient, and agile. In the 1990s early genetic algorithmic practices (3.3.2.2) were influenced by this mechanism and it is the inspiration for this research towards hardware design, implementation, and optimization. Key aspects of an evolutionary approach (overall figures of merit) and the evolution mechanism relating to the previously mentioned stressors and are summarized in Table 19.

| EVOLUTIONARY APPROACH - PRINCIPLES | | | | | |
|---|---|---|---|---|---|
| | **Evolutionary Principle** | **Description** | **Context Stressors** | **Figures of Merit** | **Natural Evolution Tool** |
| E1 | **Continuous** | *Solutions are always under continuous development, so any point design becomes an instantiation.* | Complexity | Functions | Natural selection |
| E2 | **Multidimensional Multidisciplinary** | *Hardware, software, and algorithms are foundational aspects for both systems and processes. Interrelated disciplines are combined and validated.* | Complexity Multidisciplinary | Functions Smartness | Phenotype and genotype Self-organization |
| E3 | **Agile** | *Solutions are obtained fast and with less resources.* | Agility | Resources | Self-organization |
| E4 | **Adaptable** | *System capability to adapt to changing requirements, constraints, and needs, as well as its associated cost.* | Scarcity | Functions Resources | Evolvability Adaption |
| E5 | **Evolvable** | *It is the ratio between resources and functions integrated within systems and processes that are used to obtain optimized solutions continuously.* | Performance | Resources | Mutation |
| E6 | **Networked** | *Systems are defined by interactions between components, subsystems (organs), environment, users, and even others system types (species).* | Interconnection | Functions Smartness | Self-organization Evo-Devo Bio. Network |
| E7 | **Heritage-driven** | *Advancements or slow-downs due to previous successful solutions and cultural traits.* | Heritage Innovation | Functions | Speciation Natural selection |
| E8 | **Environment-driven** | *Context characteristics influence, constrain, and foster integrated processes, products, and services.* | Scarcity Culture Disrupt. | Resources | Co-evolution |
| E9 | **Development-driven** | *Early-stage design, prototyping, and growth phases influence the final system output and design process.* | Innovation | Smartness | Eco-evo-devo |

*Table 19. Key foundational characteristics of evolutionary processes.*

### 4.1.2. Adaptive Approach

On the other hand, the complexity of the context described in section 2 highlights one key characteristic needed above all in the practice and implementation of systems design engineering, *adaptability*. This concept connects all general stressors, while encompassing both needs and gaps in the practice of design and systems engineering as identified in sections 3.1 and 3.2. These include continuity, synergy, qualification, geometry, full cycle method, and finally flexibility. Furthermore, this concept also links needs among disciplines, practices, techniques, and complex systems themselves with independence of the field of application. To this end, tools certainly condition how we can approach a problem.

The concept of adaptability can be traced back to those gaps though the design and implementation of an *adaptive system design*. This is a way to build upon some key characteristics of a complex system, link gaps between design and system engineering processes, and tackle previous stressors. In essence, the question the is how we look at the design process if adaptability becomes the main objective for both products and processes.

An adaptive design approach looks at the system from a continuous standpoint, so new designs or new requirements could be tackled easily 'on-the-fly', with a process that is not discreet. Multiple methods participate from this perspective: [1] process-driven design engineering techniques (DE10, Table 11), [2] some lifecycle-based SE (SE2-2, Table 15) such as concurrent, agile, skeleton (Badiru, 2019), IID (INCOSE, 2015), OPM (Dori, 2016), [3] cross-cutting SE methods (e.g., MBSE). All these also present an approach allowing and enabling the use of models, parametrics (Kimura, 2001), and constructs to this purpose (sections 3.1 and 3.2). Similarly, SE frameworks (SE5, Table 15) such as Harmony SE (Ramos et al., 2012), RUP (Valacich et al., 2017), and BIM (Smith and Tardif, 2009), among others also present similar capabilities.

Adaptability is also about synergy, or in other words multidisciplinarity, involving the combination of multiple disciplines to address feasibility, performance, and complexity often simultaneously. Similarly, some SOA techniques handle adaptability mostly through analytical parameters that can shared or combined such as SE cross-cutting methodologies (SE2-3, Table 15) and process-driven DE techniques (DE10, Table 12). Relevant to the second these are very relevant: axiomatic (Farid and Suh, 2016), MPM (Chakrabarti and Blessing, 2014), Set-based (Singer et al., 2009), FORFLOW (Rodenacker, 2013), and MPM. Furthermore, integrative DE approaches (e.g., Generative - Keane, 2018) combine such parameters with geometry, while evolutionary DE (Braha et al., 2007) addresses system optimization from an adaptable and multidisciplinary standpoint. Nevertheless, an adaptive approach needs to tackle this from both analytical and geometrical standpoints. Hence, this is a gap in most design engineering techniques, as well as systems engineering techniques where both geometrical and analytical information play a key part on the process for complex hardware-based systems (Section 3). The influence of geometry is certainly foundational in DE techniques as section 3.1 presented, but it becomes a gap in SE approaches. On the other hand, frameworks such as BIM combine both with great success and space to grow.

At the same time, an adaptive solution should consider both quantitative and qualitative aspects of system design. While quantitative analytical parameters are easier to compute, qualifiable aspect are difficult to measure and thus to be computed, shared, and compared. However, some of these parameters are key in certain hardware-based systems which are heavily influenced by aesthetics, user experience, or environment interaction among other complex characteristics. This specifically means that an adaptive design process needs to be able to tune solutions based on complex relationships brought by both qualifiable and quantifiable characteristic. In essence, the design process needs to handle complexity, which can be understood as multiple internal and external relationships of different nature. Thus, for such process to embrace adaptability the capability to handle, update, upgrade, and change connections is key. This is very important especially when these connections are not always considered when approaching complex systems. Often those relationships, which present a networked nature, only become evident when the design process dives deeply and broadly (full cycle) enough. Furthermore, they set connections among components (e.g., subsystems), behaviors, and values (e.g., cultural) to name a few, and they are driven by 'qual-quant' principles. Innovative DE (DE8, Table 12) due to their broad perspective and evolutionary DSE methods that can handle complexity by addressing large numbers of key relationships (Evo4, Table 17) have an adaptive nature.

But adaptability is also about considering all different steps in the lifecycle of system development, otherwise the possibility of handling changes from both requirement and implementation sides becomes very limited. An adaptive approach implies bringing connections between those phases early in the process, and across all phases. This has a foundation with

lifecycle-based SE methods (SE2.2, Table 15) such as TDSE (Buede, 2009), Spiral (Liu, 2015) ICSM, and DEJI (Badiru, 2019), among others. These tackle development challenges from a phase perspective as well, becoming models for workflows, strategies, etc. This highlights that within such processes, adaptivity is also about efficiency and therefore speed. So, an adaptive method promptly tackles with promptness complex architectures based upon all previous characteristics. In a way adaptability is about making the most with all available resources (including time), as well as aiming for a better system performance. Time is one of these key resources especially when considering design global stressors. Furthermore, an adaptive approach needs to deal with uncertainty, thus it must be flexible towards requirements, resources, results, and changing relationships. DE techniques such innovative methods (DE8, Table 12) present a high level of flexibility in both processes and results, while evolutionary SE and hardware design (HD) also handle flexibility during design phases.

Innovation is also a critical aspect since the *unknown* is one of the most difficult aspects to handle by a design approach. When heritage solutions are not really a proper starting point for a design process, due to the novelty of the design, the need for systems and implementation performance, as well as flexibility for a new system architecture becomes critical from technical, management, and business reasons. Innovative DE are a good and broad foundation (DE8, Table 12).

Overall, an adaptive design approach allows embracing constant changes, such as changing design requirements and environmental conditions. Indirectly, this also brings efficiency, agility, and resilience to the design effort. Adaptability is strongly coupled to evolution, and it is the essence of how to do better with less, answering to constraints that are driven by scarcity or complexity. Table 20 presents a summary of the adaptive approach though foundation axioms and its relationship with general context and design stressors, figures of merit, and mechanism derived from state-of-the-art techniques.

| ADAPTIVE APPROACH - AXIOMS | | | | | |
|---|---|---|---|---|---|
| | **Foundational Axiom** | **Description** | **Methodology Gaps** | **Figures of Merit** | **Related DE / SE Techniques** |
| A1 | **End-to-end** | *Method covers the full cycle including ideation, development, implementation, and recycling.* | **Full cycle** | Resources | Lifecycle-based SE |
| A2 | **Multidisciplinary** | *Multiple parameters, discipline perspectives, and relationships are tackled simultaneously.* | **Sinergy** | Functions<br>Smartness | Process-driven DE<br>Cross-cutting SE |
| A3 | **Promptness** | *The more adaptability a process or system presents, the more reactivity and thus speed brings towards any changes across the lifecycle.* | **Fast-Pace** | Functions<br>Resources | Innovative DE<br>Evolutionary SE<br>Evolutionary HD |
| A4 | **'Qual-quant'** | *Both analytical quantifiable parameters and qualifiable aspects are considered simultaneously.* | **Qualification** | Smartness | Innovative DE<br>Evolutionary DE<br>Evolutionary SE |
| A5 | **Geometry-driven** | *Beyond analytical design parameters describing the system, geometry is created, managed, and assessed across the process.* | **Geometry** | Functions | SE Frameworks<br>Descriptive DE |
| A6 | **Network** | *Adaptability is about relationships among components (e.g., subsystems), behaviors, and values (e.g., cultural) within complex systems driven by 'qual-quant' principles.* | **Connectivity** | Functions<br>Smartness | Innovative DE<br>Evolutionary DE<br>Evolutionary SE |
| A7 | **Continuous** | *Design efforts are continuous so modifications, detailed descriptions, and variations could happen effortlessly 'on-the-fly' as part of the approach.* | **Continuity** | Functions | Process-driven DE<br>Integrative DE<br>Lifecycle SE<br>Cross-cutting SE |
| A8 | **Adaptable** | *Changes in requirements, implementation, design needs, and available resources can happen without restarting the design and considering heritage.* | **Flexibility** | Resources<br>Smartness | Innovative DE<br>Evolutionary SE<br>Evolutionary HD |
| A9 | **Innovation-driven** | *Systems design processes are open to infuse new and disruptive approaches that could be validated and reinforced by heritage solutions.* | **Disruption** | Functions<br>Smartness<br>Resources | Innovative DE |

*Table 20. Key foundational characteristics of an adaptive design approach.*

### 4.1.3. Evolutive = Evolutionary + Adaptive

The concept of evolutive process and system is developed within the context of this research similarly to how the term is created, as a contraction or composition between evolutionary and adaptive principles.

Such an evolutionary approach behind natural and artificial systems addresses adaptability, change, and complexity with proven, powerful, and potentially 'simple' mechanisms. Furthermore, it effectively tackles all global stressors described in section 2 affecting both the design and implementation of complex systems, through a combination of evolutionary principles (Table 19). In essence, the evolutionary approach represents the nature of a system and its development process. On the other hand, an adaptive approach addresses through a series of axioms (Table 20) many gaps in both design and systems engineering methodologies (section 3). Thus, critical goals such as improvements in systems performance, process efficiency, and agile workflows become the foundation of this new design approach.

The evolutive workflow and perspective integrates evolutionary principles and applies adaptive axioms towards a design and systems engineering methodology (DSE) that develops full or partial evolutionary hardware-based system architectures (section 1.7.1). Furthermore, this approach also complements and fills gaps across state-of-the-art SE and DE techniques while also being used to infuse key evolutive principles into any new system design efforts. This approach is inclusive by nature, and presents a foundation based on adaptability to design upon. Therefore, upcoming sections will present characteristics and fundaments of an evolutive architecture (section 4.2) as the ultimate objective, the associated system design methodology (section 5), as well as an example of such an approach (section 6).

However, combining adaptability and evolution presents deeper consequences in the process and conceptualization of such systems. Their inherent complexity intertwines both processes (designs) and outcomes (products) very closely. Thus, before addressing all key system characteristics and subsequent methodologies it is critical to define fundamental keystones that are created by such contractions and connections. The combined evolutive approach is based on three keystones:

- **Adaptability**. An evolutive architecture system is adaptive in nature from both geometrical and analytical standpoints, in response to changes driven by requirements, design, and implementation. Thus, a system design is understood as an instantiation within a continuous design process, rather than a final static solution. Such design is always conceived with potential changes in mind, allowing variations in subsystems, manufacturing techniques, and materials, among other environmental or conceptual changes. Techniques, designs, concepts, and materials among others 'genetic' inputs of the future system include both quantifiable and qualifiable variables. Evolutive thinking then works with the same principles behind them and enables not only disruptive methods to be infused into the process but also easier alternative solutions or selections across the full lifecycle. Therefore, adaptability brings a broader spectrum of functions, and the more functions with less components a system architecture achieves, the more efficient it will be. Within such process, heritage solutions are validated as parent inputs for an evolutionary process. This keystone addressed these three context stressors with the following characteristics (in italic):
  - **Complexity.** Both qualifiable and quantifiable variables are used on a continuous system design workflow, and are integrated within a geometry-inclusive system design framework.
  - **Heritage.** Heritage solutions are non-limiting foundations for adaptive solutions across the full systems lifecycle.
  - **Cultural disruption.** New methods can be infused into an evolutive design workflow, enabling subsequent feasible alternatives in response to environment or design changes.
  - ***Process:*** *This point relates especially to geometrical aspects of the design process.*
  - ***Functionality*** *(measurement): Number of functions per geometry for a given set of interactions and resources.*
  - ***Range*** *(Figure 128, Y): Unadaptable (less functions, more parts) to evolutive (more functions, less components)*
- **Reactivity**. Complex and simple solutions can present different levels of interaction with the environment, as well as interaction capabilities as a function of their inherent smartness. An evolutive approach addresses the adaptability of system also from its interactive potential. The more capability of the system to interact, the higher success it can have against environmental changes, increasing its evolvability. This also leads to less resistance towards new solutions, since the systems can be more easily upgraded. At the same time, complex and interactive systems, such as organisms have a higher level of networked interconnections between subsystems and their behavioral functions. Natural evolutionary methods are based on deep connections between subsystems as well as with internal

development processes. This connection can be traced back to the very genetic information used to drive the creation of a phenotype, which also presents complex multidisciplinary interconnections. Thus, reactivity encompasses and responds to three general design stressors with the following characteristics (in italic):

- **Performance.** The less resources a system uses in response to design requirements or environment (context) changes the more efficient the system would be.
- **Interconnection.** Subsystems and behaviors present networked connections among them within evolutive systems. This key stressor becomes a strong advantage regardless the level of complexity.
- **Innovation.** Evolutive solutions and processes facilitate new and disruptive solutions to be integrated, and a networked approach ensures feasible solutions are always possible backups. Similarly, heritage inputs are a validated baseline towards developing, infusing, and advancing new solutions.
- ***Process:*** *This point relates especially to behavioral aspects of the design process.*
- ***Interaction*** *(measurement): Number of reactions per system behavior, for a given set of functions and resources.*
- ***Range*** *(Figure 128, Z): Passive (less interaction, less smartness) to reactive (more interaction, more smartness)*

**Regeneration**. The utilization of resources either for the design process, or by the system itself during its lifetime is key in an evolutive system and its design approach. This keystone is highly related to the other two, and directly responds to remaining global design stressors with these characteristics (in italic):

- **Scarcity.** An evolutive system should adapt to both the need for optimization and the lack of resources from a continuous perspective. The relationship between the system and its environment will change over its lifecycle and operational cycles. This is a foundational part of a system architecture that considers different phases.
- **Multidisciplinarity.** Complex systems depend on a multidisciplinary perspective to achieve higher levels of performance, through design, analysis, operational optimization, or all of them at once.
- **Agility.** The evolutive methodology is designed to provide promptness into the design process as well as into further modifications or generations of a system architecture.
- ***Process:*** *This point relates especially to substance and material aspects of the design process.*
- ***Resource utilization*** *(measurement): Resource utilization across design process and system operations.*
- ***Range*** *(Figure 128, X): Depleting (only consuming resources) to regenerative (replenish resources).*

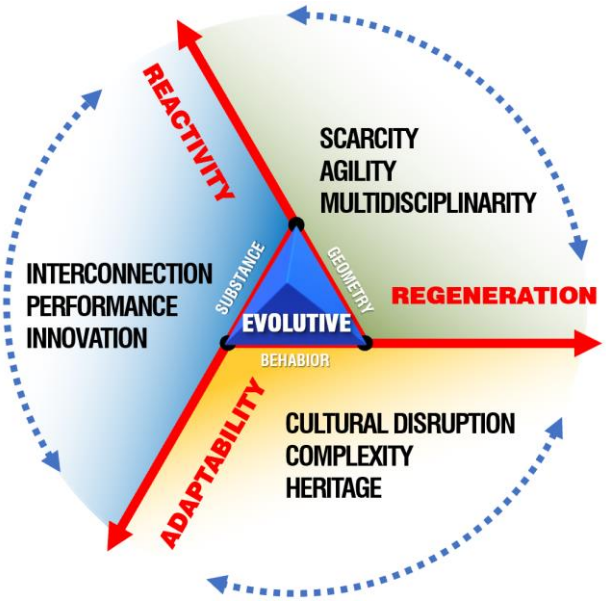



*Figure 112. Evolutive tetrahedron of system design.*

These three keystones are interconnected and present the foundational scheme for an evolutive design methodology, described by the **evolutive tetrahedron** (Figure 112). Furthermore Table 21 presents correlations between general stressors, methodology gaps, evolutionary principles, adaptive axioms, and finally these evolutive keystones.

| EVOLUTIVE APPROACH | | | | | | |
|---|---|---|---|---|---|---|
| | Context Stressors | Challenge Needs And Responses | DE / SE Gaps | Evolutionary Principle | Adaptive Axiom | Evolutive Keystone |
| Ev1 | Complexity | This is the capability to manage large amounts of components, features, relationships (quantifiable and qualifiable), subcomponents, and behaviors that can change over time. | Geometry<br>Qualification<br>Sinergy | Continuous (E1) | 'Qual-quant' (A4)<br>Geometry-driven (A5) | Adaptability<br>Reactivity |
| Ev2 | Heritage | This is the influence of past proven solutions in risk assessment, decision making process, and design features behind complex systems, as well as related design, implementation, and operations processes | Continuity<br>Sinergy | Heritage-driven (E7) | End-to-end (A1) | Adaptability |
| Ev3 | Cultural Disruption | It is related to the easiness to infuse new methodologies and approaches that stress and disrupt established cultures and design inertia towards new system designs and workflows. | Full cycle<br>Sinergy<br>Continuity | Environment-driven (E8) | Adaptable (A8) | Adaptability<br>Reactivity |
| Ev4 | Performance | It is about a better ratio between required resources, and system functions that are being served across the system lifecycle. This applies to both system architectures and development methodologies. | Full cycle<br>Sinergy | Evolvable (E5) | Adaptable (A8) | Reactivity<br>Regeneration<br>Adaptability |
| Ev5 | Interconnection | Complex system architectures are becoming smarter and more connected among subcomponents, other systems, environments, and frameworks of operations. | Network | Networked (E6) | End-to-end (A1)<br>Network (A6) | Reactivity<br>Adaptability |
| Ev6 | Innovation | The constant need or drive towards new solutions and methods ripples through both design and development processes. There are multiple cultural postures such as conservative, incremental, and radical. | Disruption<br>Sinergy | Development-driven (E9) | Innovation-driven (A9) | Reactivity<br>Adaptability |
| Ev7 | Scarcity | This is the capability to continuously address the availability, uncertainty, and variability of all resources required for a feasible system architecture, across the full system lifecycle and from every perspective. | Continuity<br>Sinergy<br>Full cycle | Adaptable (E4) | End-to-end (A1)<br>Continuous (A7) | Regeneration<br>Adaptability |
| Ev8 | Multidisciplinarity | This concept reflects the simultaneous capability to tackle both multiple and discrete disciplinary standpoints providing feasible, reliable, competitive, and efficient system architectures and subsequent design processes. | Sinergy<br>Geometry<br>Qualification | Multidimensional<br>Multidisciplinary (E2) | End-to-end (A1)<br>Multidisciplinary (A2)<br>'Qual-quant' (A4) | Regeneration<br>Adaptability<br>Reactivity |
| Ev9 | Agility | Finally, this relates to the speed and leanness in the use of resources addressing flexible constraints due to changes in context, requirements, or design parameters. | Fast Pace<br>Flexibility<br>Sinergy<br>Continuity | Agile (E3) | Promptness (A3) | Regeneration<br>Adaptability |

*Table 21. Correlations between general stressors, methodology gaps, evolutionary principles, adaptive axioms, and evolutive keystones.*

## 4.2. Evolutive System Keystones

### 4.2.1. Adaptability

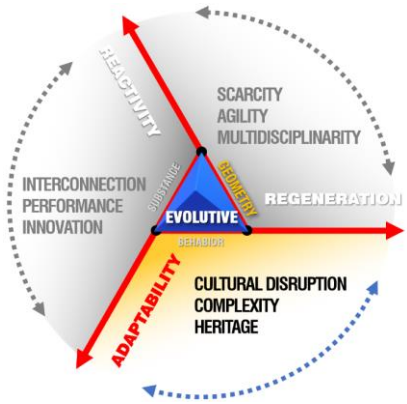



*Figure 113. Adaptability within the evolutive tetrahedron of system architecture design.*

Adaptability is a keystone of the evolutive tetrahedron that highlights geometry aspects and design activity (Figure 113). As it was previously presented, this evolutive design keystone is connected to all context stressors, but it is especially related to complexity, heritage, and cultural disruption.

Furthermore, this keystone can also be understood as being directly related to the concept of continuous heritage. Any design under the evolutive approach can be partially based on previously proven solutions (heritage), but it is also always an instance in a continuous design process that keeps adapting to new changes.

Therefore, adaptability in the end addresses the capability of a system to respond to environmental, cultural, and design changes. The more changes its contextual environmental requires, the more design changes need to happen to enable new system functions as a response. Adaptability of a system design can then be measured by its **relative design functionality**. This is how many functions a system geometry can perform given a specific interaction capability and a resource utilization level. Within this paradigm the goal is therefore to do ***better with less***. Thus, the more and better functions a system can perform, the with less resources needs and the more adaptable it is. Furthermore, under this standpoint the geometry of the system, including shape, assembly elements, etc. is critically related to both its functions (behaviors) and the resources used (substance) by it. In this case, the concept of behavior refers to the system itself, not to the design process as it could be understood under Gero's FBS framework (Gero and Kannengiesser, 2004).

An evolutive system architecture is aimed to be highly adaptable at both system and subsystem levels. This happens by design and from both hardware and software standpoints. It is also enabled by the nature of a continuum evolutionary design process as it has been introduced before. Thus, multiple instances can be created simultaneously as an outcome when modifying key system variables that define its most relevant characteristics. For instance, designing a clothing piece such as a firefighter jacket (Figure 114) under this approach would tackle multiple color and materials but also subsequent thermal and weather protection capabilities. When that jacket is designed, a baseline is created, and it can be easily be tweaked so patterns can address multiple sizes (e.g., small, medium, large) and different materials (e.g., color, texture, properties, reflective, etc.). This way the design can respond to different chemical and thermal situations, as well as alterations attend updates, upgrades (e.g., chemical resistance, thermal protection, etc.) and especial solutions such as identification, lighting conditions, etc. (Watkins and Dunne, 2015). Thus, the system itself is a one-off product, but it belongs to a species (collection) that includes systems with similar characteristics and multiple variations. However, even if the system itself does not require variations, the infusion of this perspective brings enormous benefits later regarding upgrades, work repurposing, and ultimately system efficiency. Designing for adaptability tackles implementation constraints and functional drivers upfront in the design process, benefiting both the overall relative cost of the design process (e.g., time, resources, workforce), as well as the system performance itself. If this approach is broadly infused within the culture of an organization, all initial efforts required to bring this process online, to outweighed by bringing new levels of adaptability across product lines, teams, and projects too.


*Figure 114. Firefighter protective jacket has a complex design architecure with multiple variations.*

Such design effort to create one unit is distributed over multiple instantiations of that architecture species and addresses multiple design variables represented by a networked framework of characteristics and variables, rather than a linear list or even a matrix of requirements. The

final product is a result of weighing these needs which are often interrelated or even opposed among themselves (Figure 115). In other words, under this evolutive perspective those relationships among variables are not necessarily constant and they could vary over time due to changes in the external context (e.g., design stressors). If such open-design and adaptable approach could be streamlined, even when tackling just a one-off solution (with heritage or not), then not only variations and upgrades of such solutions would be easier and cheaper to make, but they could be included as useful heritage inputs towards newer or even unrelated solutions. So, capturing and validating such relationships becomes in many ways part of the *genetic* material of both the system as well as the process. Thus, under this approach such system architecture could be defined from the perspective of an evolutive framework, as an adaptable network of interconnected variables that evolves continuously (Figure 115), rather than a static hierarchical structure (Figure 116). The system architecture then is defined by connections among most relevant variables, and adaptability means handling change in selected network nodes (blue lines) within a changing framework. The more fluid such a network is, the bigger the need is for an adaptable system architecture to reduce cost and to improve the efficiency and capability of any system design.

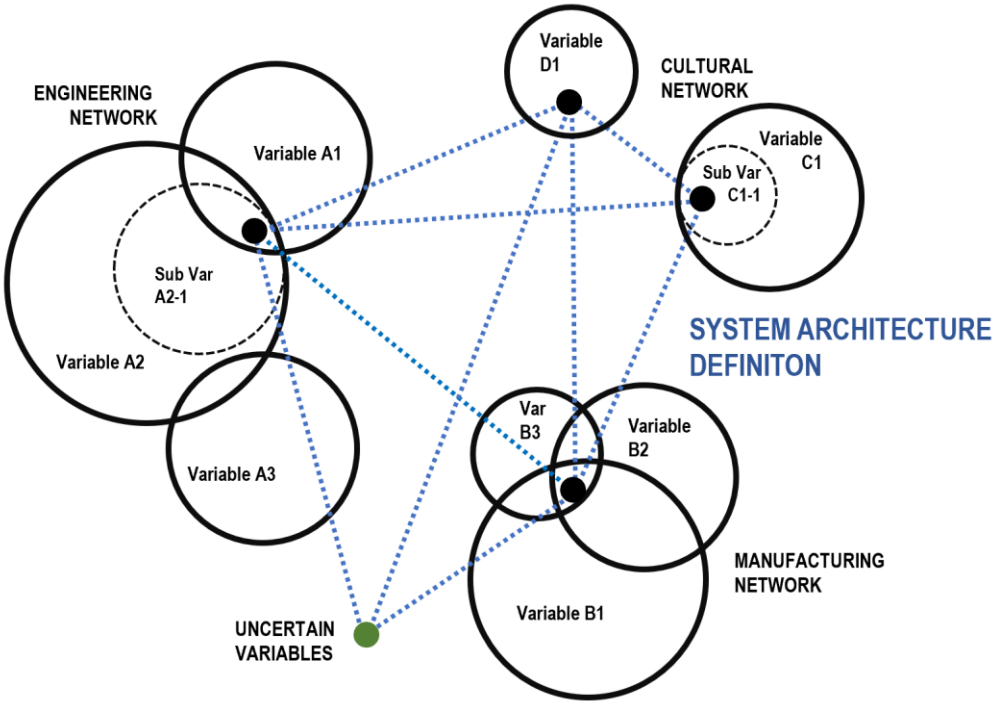



*Figure 115. Visual representation of an architecture definition based on an evolutive network of variables.*

However, from this framework perspective the adaptability of the system can be pushed further away to: [1] stress the design for more **efficient solutions** that use less resources, as well as [2] to turn the **uncertainty** inherent to the design, implementation, and operations phases into an advantage. In essence, the more areas within the network that a system architecture can address with less resources, the more efficient it becomes and the more uncertainty it can handle without increasing its re-design, upgrade, or interconnection design effort or cost. Furthermore, the earlier this is done in the design effort, the more efficient the process becomes as well and more likely the system is to be optimized.

Regarding such efficiency the initial design for the firefighter jacket could require it to be strategically redesigned or enhanced, by adding conditions such as detachable sleeves, integrated smart technology (e.g., sensors, heaters, etc.), as well as other more complex cultural fashion variables. Those parameters were nor part of the initial design, and therefore not included in the initial adaptability requirements and evaluation. However, these conditions are driven by future foreseeable uses, manufacturing constraints, and market changes pushing the limits towards a future system. If that architecture has been designed with adaptability in mind (evolutive), then part of its requirement definition (blue lines) would include also open 'nodes', or areas for possible or uncertain future variables. For instance, a new sewing technique could

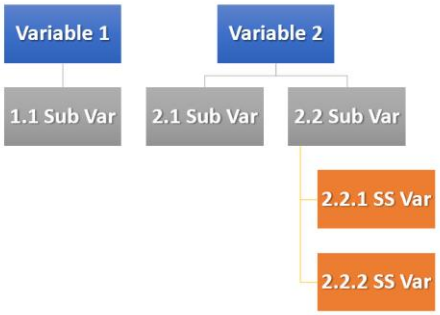



*Figure 116. Example of static variable framework.*

allow to easily add zipper lines to fixed sleeves so sleeves can be removed. While this would apply to the architecture itself, it would also affect the design method. Considering this early in its implementation can help discard too unique, and therefore not adaptable solutions. This of course, would seem highly inefficient and perhaps an unnecessary source of complexity. However, if curated properly it can be the key towards feasible quantum leaps in performance. The cost of including general open nodes (green), if done properly, is minimal in comparison with the benefits of reusing such design effort for later changes. Similar to what nature does with evolution, openness, and randomness in the system definition (genetics) opens the path towards adaptability due to the uncertainty in the reality. In other words, the more potential adaptability is infused early in the design the better the risk management (Costikyan, 2013) and less costly the system will be in the mid- and long-term.

Under this approach of evolutive adaptability, stressing the system is also a strategy towards achieving system reliability and resilience as well. If many or all of the multiple relationships describing the system architecture within such framework are addressed by a system, its design addresses known and most likely also known requirements. The more a system design is being pushed against all design paths, the more that design 'kills' efficiently the challenge or problem that system is aiming for. Thus, the more adaptable a system is towards addressing changes though a better with less approach, the more capable such system is to address uncertainties and more resilient it becomes. Furthermore, even if this approach is partially implemented, it still provides a solid foundation towards future design trade-space options and expansions.

However, the integration and addition of requirements can also lead to a hyper-integration making difficult future changes upgrades, repairs, or updates. We cannot forget that while an initial increase in complexity, means a greater effort and more variables (and requirements), in the ends it means much more efficient design efforts and less use of available resources. In essence this is what nature does, since the reference point is not design effort towards a point-design or a one-off architecture, but of the system (organism) as an instantiated part of a continuous evolution (species).

Then, pushing the limits of a system architecture towards higher levels of performance and adaptability is done through a careful process that builds upon [1] the balance between needs and resources, and [2] a synergetic connection among subsystems and disciplines within a requirements network. Chapter 5 will describe this process in detail. Furthermore, infusing a high level of adaptability in the design also has the benefit of better dealing with uncertainty. Within an evolutive architecture, the system is considered as an open solution, that is a family of solutions rather than a locked point design, which ripples across multiple levels such as subsystems, components, parts, and even strategies (e.g., manufacturing, marketing, etc.). Thus, design uncertainty is built up in the system as the likelihood, feasibility, and availably of statistical parameters that need to be captured, tracked, and used for subsequent optimizations.

The range of this critical keystone, varies incrementally across a range defined by these levels:

- **Unadaptable** (no adaptability). These system designs present the minimum number of functions with the maximum number of elements. These cannot handle high levels of design uncertainty efficiently, and they tend to gravitate towards rigid and often limited point-design single solutions.
- **Adaptable** (balanced adaptability). Systems designs in this category present a balance between the number of functions and their constitutive elements. They lean towards short series and limited customizable solutions.
- **Evolutive** (highest adaptability). On the other extreme of the spectrum these designs present the maximum number of functions with the minimum number of elements and components. They handle high levels of design uncertainty efficiently and they gravitate towards open solutions or families of solutions addressing open requirements very well.

The concept of evolutive adaptability could be applied to any system architecture design regardless of whether it is physical, digital, or virtual. However, this is especially relevant for complex and smart hardware-based system architectures. The especial nature of these complex systems integrating complex physical geometries, actuator-driven functions, and data-driven operations certainly highlights multiple intertwined aspects of the evolutive approach as the next chapter will present.

### 4.2.2. Reactivity

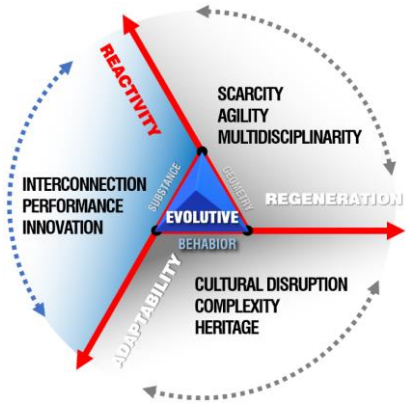



*Figure 117. Reactivity within the evolutive tetrahedron of system architecture design.*

Reactivity is the second keystone within the evolutive tetrahedron of system design (Figure 117). Similarly, this one tackles all context stressors that were summarized in section 4.1. However, this concept is especially related to interconnection, system performance, and innovation.

From the evolutive standpoint, a system architecture is dynamic, and it presents a multidisciplinary nature to deliver high-performance characteristics. For instance, a mechanical evolutive system could be competition race car that optimizes thermal performance and mass reduction. Furthermore, a physical and adaptable configuration responds to changing design requirements, with key control and management functions associated with it. In this mechanical example, the management of electro-mechanical actuators in the assembly, could allow improvements and adjustments over the different phases of the race to improve performance, as well as upgrades based on data collected over time. Therefore, physical design, actuator controls, and data-driven decisions are combined within a complex adaptable evolutive architecture to react against environmental or design changes. In this case, reactivity is essential to address the **dynamic holistic synergy** of the system components, as well as its capability to manage its adaptability across the multiple realities of a system (physical, digital, virtual, databased, etc.). Reactivity is also related to the transient nature of the evolutive system complexity, as the continuous development between the system and its environment. Hardware (geometry), software (behavior), and resources (substance) are all integrated within the capability of the system to interact with external and internal integrative changes.

Therefore, **transient system interaction** is the measurement behind the reactivity of the system and is defined as the number (and complexity) of reactions the behaviors of the system can provide given a specific system geometry and utilization of resources. The more interactions with the environment a system architecture is capable of, the more reactive it is. The less interactions the system requires to handle external changes, or in other words the smarter and more adaptable it gets, the more efficient the system architecture becomes. In essence, the main goal brought by this principle is for the system to become ***smarter (more reactive) with less.***

Nowadays, modern complex system architectures across fields are becoming more and more robotic in nature. This means they increasingly combine software, hardware and data, through some type of intelligent management, assessment, and control (Chen et al., 2018). For instance, a modern car today has several million lines of code (Desjardins and McCandless, 2017), which is a growing tendency as autonomy starts becoming a standard capability of any car in the future (Towns end, 2020). The same happens with apps or software, as well as phones, vehicles, appliances, and many other objects around us today. At the same time, the amount of information used within our systems keeps increasing, so another growing technology trend brings connectivity among all those systems such as the internet of things (IoT). All this portrays a near-term world of interconnected devices and sensors all over (Soro et al., 2019). Thus, the growing infusion of software-based behaviors and control in any hardware system is evolving into a ubiquitous and increasing smart capability (Figure 118) for every one of these systems. As such, intrinsic design rules for any hardware or robotic-driven hardware of the system will change. For instance, under this approach a house would manage its own lighting or energy consumption based on user interaction, while a car will drive itself changing speed, suspension profiles, and torque depending on the road conditions and the environmental stimuli. Our human-built world is becoming smarter, and suddenly thermal performance, mechanical fatigue, or system longevity will be driven by such inherent capability. Thus, an approach like this will bring great opportunities in that balance between scarcity of resources and complexity of the system.

These smart systems need an equally smart design effort to harness, improve, and upgrade the reactivity capability itself, but also a way to match hardware adaptability, system efficiency, and interactivity across systems. In other words, an evolutionary approach implies the capability to prepare and design for a constant flow of information and interactions among systems, components, and their environment. These growing evolutionary approaches will change business and industry models (Kranz, 2017), affecting how we design, build, manufacture, and use objects around us.

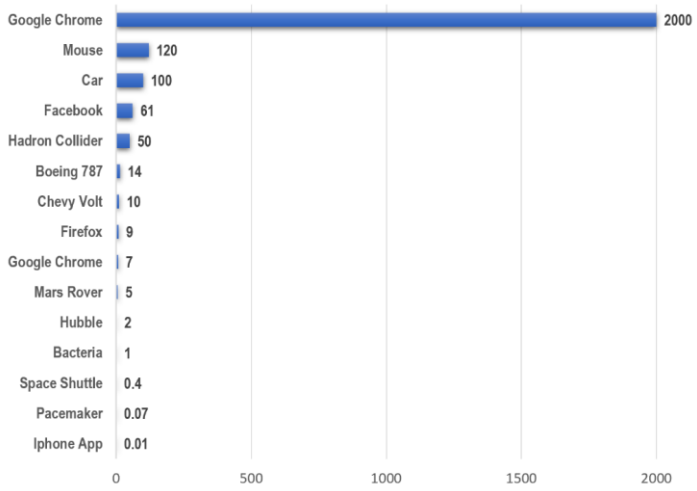



*Figure 118. Millions of lines of code across different systems - multiple sources (Desjardins and McCandless, 2017)*

As previous sections referenced, organisms (systems) in nature are not isolated. They are in constant interaction with their environment, with other organisms, and with themselves, regardless of the volume of information and the vehicle for such exchange or survey. That interaction with the environment drives: [1] system design efforts including definition networks like Figure 118 shows, as well as [2] interaction processes enabling the system to react and adapt. Figure 119 shows graphically how this concurrent flow affects each process. In general, the operation of an interactive system in the environment allows it to both send and gather data, which is used to perform variations and changes. These depend on the capabilities of the system (e.g., moving parts), which again enable the best reaction towards those stimuli. However, this process also has consequences towards the definition of the system and its subsequent design efforts, enabling also changes in the design that could improve its performance based on each new situation (continuous heritage). As this small summary presents, the system architecture needs then to be designed to enable such process, having reactivity and adaptability at the core of its definition. Beyond these, manufacturing and operations constraints need to be integrated as well. Currently, highly reactive system architectures, such as autonomous cars, present a different level of autonomy and data-driven induced behaviors as part of the design process. Hence the design process itself should be changed, optimized, and evolved based on the information management inherent to this key characteristic. Under this approach, the more a system architecture with an intelligent baseline is used, the more the design process would change based on such feedback data loop. This becomes feasible by an integrated and concurrent data-hardware architecture (Figure 119).

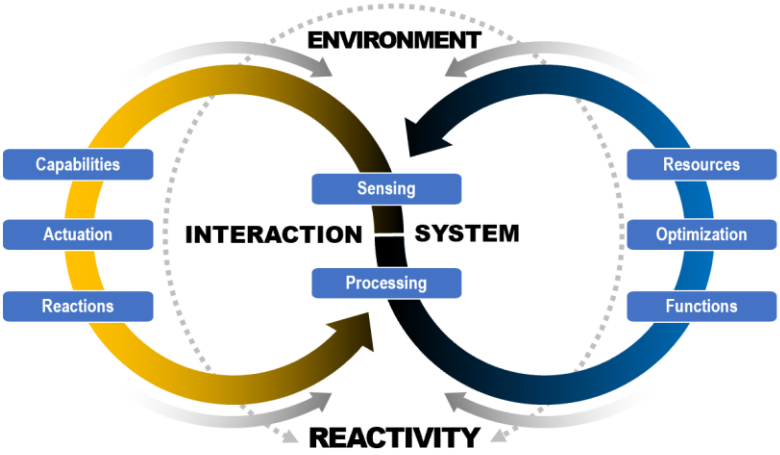



*Figure 119. Evolutive reactivity, system design, and system interaction concurrent flow.*

The range of this critical keystone, varies incrementally across a range defined by these key points:

- **Passive** (no reactivity). These system designs present the minimum capability of interaction with the maximum number of functional elements and resource utilization. They cannot handle many external or unexpected changes, and they tend to gravitate towards low-tech and simple solutions.
- **Active** (balanced interaction). Systems here present a balance among the system interactivity and complexity and resources required across their lifecycles. Programable, modular, and upgradable systems belong here.
- **Reactive** (full reactivity). These systems present the maximum capability of interaction with the minimum number of functional elements and resource utilization. They are highly smart systems such as advanced robotics, AI-driven architectures, autonomous systems, etc. They can handle many external and unexpected changes efficiently with highly interactivity. They gravitate towards high-tech, biological, and software-based solutions.

This evolutive keystone does not only apply to very complex and high-tech systems, but it can also be identified in designs as simple an adventure jacket. Such an evolutive jacket could simply have sleeves that can be removed, and openings or pockets that could tweaked by hand for thermal management reasons. This evolutive principle of reactivity is applicable to all technical and creative sectors. While this approach is emphasized towards hardware-based system architectures, it can also be applied to software or virtual architectures requiring both interactions and adaptability.

### 4.2.3. Regeneration: Resource Performance and Sustainability

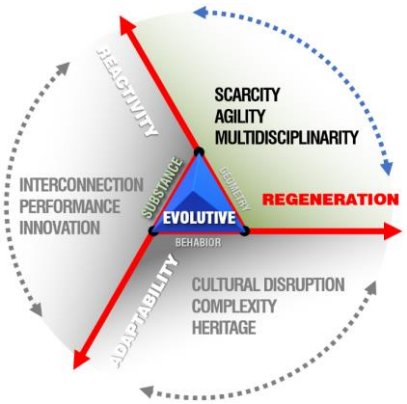



*Figure 120. Regeneration within the evolutive tetrahedron of system architecture design.*

Finally, regeneration is the last keystone of an evolutive system architecture. Similarly, this tackles all general stressors mentioned in section 4.1 as the last foundation for the evolutive tetrahedron. Among them, regeneration is especially connected to resource scarcity, agility, and multidisciplinarity.

Regeneration is about the utilization, management, and restoration of resources across the full lifecycle, whatever they may be. These could include energy, building materials, computing code, mechanical components, or workforce availability, among many more. Resources do not need to be physical, and they do not need to be man-made either, however they all include **substance**. This is understood as what the system architecture is made of or what resource is required for its operations. So, this concept relates to the **resource lifecycle optimization** of a system within a given external environment. The sources used to make the system and their management are part of an evolutive design process. Resource considerations should be done across the full lifecycle from generation to recycling, including: [1] energy, [2] materials, [3] people or workforce, [4] data, and [5] coding or programming.

Therefore, a key measurement behind this constitutive concept of an evolutive system architecture is the consumption and utilization of resources for a given system geometry and reactivity capability. In other words, this is the concept of **resource utilization** within this context. This addresses the consumption of resources by the system across the lifecycle from design to decommission. Furthermore, it considers both all resources used to design the system, to develop it (evo-devo), used by the system itself, and by the relationships with its environment be it physical, digital, or both (eco-evo-devo).

From this perspective, an evolutive system is aimed not only to be sustainable but to become resource positive (e.g., producing more energy that it consumes) or regenerative (Lyle, 1996). The first has clear implications on a cradle-to-cradle approach, and it is not just about pollution or scarcity, but about efficiency across design, implementation, operations, and all the way to decommission (Bhamra and Lofthouse, 2016). This consideration of resources could be negative (the system only consumes), neutral (the system is sustainable), or positive (the system replenishes resources). There could also be multiple grades across these which are applied at system level as well as at a component or sub-system levels.

Hence, the management of resources within an evolutive system is related to the concept of **eco-evo-devo-lifecycle**. This is an evo-devo approach that looks the development process of the system itself by considering the environmental ecosystem and the lifecycle of the system from a resources standpoint (Figure 121). Under this perspective, concepts such as sustainable recycling (Bhamra and Lofthouse, 2016) and cradle-to-cradle (McDonough and Braungart, 2010) are

integrated into the understanding and optimization of resource utilization across multiple phases:

- **Design** in this phase is about considering both [1] all resources required to create the design (e.g., workforce, tools, computing powers, paper, etc.), and [2] resources required by the system to function.
- **Implementation**. Physical and digital manufacturing of a system architecture require both direct and indirect resources such materials, tooling, and coding. In this phase it is critical to consider especially all losses due to inefficiencies and other intermediary steps. This phase should also address integration, transport, and installation.
- **Operations**. This phase addresses all resources required to operate, maintain, and even upgrade the system. Also, system operations are critical in this phase from both active and passive standpoints since it affects all the other phases in the lifecycle. Among other resources, workforce management and coding are tracked here.
- **Decommission**. Finally, this last phase considers resources regarding the repurposing, recycling, or reusing of systems at the end of their life span. This critical phase goes beyond the sustainability of the system at any level of resource utilization and connects the end of the lifecycle with the initial design process.

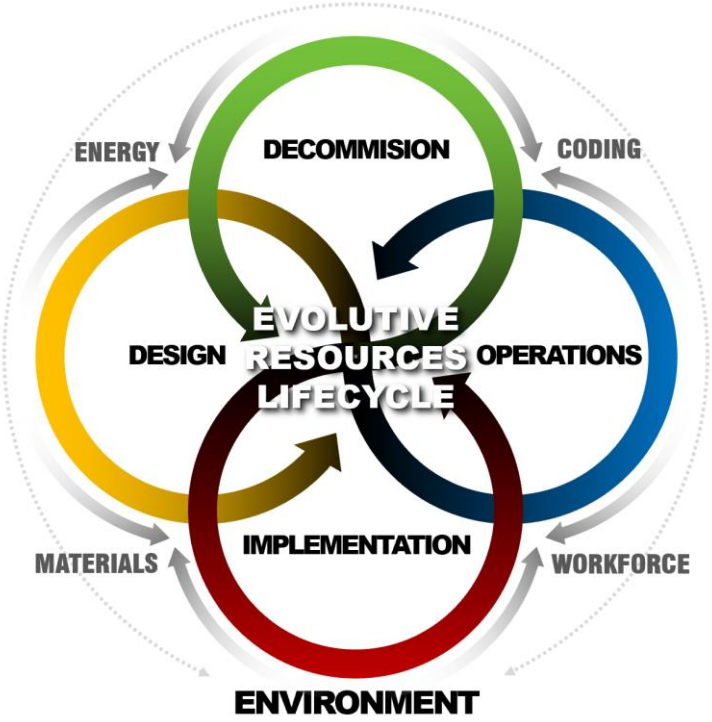



*Figure 121. Full evolutive resources lifecycle within the evolutive systems design process.*

From this perspective and across the lifecycle of an evolutive system, the relationship between the system and its environment is always considered as a design constitutive, regardless of any given design requirements. This relationship drives the sustainability of the system, the design posture towards resource scarcity (section 2.1), and its implementation cost. It also conditions and stresses both system design and design methodology to deliver better performances. For instance, if the design allows to infuse, use, or swap materials and energy sources among other constraints, it will increase the system design adaptability and potentially the system performance in the long run. Similarly to the eco-evo-devo approach, the study and design of the system is always done under the light of its relationship with its changing environment. **Any system is therefore defined by the design of system and its context**. This context could be the assembly in which resides (e.g., mechanical part), the natural environment (e.g., building), its software framework (e.g., app), etc.

Thus, this third keystone is a key characteristic for any system architecture, but it is especially relevant for evolutive architectures under scarcity-driven environments. As stated previously, energy scarcity is a general global constraint for

humanity, and energy efficiency is particularly critical for longer and even more affordable operations. The use of energy is related to the use of natural resources, including material extraction, processing, prototyping, manufacturing, and all the way to recycling (Johnson and Gibson, 2014). This always needs to be considered for scarcity, cost, and reliability reasons as it is critical across all general, design, and even cultural stressors (chapter 2).

For instance, cellulose-based recyclable materials that are available in the area are key to create an evolutive approach towards printing products such as a magazine as seen in Figure 122. Regularly, the final selection of material and vendor for the printing will come at the end of the design process. But an evolutive approach considers such key details this early in the process, and includes local sources, alternatives, recycling schemes, and manufacturing constraints. Furthermore, the approach should consider how the publishing can replenish trees and energy used during its design, printing, and delivery. This leads to managing inks, formats, vendors, and transport, as well as marketing approaches, environmental aspects, and other social constraints. All these aspects are considered towards making the final product richer, more adaptable, and more tuned to its context. Such an approach requires an extra effort for both the designer and the design process, and it could certainly become overwhelming. However, the key is assessing which one of those variables and connections in such system-environment interaction are critical. Chapter 5 will present this process and its method.


*Figure 122. Resource regeneration during the design cycle considering both system context and system design.*

Aiming for a surplus in the system provides several benefits from a design standpoint since [1] it stresses design requirements enabling more adaptability, [2] it creates performance margins, and [3] it implements key environmental principles with key economic, social, and conservational consequences. Therefore, an evolutive system architecture could often present key design trades that both enable and use these principles. Such principles include among others: multifunctional system architecture, repurposing, easiness in its upgradability, recyclability, mass reduction, cost reduction, return of investment increase, etc. In essence, the design principle behind this keystone is about **doing more with less** from the standpoint of resources. The range of regeneration, varies incrementally across a range defined by:

- **Depleting** (net negative resource consumption, consumer). These system designs present the maximum consumption of resources and no replenishment strategy. They tend to present lower levels of performance, less adaptability, and less reactivity. They also gravitate towards non-recyclable, disposable, and unsustainable solutions. An example of this area could be thermomechanical systems with a high carbon manufacturing footprint.
- **Sustainable** (neutral resource consumption). System designs in this category present a balance between resource consumption and replenishment. These include sustainable systems and carbon neutral solutions.
- **Regenerative** (net positive resource consumption, prosumer). On the opposite side these designs have a minimum consumption of resources and a full replenishment strategy. Thus, they also have the highest levels of performance, with more adaptability and higher system reactivity. These systems gravitate towards net positive and regenerative solutions such as and CO2-sequestration-based electromechanical systems.

The concept of regeneration as a keystone of an evolutive system could be applied and observed across many domains including thermomechanical, digital, computational, and biological, among others. Therefore, under this approach

energy, matter, and information (data) are multiple faces of the same reality as the substance of complex systems. This is a fundamentally a holistic way of looking at any system architecture, independently of its complexity or scale.

Examples of sustainable design that aspire to be integrated with nature can increasingly be seen in multiple sectors such as clothing, consumer products, and houses (Kwinter, 2017), among many more. Nevertheless, designing and producing for the abundance of resources, rather than the rationing of available resources (McDonough and Braungart, 2013) is what regeneration opposes. This in essence means [1] to design for either a system architecture that produces more resources than it consumes (Mang et al., 2016), or [2] to have an integrated close-loop functional scheme so the system restores, renews, and transforms any used energy and resources (Burke, 1999, Colozza and Maloney, 2003). This is especially applicable towards the development of energy production systems, large-size projects, and infrastructure-oriented architectures (Hemenway, 2015). Among other smaller scale examples, we could identify sustainable buildings, regenerative energy systems (Alotaibi et al., 2020), or plant-based food production systems to name just a few. So, this approach is a growing trend due to the scarcity stressors and the increasing complexity of systems.

## 4.3. Evolutive Design Drivers

Previous sections have presented the evolutive system architecture approach in response to global design stressors and methodology gaps. These fundamental keystones characterize any evolutive systems among the large category of general complex systems. Figure 123 summarizes this graphically highlighting the three keystones at the base of the evolutive design tetrahedron: adaptability, reactivity, and regeneration. However, it is necessary to fully define an evolutive system to address more specific design drivers behind these overarching keystones principles.

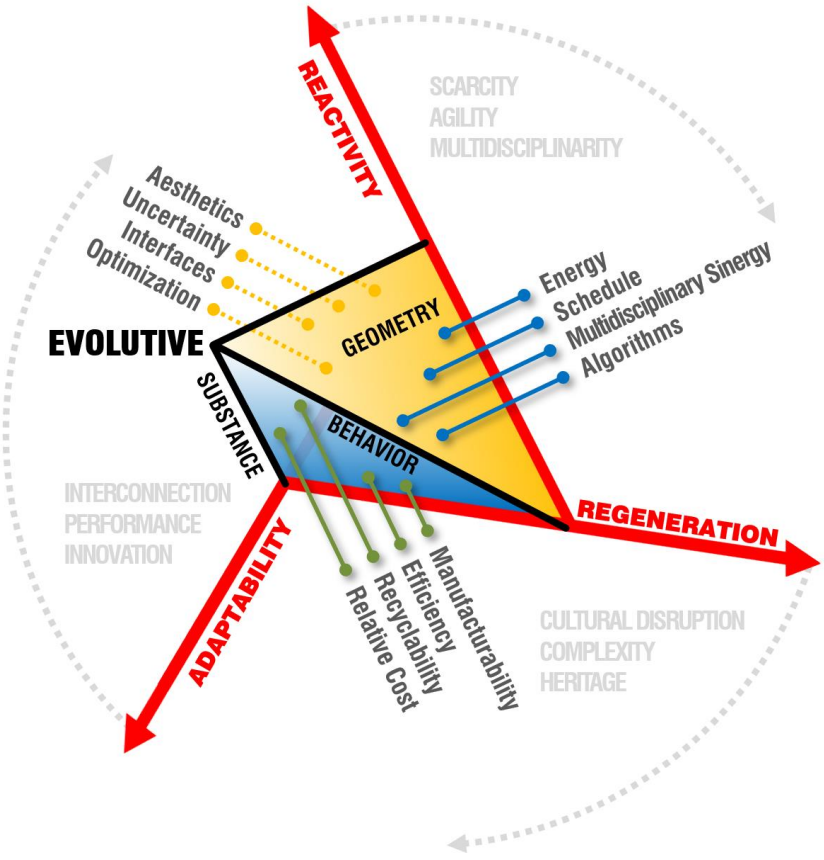



*Figure 123. System design drivers as faces within the evolutive tetrahedron.*

These keystones are partially based on ancestral basic architectural components for any building, which were described by Vitruvius during the Roman Empire (Vitruvius, 2012). There are structure (*firmitas*), function (*utilitas*), and perception (*venustas*). But these principles need to be adapted to current times. It is relevant to highlight that unlike other modern interpretations such as the FVS model by Gero (Gero and Kannengiesser, 2014), these principles and subsequent design drivers relate to the system itself and not to the design development process behind it since it is a practical approach.

From the described base of the evolutive tetrahedron, three faces or planes represent these design driver groups: [1] geometry, [2] behavior, and [3] substance. Each one of these planes is opposed to its more direct keystone (adaptability, reactivity, and regeneration), which are the vectors in between these planes. Understanding these drivers while considering both known and unknown relationships among them is key for designers and design processes to produce good complex architectures. The following sections elaborate in detail these design drivers for any given evolutive system architecture.

The following descriptions are based on [1] the study of general needs regarding complex systems, [2] design and systems engineering gaps identified in section 3, [3] key characteristics of evolutionary, adaptive, and evolutive systems, and finally [2] almost two decades of practice designing complex systems across multiple technical fields.

## 4.3.1. Geometrical Complexity

In response to design requirements and context stressors, the development of an evolutive system architecture design involves a continuous cycle that creates a geometry. This activity includes among others the definition and development of volumes, shapes, component three dimensional assemblies, interfaces, mechanical properties (e.g., center of gravity), mass estimates, material design constraints, packaging studies, deployment studies, integration feasibility studies, etc. This geometrical complexity is always in constant state of change within an evolutive process. Such geometrical continuity could be implemented using advance computational systems (e.g., generative design tools), as well as low-tech techniques (e.g., pen and paper) just by keeping design trades open.

Practically this means that the multiple aforementioned trades are flexible and subject to an overall architecture design strategy. Thus, under this approach both designer and the design workflow always maintain the design as unfinished and keep floating new foreseeable design needs in the trade space of design solutions. This is not because the component requires that particular need to be addressed right away, but because it enhances the adaptability of the current solution which improves ultimately both performance and efficiency.

The geometry face of the tetrahedron is: [1] define by the edges of reactivity and regeneration, [2] limited by the behavior and substance faces. This means that for a specific system architecture design seed, the geometrical aspects of the design are bound by the materiality of the design and the functional behavior. In other words, multiple designs exist in planes perpendicular to the adaptability axis. These axes hold the geometrical drivers, while the associated face (yellow face, Figure 123), is bound by eight key design drivers that are interrelated among themselves and with the other faces (Figure 124). A three-dimensional body is selected because those complex connections happen metaphorically within the internal volume. Surfaces and edges define specific parameters and approaches, while system connections occur multidimensionally underneath. Geometrical drivers include among others:

### *4.3.1.1. Aesthetics (perception)*

A complex evolutionary architecture does not only mean it is only technical solution for a machine or component only driven by its functions. Such complex architectures could be designed for human use and interaction. Indeed, under this light the perception of such a solution from an aesthetical standpoint is a crucial and very complicated. Styles, cultural references, political notions, and even social nuances along with many more are part of a design process. Often, the management of this complexity relies on the capability and experience of the designer, as well as the cultural and heritage trades of the institution or field of practice. Thus, perception of the system from the user-center or culture-centered perspectives needs to be addressed, captured, and balanced within the development of many evolutive systems.

The goal of this research is not to describe such complexities, but rather to emphasize they are a reality of the system, with quantifiable and quantifiable variables and parameters. An evolutive system architecture will not be complete if this area has not been addressed and certainty its subsequent design workflow would be incomplete if it cannot handle it. At the same time and beyond perceptual aspects, aesthetics could also be used within an evolutive approach to assess and manage

other secondary relationships across design drivers. This not only addresses often the 'technical' intuition of the designer about the performance of the system, but it also establishes a comparative reference across solutions. For instance, the components in the design of a suspension system in a race car, are going to provide a more extreme, more lightweight, and lower profile of the car, presenting a very different aesthetics than a similar suspension system now applied to a compact utility car. Cost, optimization (e.g., mass), etc. could be related and tracked by different aesthetics as and styles not only as a design driver, but as a foundational principle for a development process.

For instance, the use of certainty materials, a practical manual craftmanship process, or a design allowing multiple modular solutions by an increased the number of interfaces would change the perception, user experience, and style of the system. As an individual design driver this one presents the following general characteristics:

- **Direction**: It tends to be a set driver thus it is basically unidirectional, affecting other drivers.
- **Criticality**: medium to high depending on the other drivers.
- **Complexity**: It is non-quantifiable driver for products mostly, but it could be also quantifiable (e.g., material quality).
- **Range**: this mainly affects product (prod) or system architecture, but it could influence the process indirectly too.

#### *4.3.1.2. Design for Uncertainty*

When designing a geometry that is the shape and material organization of a system section 3.1 presented the key differences between descriptive and prescriptive methodologies. In the first one, an initial concept (synthesis) is created as the starting point of the process. In the second a set of axiomatic rules and analysis are used to develop the concept. Both approaches are iterative in nature, enabling other approaches such as design thinking, integrative, etc. However, in a complex system and especially under the continuous evolutive approach uncertainty in the design is a critical drive in the process. Since the system architecture needs and it is forced to respond to environmental, design, and context changes, the unknown is critical. This means that system needs to allow design margins to enable new adaptability schemes, as well as to integrate current design changes and future design traits. These design drivers could be present across fields, systems, and practices. Examples of this can be found in the ultimate changes driven by packaging constraints, numbers of user, or final range of the system, among others. Uncertainty as an individual design driver presents these general characteristics:

- **Direction**: this is a bidirectional driver across the design space.
- **Criticality**: low to medium. Because it is not specified it not considered as high.
- **Complexity**: it could be both quantifiable and quantifiable.
- **Range**: this driver affects both product (prod) and process (proc).

#### *4.3.1.3. System and Component Interfaces*

A key design driver within any complex system, especially for a an evolutive system architecture, is the interfaces among components, subsystems, and other adjacent systems. It is crucial to identity, manage, and describe the number, nature, and interrelation of these interfaces between the integrated components of a complex system. These interfaces certainly affect both processes and products. Beyond the traditional system engineering approach, they require considering geometry, materials, and data. Good examples of this can be seen in modular system architectures as well as those requiring updates and upgrades frequently. This driver presents the following general characteristics:

- **Direction**: it is also a bidirectional driver.
- **Criticality**: is medium to high within evolutive systems.
- **Complexity**: it is both quantifiable and quantifiable as a driver.
- **Range**: it affects both products (prod) and processes (proc).

#### *4.3.1.4. Design for Optimization*

Under the paradigm of continuous design and with the overarching objective of achieving higher performance levels, optimization becomes a critical design driver. This not only makes a difference in reducing resources consumption and improving multifunctionality among other goals, but it is also about approaching the system's design from an adaptable perspective. The evolutive process is about a constant design workflow and a new generation that both adapts to new conditions and surpass previous heritage solutions by integrating them.

Finding the better balance between system characteristics and design drivers is constantly present under an evolutive approach. Examples of this are mass reduction, energy efficiency, the reduction of the number of assembly components, or manufacturing cost reduction, among many more. This driver has some general characteristics such as:

- **Direction**: it is always bidirectional.
- **Criticality**: is high within high performance evolutive systems.
- **Complexity**: It is a quantifiable driver.
- **Range**: mainly this affects the product (prod) or system architecture.

## 4.3.2. Functional System Behavior

The second big group of design drivers emphasize the response to the overarching principle of reactivity, while also addressing adaptability and regeneration aspects. As previously mentioned, this category relates to the characteristics of the system architecture itself and not to the design process per se. Given the adaptable nature of an evolutive system, the design process needs to address the management and optimization of such changes. As section 4.2 developed, reactivity classifies systems from passive to highly reactive both against their context environment and within their subcomponents.

Thus, designing towards those interactions involves going beyond the continuity of the process, to study the system from a functional standpoint. Such functions will always be intrinsically related to the geometrical adaptability of the system, as well all available resources used or needed, both of which are related to the other two faces and edges of the tetrahedron.

These behavioral design principles include governing forces that enable the adaptability of the system and materialize its reactivity. For instance, any complex system needs energy to work and to be manufactured. However, a very efficient energy management can be executed if the system is highly reactive, allowing a higher level of adaptation to different power needs. So, all these principles and drivers describe a highly circular and networked approach that enables the complexity of a system through the interconnection of multiple design variables and principles. Among the multiple design drivers related to the behavior of a complex reactive system the following are initially highlighted within the evolutive approach.

### *4.3.2.1. Energy*

Energy is universally a critical design driver which is connected to every intrinsic aspect of the system architecture, functions, and related design processes. Firstly, addressing energy from an evolutive design driver standpoint implies to considering the use, consumption, and production of it across the full lifecycle of the system. Within such a design consideration there are three levels to be studied: [1] the cultural context and operative environment of the system, [2] all energy needs across the system lifecycle including consumption, regeneration, and overall efficiency, and finally [3] key operational modes and functions of the system that have direct consequences towards the other two points. Given a specific design, this is essentially about where the system operates, what it needs, and how it could improve its efficiency through reactivity and adaptability. As a design driver and a tool, this presents the following overall characteristics:

- **Direction** is always bidirectional, but usually there is preferred one.
- **Criticality**. This driver always has a high criticality within high performance evolutive systems and processes.
- **Complexity**. It is a quantifiable driver, but it can also have multiple associated qualifiable drivers.
- **Range**. Energy considerations affect the design of both products (prod) and processes (proc).

### *4.3.2.2. Time and Schedule*

Complexity often means more time and more demanding schedules with consequences that ripple across the system. This limits the number of resources available to a company or a designer and greatly constrains the possibility of a better design. The more complexity that is required, the heavier the influence of heritage and subcontracted tasks become. The evolutive approach specifically tackles these challenges from multiple and complementary perspectives such as:

- **Simplification**. The design of the system should reduce as many manufacturing steps and the number of parts as much as possible. This increases the multifunctional aspect of its components, subsystems, and overall architecture.
- **Multitasking**. The design of the system and its related process should enable all possible a synergy among

components, agents, efforts, and resources.

- **Continuity**. Making design requirements much more thorough and broader allows creating better solutions in the short terms, and easier upgrades and new systems in the long term. Like an athlete, the more the design workflow and the system is 'trained' to address current requirement needs, the easier it is to get to the next the level.

Among the most relevant characteristics of this design driver, these are critical:

- **Direction**. It is mainly a bidirectional driver.
- **Criticality**. This driver has always high criticality within complex systems.
- **Complexity**. It is a quantifiable driver with multiple associated qualifiable drivers.
- **Range**. It strongly relates to both products (prod) and processes (proc).

#### *4.3.2.3. Multidisciplinary Synergy*

A key aspect of the evolutionary approach developed in chapter 5 will develop is the fact that the system architecture is built upon synergies across disciplines and subsystems. This design driver also has consequences across all the other drivers and dramatically influences any further evolutive process. Rather than looking at the system from a serial disciplinary standpoint, as chapter 2.9 presented, the system could be studied from within the connections among those disciplines. For instance, rather than looking at it from a purely mechanical and then a thermal standpoint, an evolutive design architecture handles synergetic thermomechanical requirements and questions simultaneously affecting and enabling both. Addressing synergy within the design is also addresses opposing forces that have physical, cultural, heritage, and business backgrounds. Outcomes of this are: [1] multifunctional architecture designs, [2] integrated implementation steps, [3] simpler schedules, [4] increased sustainability, [5] improved efficiency, [6] lower cost, etc. This multifaceted driver could be characterized as:

- **Direction**. It is always bidirectional, and it has preference nature as well.
- **Criticality** regarding the system design requirements goes from medium to high.
- **Complexity** is both a quantifiable and qualifiable driver here.
- **Range** varies from products (prod) to processes (proc).

#### *4.3.2.4. Algorithm*

An evolutive system is by nature a smart system, meaning it is highly interactive. This also means that its behavior in terms of operative functions is driven by programming and data, so the behavior of the system has an algorithmic nature. Furthermore, the design itself can also be algorithmic since its geometry, reactivity, and even regeneration scheme could be based totally or partially on algorithmic models. Examples could be found in 3D printed generative components, autonomous self-driving cars, and interactive robotics. This algorithmic nature coexists with other designer-driven and design-workflow-driven decisions, models, and guides. Nevertheless, an evolutive system or process along with their subsequent development processes need to address both. This specific driver presents the following characteristic traits:

- **Direction**. This driver is mostly bidirectional.
- **Criticality** goes from low to high. It is high on reactive evolutive systems and processes.
- **Complexity** is a quantifiable driver, but it can have associated qualifiable drivers.
- **Range**. This driver also affects product designs (prod) and process designs (proc).

### 4.3.3. Material and Data: Substance

Finally, the last group of design drivers under behavioral functions relates to the implementation of the system and the resources used in such processes. These systems could be physical (e.g., metal alloys used in the structure) or digital (e.g., programming language, GUIs, and datasets). Geometry and design provide the rules to implement and forge the system, while behavior drivers tackle how it performs. However, all three groups are intimately related among each other. Within this collection of drivers there a few critical ones that are elaborated upon in the following sections.

Given a specific context and framework, or in other words an ecology, we could look at the design process from a **resource standpoint** that including energy. The utilization of resources (from less to more), the capability to restore them

(from depleting to regenerative, Mang and Reed, 2017), and finally any achieved functionality (from better to worse) provide us with a three dimensional reference system (Figure 125).

In such a framework the complexity of the system architecture becomes the curve connecting system solution extremes. An evolutive architecture is defined by such coordinates, as is the design process. Thus, the goal is to be efficient, meaning more complexity and more functionality, with less resources and more regenerative capabilities. The more this is achieved, the more resilient the system will be within a scarcity context. Regeneration of course, is key for an evolutive architecture design, since any given complex system requires full design definition, development, and optimization among its parts and its context (e.g., environmental, commercial, social, economic). With this, complex design here is always continuous, contextual, and evolutionary in nature, as well as the result of a specific process including the following drivers.

#### *4.3.3.1. Implementation*

A given system design, behavior, and selection of resources must have a feasible way to be implemented. Within an evolutive architecture, such development and implementation are part of the system itself (evo-devo). So, this driver is about the manufacturability on the physical side, as well as its programmability on the digital side, among others. In other words, it is about the feasibility of the system across the full design and lifecycle phases. For instance, a specific design geometry could have multiple ways to be implemented with multiple consequences and dependences across all other design drivers. Chapter 5 and 6 will elaborate more about this driver, which present the fowling overall characteristics:

- **Direction** is bidirectional in nature with multiple caveats depending on the nature of the system.
- **Criticality** across systems and processes goes from low to high.
- **Complexity** here is a quantifiable driver, but it can also have multiple qualifiable drivers associated to it.
- **Range** varies from products (prod) to processes (proc) like the other drivers.

#### *4.3.3.2. Relative Cost*

Cost is both a driver and a consequence. As a driver for an evolutive system, it is about the relative balance across options. These options could be resources, workforce, energy, as well as associated monetary values, among others. Its consideration is not more special in an evolutive system than in any other approach. However, its approach is broader in the sense that cost is not only monetary but also related to decisions made over the continuous development of the system.

Heritage solutions and standards do have a critical influence within this driver since they often drive the decision tree behind any development or design of a complex system. It is important though to remember that relative cost is constantly changing, therefore it is more of probabilistic driver than an absolute constraint. Part of the rationale behind the evolutive approach could be summarized in managing this driver, which presents the following characteristics:

- **Direction** has a directional nature, but it could drive other alternatives bidirectionally.
- **Criticality** always present a high criticality for this driver.
- **Complexity** is mainly quantifiable driver.
- **Range** goes both products (prod) and processes (proc) simultaneously.

#### *4.3.3.3. Efficiency*

Finding balanced solutions among opposing forces is often the most complicated aspect of a system design task. Common examples of this are for instance the balance between [1] mass and power, [2] mass and thermal performance, and [3] volume and complexity, among many more. This balance could be tuned by the efficiency of the system, which is often related to a system optimization based on synergy and refined algorithms. Chapter 5 will elaborate this in more in detail. Therefore, efficiency here is an everlasting search among opposites with the following characteristics:

- **Direction** is mostly bidirectional unless it assessed as a design requirement or constraint.
- **Criticality** goes from low to high.
- **Complexity** is a quantifiable driver here.
- **Range** affects the design of both products (prod) and processes (proc).

#### *4.3.3.4. Recyclability*

An evolutive use of resources addresses the full lifecycle considering: [1] origin, energy, and cost, [2] reduction along the system life span, and [3] recycling and or repurposing. Bringing this consideration upfront not only tunes solutions to global stressors (e.g., scarcity), but it also enables a more robust and reliable system. This is characterized by:

- **Direction** is mostly bidirectional if trade options are possible.
- **Criticality** is high for sustainable or regenerative evolutive systems and processes.
- **Complexity** is both a quantifiable and qualifiable driver.
- **Range** also affects the design of both products (prod) and processes (proc).

## 4.4. Interrelationships Among Design Drivers

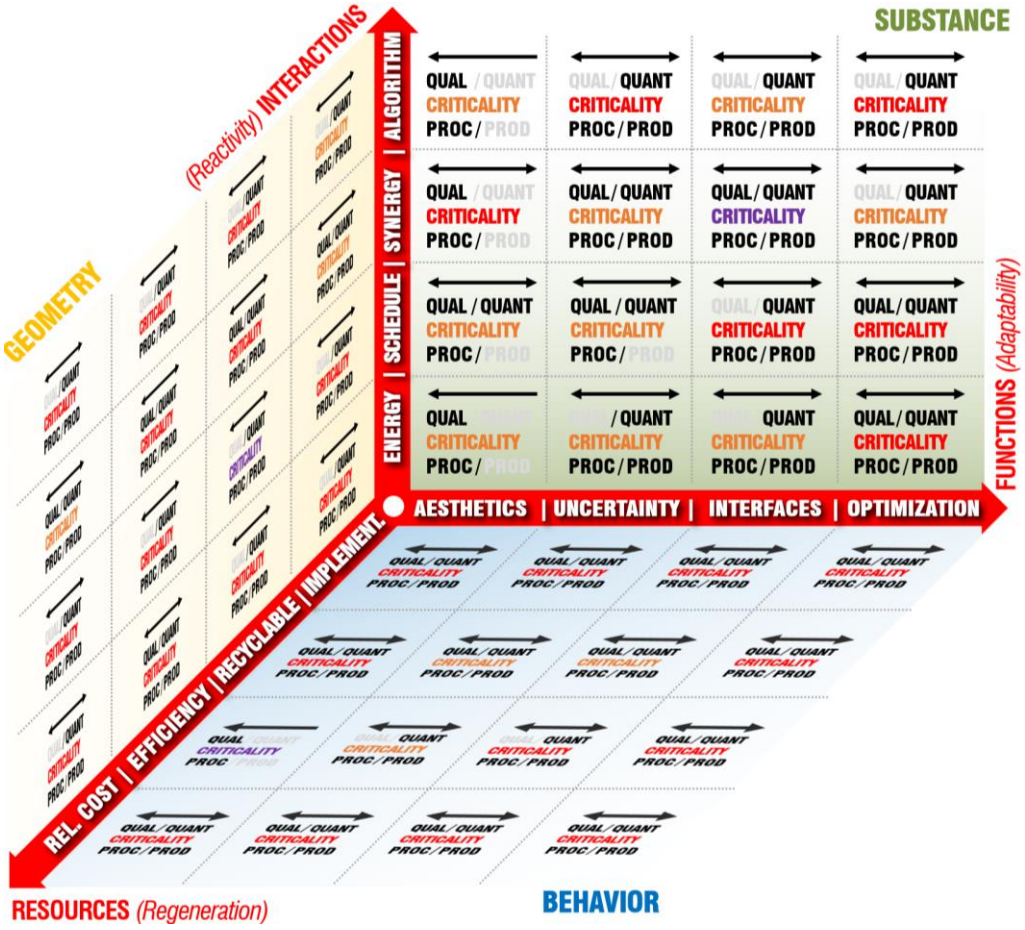



*Figure 124. Relationship across evolutive design drivers from a geometry, behavior, and substance standpoint.*

These multiple drivers are organized across the three upper faces of the evolutive design tetrahedron, representing geometry, behavior, and substance. Among them we could see internal relationships summarized in Figure 124. These relationships within an evolutive architecture are not fixed and they should be understood under both statistical and ad-hoc perspectives following the continuous nature of this approach. These graphics shows the relationship between them. The arrow shows if the relationship is bidirectional or mainly one way, and the point from the main driver to the subject driver. The reading order is from right to left. The relationship could be only quantifiable (quant), qualifiable (qual), or both (qual / quant). The criticality of such relationship could be low (purple), medium (orange), or high (red). These initial relationships among design drivers should be considered as adaptable tools to help the development process, and most importantly to enable better system solutions from a holistic, feasible, detailed, and evolutive perspectives.

## 4.5. Complexity as Integration

As previous sections have elaborated, an evolutive system architecture presents three major principles addressing context, design, and environmental stressors which are **adaptability, reactivity, and regeneration** as previous sections elaborated. While these have multiple design drivers connecting them at multiple levels, they also provide a three-dimensional coordinate system for complexity in the context of both evolutive and complex systems. Under that reference, complexity could be understood as the integration of all three providing a map towards design objectives and methodologies, as well as a performance measurement towards comparisons. Figure 125 exemplifies this reference system presenting two extremes from the worst to the best evolutive system where A is an unadaptable, passive, and depleting system, and B is a highly evolutive, reactive, and regenerative system. The curve between both extremes exists within a 3D surface created by all possible solutions. This curve or line represents the **inherent complexity** of the system and a direction for a system **evolutive optimization**. For instance, a solid and passive brick could require large amount of energy to be manufactured, with very low-tech, and a limited adaptability beyond its spatial positioning. On the other extreme, we could envision a multifunctional construction block with many configuration options that manages air and hygroscopic flow, uses recyclable materials with very low energy consumption, reduces mass, and collects solar power through integrated external photovoltaic cells. Both are valid solutions, but how to make the second a more capable solution that is easier to design, implement, and manage is the objective of this research.

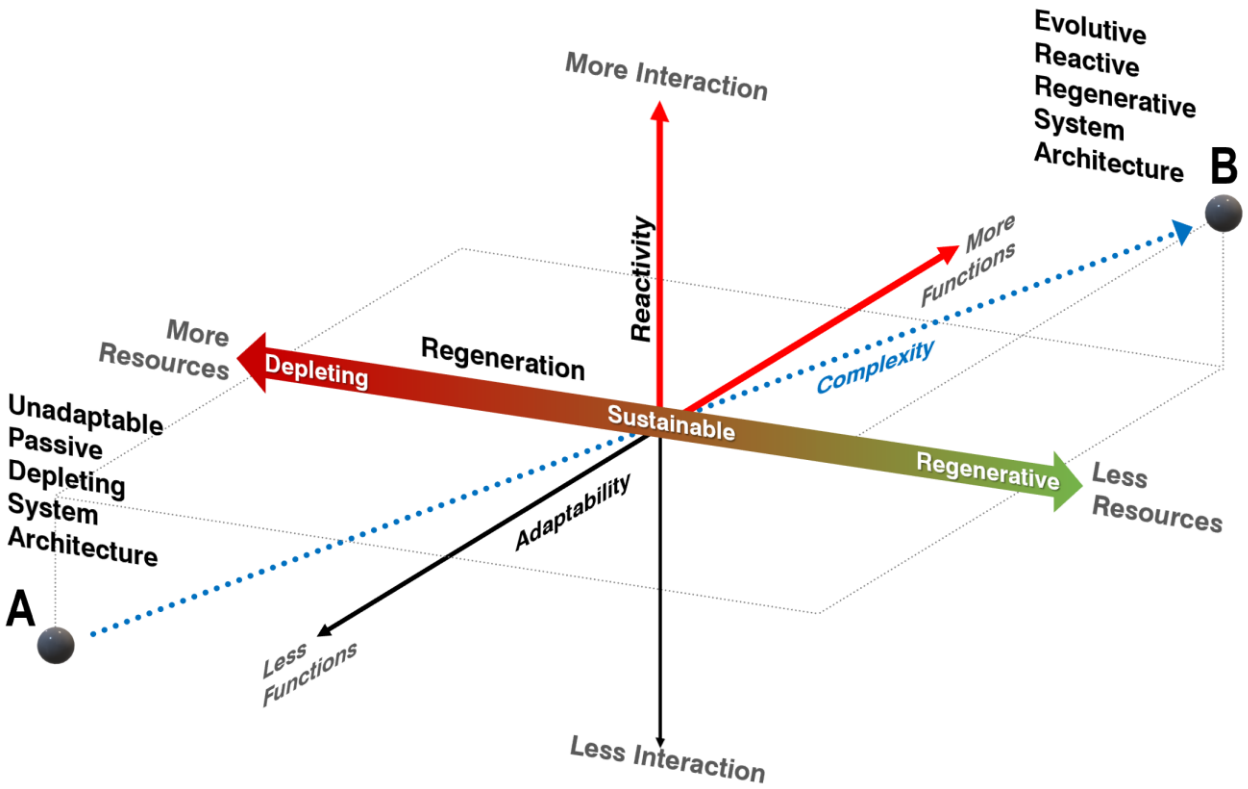



*Figure 125. Evolutive three dimensional reference framework with adaptability, reactivity, and regeneration as coordinates.*

## 4.6. Conclusion

In response to chapter 2, section 4.1 presented new stressors affecting the practice of systems design engineering in the upcoming decades, and the subsequent needs to manage more complexity in such endeavors. Studying the drivers behind this new reality, as well as deeply reviewing the state-of-the-art techniques from both engineering design and systems engineering, there are several gaps that define a new and growing subset of complex systems by themselves. They are: [1] the relevance of geometry as the common ground between disciplines which condition the capability of the system, [2] the consideration of fundamental functions or basic 'behaviors' of the system at hand that is no longer static or incapable of adaptations, and [3] the importance of addressing the need and use of all the resources required to produce, use, manage, and repurpose the system. These gaps not only constrain the system outcome but also the methodology itself. In other words, new needs required new methods and new standards. This is the starting point of the evolutive approach, addressing the need of adaptability in the system design, as well as the complementary and evolutionary nature of nature-inspired methods that help these challenges from a fast-paced, data-driven, self-organized, and multidisciplinary approach.

These points are often found and combined throughout the intuition and gut feeling of talented architects and chief engineers across multiple technical and artistic fields. Thus, this research aims to create a baseline approach to explore the full potential of such approaches to enable quantification, qualification, and more importantly optimization of new architectures, especially those hardware-based ones without heritage or previous generations.

This involves firstly studying first the special nature and characteristics of evolutive system architectures within the context of complex hardware-based architecture systems, and secondly how to develop a methodology to enable such system and compensate for current state-of-the-art techniques gaps.

Upon those increasingly present general stressors within scarcity scenarios described in chapter 2, and gaps in design methodologies the evolutive approach presents three constitutive keystones of basic principles: adaptability, reactivity, and regeneration. These were described in detail in section 4.2. They both characterize evolutive system architectures while also providing the foundation towards the subsequent design methodology. These keystones are intertwined (section 4.4) through a series of synergetic design drivers that map the full cycle of systems capable of reacting and adapting to any changes within their context and among components. These were described and grouped in section 4.3 around those three main principles. Furthermore, these drivers also address the use and management of all resources across all the design phases and lifecycle of the system. While this will be developed more in detailed in chapter 5, section 4.4 graphically presents in detail relationships across these drives within a three-dimensional reference system. Such a reference system is based on measuring functions, the use of resources, and interactions of the systems, in response to the three evolutive keystones, as well as those three areas describing any general system within this context: geometry, behavior and substance. Evolutive system architectures are physical, digital, virtual, or a combination of all of them. In essence they are highly adaptable, reactive, and sustainable or regenerative systems, as we could find among some robotic, architecture, aerospace, and organic systems.

In essence, evolutive systems are inspired by nature and they aim towards having the same level of performance, efficiency, and adaptation. Furthermore, the way this new class of architectures is conceived infuses basic principles proven through millennia of natural evolution on the planet. Simple but very powerful forces describe both system (product) and technique (methods) such as: [1] genetic and heritage information driving adaptability and selection, [2] multifunctional optimized implementations and designs, [3] a continuous approach towards the system always in constant change, [4] the relevance of the context or environment for the system design including cultural, technical, physical, digital virtual, etc. These

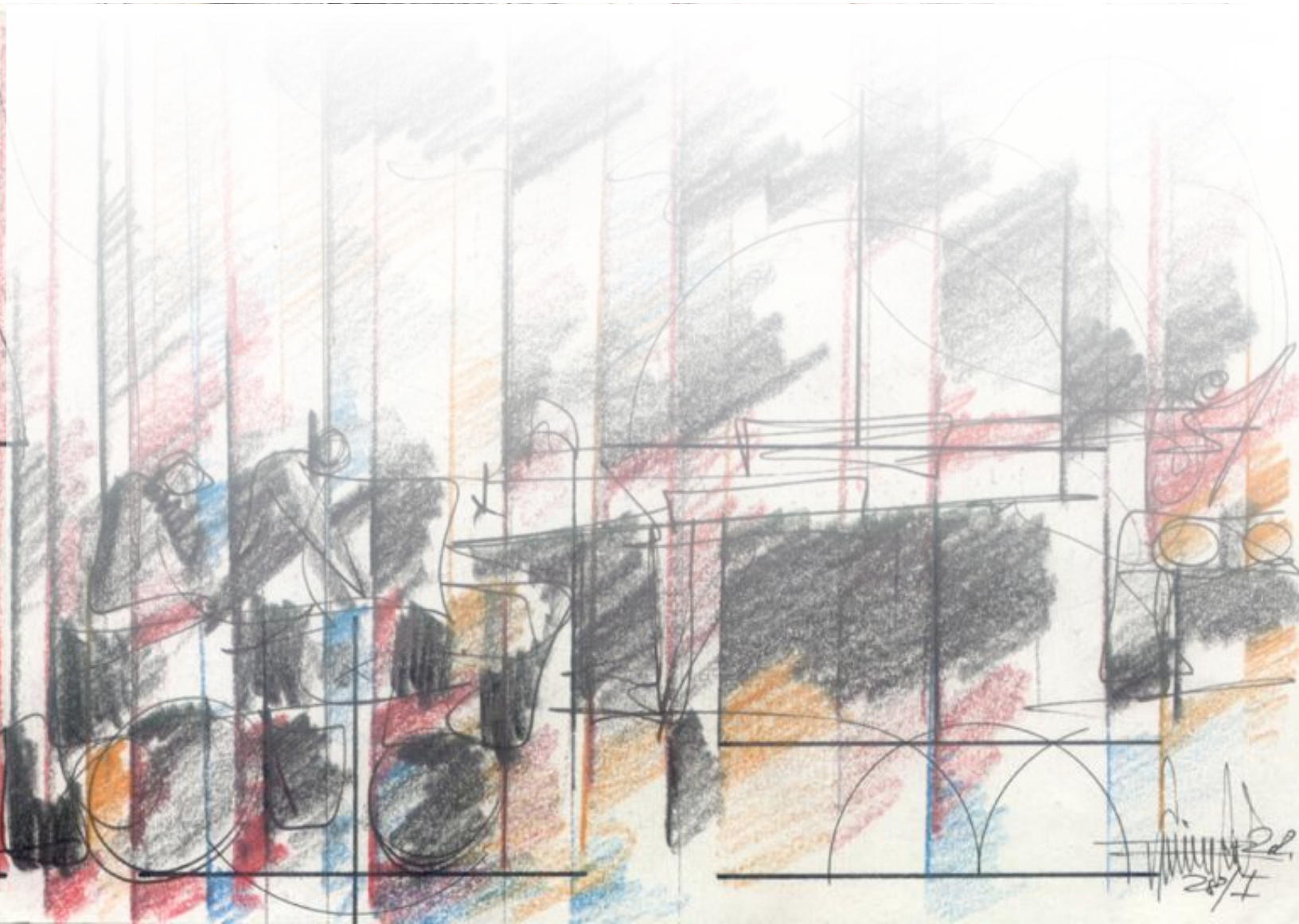
After this introductory description of evolutive system architectures, the following sections will elaborate on this research regarding techniques and methodologies that enable them (chapter 5), as well as a simplified example (chapter 6) that showcases both. While a key step in a networked process of development is the design phase, optimization and implementation phases are intimately related and will also be briefly introduced as perspectives behind this new way of looking a hardware-based system shadowing the ultimate system design, natural life.

# EVOLUTIVE SYSTEMS DESIGN

## Evolutive System Architecture Methodology

## CHAPTER 5

*"Computers are useless. They can only give you answers."*
***Pablo Picasso***



# 5. Evolutive System Architecture Design Methodology

Evolutive system architectures as described previously in chapter 4 are a subset of complex systems responding to multiple external design stressors based upon the key principles of **adaptability, reactivity, and regeneration**. Those principles are founded on **adaptive and evolutionary** approaches, and they map some key gaps in current state-of-the-art systems engineering and system design techniques. In essence, the tool conditions the outcome and the way to tackle a challenge. However, to make the most of those gaps (sections 3.4, and 4.1), as well as to provide a more efficient way to develop evolutive architectures, a subsequent methodology needs to be created. This chapter will present an approach towards such a process developed within this research, the **evolutive system architecture design** (eSARD).

This method is neither closed nor rigid. It introduces a foundational path that could and should be expanded and tuned for any especial needs required by designers, teams, machines, workflows, sectors, and industrial fields, among many others. Thus, an evolutive design approach should be applicable to any system design architecture development, independently from the field of application. This method presents the following general and interrelated goals:

- To develop an effective **design engineering method** that delivers mature evolutive system architectures without heritage, covers the full design lifecycle, optimizes time and resources, and enables the possibility for quantum-leap solutions. In other words, it aims at a leaner way for ground-breaking solutions with no heritage.
- To draft the foundation for a **SE system approach** that serves also as design methodology (DSE), and towards further infusions of computer-aid methodologies enhanced by data-driven methods (e.g., AI workflows).
- To also create the foundation for an **organizational and managerial scheme**, serving both DE and SE approaches to handle schedule, resources, and workforce, as well as any required technology and machine support.

The next sections will present in detail the development of this method through its objectives, principles, foundation, workflows, tools, and environments. However, this research is concentrated on the design and systems engineering foundational part. Thus, it only presents basic pointers towards the optimization and implementation aspects of SE applications, and other organizational and managerial portions within the full evolutive methodology ecosystem.

## 5.1. Applied Evolutionary Process

While the universe tends towards chaos (increasing entropy), natural evolution tends towards a greater self-order (decreasing entropy). Therefore in that process evolution increases and manages complexity (Brooks et al., 1988). Under this perspective, a living organism could be understood as a complex system architecture. As such, it is a member of a species, so it is part of a continuous series of similar architectures (Figure 126) as section 3.3 presented. Its adaptation is provided by a range of mechanisms leading to the survivability of such system, and it includes genetic changes that provide advantages (and disadvantages) against changes, development aspects, and external environmental factors.

Those evolutionary changes are based on proven solutions, in the sense that any previous generation was capable of reproduction up to that point. So, a heritage solution is a proven solution that paves the ground for a new generation. But this parental base also enables modifications in the offspring capable of developing quantum leaps in terms of adaptability from an evolutive standpoint (Gennaro et al., 2011). This is also a key hypothesis for an evolutive approach as previous chapters presented (section 4.1), which following nature enables something new based upon validated solutions and paves the way towards reaching much better system performances organically. This has been developed in algorithmic and data science methods for dynamic systems (Dempsey et al., 2009) , complex systems engineering (Braha et al., 2006), self-organization methodologies (Vijver et al., 2013b) and optimization techniques (Zhang and Sanderson, 2009), such as differential evolution (DE).

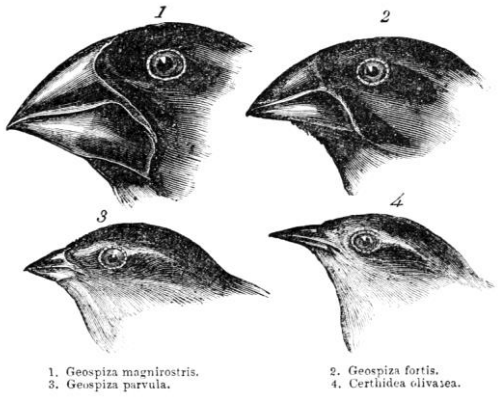



*Figure 126. Species characteristics. Engraving in 'Voyage of the Beagle (Darwin, 1845).*

However, natural biological evolution is also about 'hardware' since the end process is a physical biological system. There have been several applications of evolutionary computing methodologies to new manufacturing techniques, such as fabrication of advanced FPGA (Fernández et al., 2004) and topology optimized geometries (Chen and Hwang, 2009), among many more as presented in section 3.3. Nevertheless, most evolutionary methodologies are applied to the systems engineering side of a product design such as form assessment (Shieh et al., 2018), structure optimization (Ma et al., 2020), and control development systems engineering (Yan et al., 2011). However, these techniques always concentrate on specific SE or manufacturing aspects, highlighting an application gap towards a more system-level and holistic thinking towards complex system architectures. Similar multidisciplinary approaches are used in the architecture practice.

The evolutive methodology addresses that gap. It starts developing a foundational workflow towards a system-level thinking that considers the full cycle (design-implementation-operations) of a complex multidisciplinary architecture development, while being agnostic of tools, modeling techniques, and fields of application. Design principles, methodology steps, and overall phases are defined in the coming paragraphs and chapters through process definition and practical examples. Evolutive methodology is a system design engineering (DSE) approach born from an evolutionary methodology, adaptive principles, and architecture design mindset.

Within the broad spectrum of tools developed over the last decades with advancements in computer science, there is an area of special interest in software development, intelligent design (ID) and evolutionary computation (Ford et al., 2017). Instead of assuming that a preconceived code would address all possible scenarios, evolutive algorithmic techniques allow for the code to change and evolve, similarly to Nature's genetic evolution. An evolutionary approach assumes constant change, and with an integrated evaluation scheme, concepts, and solutions can be evolved or optimized (EO). Suddenly there are more than one valid solution, however this comes with the cost of a much more interrelated workflow, and the requirement of a much deeper knowledge of such algorithmic developments (Hingston et al., 2008). In essence, more flexibility in the process comes with more management challenges. However, the current software development ecosystem worldwide makes coding efforts a lot easier than decades ago. The same approach applies towards the study of complex adaptive systems (CAS) and self-organization (Georgiev et al., 2019) across many technical and scientific fields.

But what happens if such an evolutionary approach is applied towards hardware-based and software-enhanced system architecture designs? What kind of process would enable and streamline evolutive designs? Such process should address design engineering as well as systems engineering topics from an evolutionary engineering approach (Norman and Kuras on Braha et al., 2006), applying adaptive principles as described in section 4.1 to an open development process. This process would aim at complex systems, and especially at those in need of modernizing their design or requiring the infusion of new technologies. Complexity brings failures, delays, and budget overruns, among others because of the inherent nature of both such systems, but also because of the complexity of the multiple development processes required in that endeavor.

Under an applied evolutionary approach, rather than dividing a complex overarching system into smaller and manageable subsystems, the goal would be to improve the efficiency of the process by looking at the system holistically. The next step in such a process would be to address the design of the hardware-based architecture itself, even if it is the first of a kind and its requirements go beyond anything produced before. A design path that includes and combines implementation and optimization represents a gap in current methodologies as chapter 3 explored, and it is the objective of this research activity.

Nevertheless, a design project could become a never-ending story. The more complex the challenge becomes, the more difficult is to define when something is good enough, as experienced, and passionate designers know well. From a requirements standpoint on the other end it could be easier to assess if thresholds are met or not. However, along the way the discrete nature of requirements can miss the discovery of critical designs and optimization strategies based on connections and synergies between them. Nature does indeed manage synergy very well, as the ultimate efficiency tool.

This is especially relevant when the design process tackles something that has never been done before. Deepening in the design problematic, often reveals hidden connections and synergies that were not explored or known before. However, a fast process based in synergies could offer a better platform from which to make the most of inevitable iterative design phases. Certainly, the main characteristics of an evolutive architecture such as adaptability, reactivity, and regeneration are among those requiring interconnection among design variables and system design, as well changing complexity management.

## 5.2. Design Process Approach

The development of the eSARD approach starts with the evolutive design tetrahedron characterizing the evolutive architecture system (Figure 127). In addition to other general characteristics for complex systems, evolutive system architectures present three major characteristic principles or keystones as elaborated in chapter 4: **adaptability, regeneration, and reactivity** (ARR). However, those overall principles only describe high level architecture characteristics, so a design process needs to address all three system descriptive areas, such as **geometry, behavior, and substance (GBS)**. Finally, as a practical method eSARD also tackles the scale of all **design, implementation, and operations (DIO)** system details.

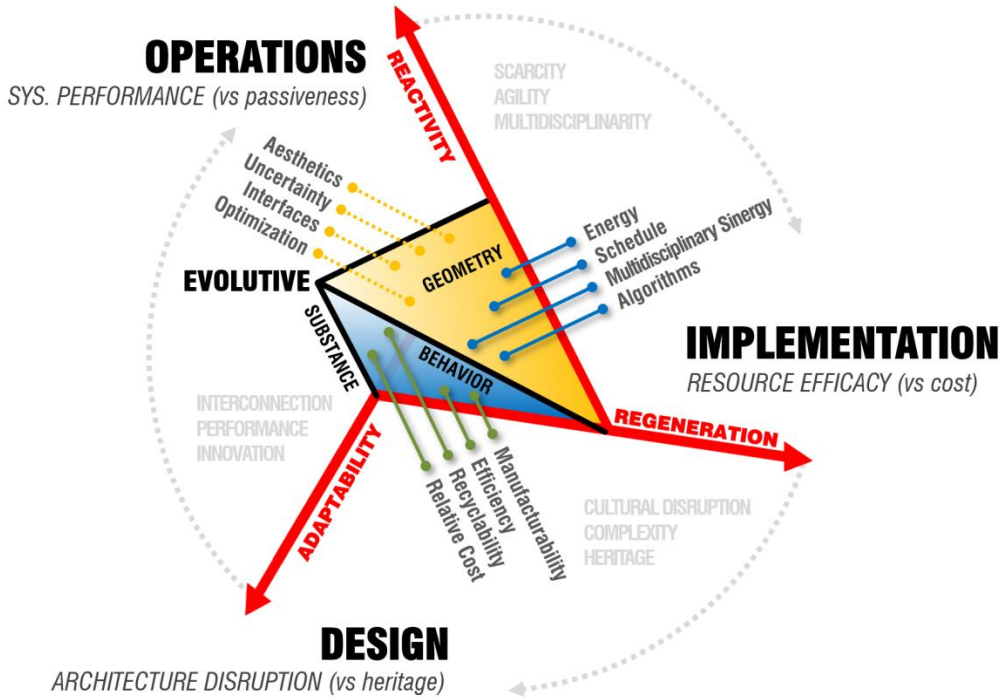



*Figure 127. Evolutive design tetrahedron defining key methodology phases such as design, implementation, and operations.*

As previous points have introduced, the development of complex systems architectures in the beginning of this century is conditioned by the potential growing scarcity of resources due to multiple factors and increasing levels of systems complexity. The balance between needs and resources is currently changing, often demanding the infusion and integration of new and disruptive toolsets that complement more traditional methods. From the standpoint of a near 4th industrial revolution (Machado and Davim, 2020) to new human-machine collaborative workflows (Daugherty and Wilson, 2018), everything points towards a change in the paradigm. Such transitions have been happening at much faster rates in the fields of software and computer systems than in hardware implementation developments (chapter 3). This is the context of this system design engineering research, which among others is addressing two critical gaps in the design of complex hardware-based architectures:

- How can we **design more efficiently towards optimization and the implementation** of better performance architectures presenting evolutive characteristics?
- How to tackle the **lack of heritage and increasing multidisciplinarity complexity** in such processes?

An evolutive system design approach starts with a full-cycle perspective, which tackles design, implementation, and operations simultaneously to enable **higher system performance and more efficient system-level architectures by building upon synergistic connections among disciplines and subsystems** (Figure 128). This evolutive methodology is especially useful when designing under a significant lack of heritage (first-of-a-kind), time constraints, as well as a broad spectrum of feasible and yet new subsystems or technologies that must be infused for the first time.

While chapter 3 identified critical gaps in current state-of-the-art DE and SE techniques, chapter 4 highlighted and presented in detail the characterization of evolutive architecture systems. Thus, this thesis tackles gaps and characteristics showcased by those architecture design principles (the what), while it develops a methodology around it (the how). Inspired by nature design methodology (Kliman, 2016) and the more holistic or multidisciplinary practice of architecture (Jarzombek and Prakash, 2011), this approach applies proven and even ancient methods to new implementation fields


*Figure 128. eSARD evolutive design and systems engineering approach scheme.*

From a methods standpoint, this approach applied some aspects of evolutionary systems engineering (Braha et al., 2006) in computer science to the realm of hardware-based implementations. As such, rather than linear and monodisciplinary or even parallel methods, this approach has a **network-driven** scheme, embracing and combining both **concurrent** and **collaborative** engineering practices to the extreme. Furthermore, this methodology is not only about **quantifiable disciplines** (e.g., mechanical design) supported by analytic parameters, but also **only-qualifiable** subjects (e.g., aesthetics) based on geometrical design, as well as open or **changing requirements** workflows.

Under this light, an evolutive approach (eSARD) does not concentrate on single point-design solutions. Rather it tackles the system architecture development from a **continuous solution** scheme, while addressing further **optimization, implementation (including management), and operations** from a **geometry, behavioral (functions), and substance (resources)** perspectives (GBS). This is agnostic of both applications and tools, and the approach also aims to **infuse higher levels of adaptability in the methodology** itself from both design (geometry) and SE (abstract) perspectives.

From a product, artifact, and system architecture perspective, an evolutive system design process aims in the end to produce an implementable evolutive system architecture. This presents several complementary characteristics when compared to more traditional hardware-based systems such as: **[1] high system adaptability, [2] an intelligent reactive baseline, and [3] a regenerative or sustainable resources strategy.** From concept to implementation, the evolutive approach tackles maturity gaps within the system and its parts. Then it builds upon commonalities and synergies among disciplines, subsystems, and stakeholders. The feasibility and functional capability of the architecture at hand drive the approach, while always keeping in mind the overall efficiency in terms of resources, agility, and adaptability.

This methodology fills the gaps in currently applied design engineering (Pahl et al., 2007), and systems engineering (INCOSE, 2020b) techniques for physical and hardware-based systems. In essence, it also enables a novel theoretical foundation to **do better with less** as its key design philosophy principle. Hence, the development of this process is based upon: [1] extensive literature reviews, [2] research, prototyping, and hands-on activity, and finally [3] several decades of validated professional experience as an architect and system architect across multiple industrial fields worldwide, including almost a decade of practice at the NASA Jet Propulsion Laboratory developing complex system architectures. Nevertheless, this approach is developed from a fundamental research perspective, so it is completely agnostic of tools, the field of applications, and any specific technology. In summary this aims to be a universal approach towards system design (DSE).

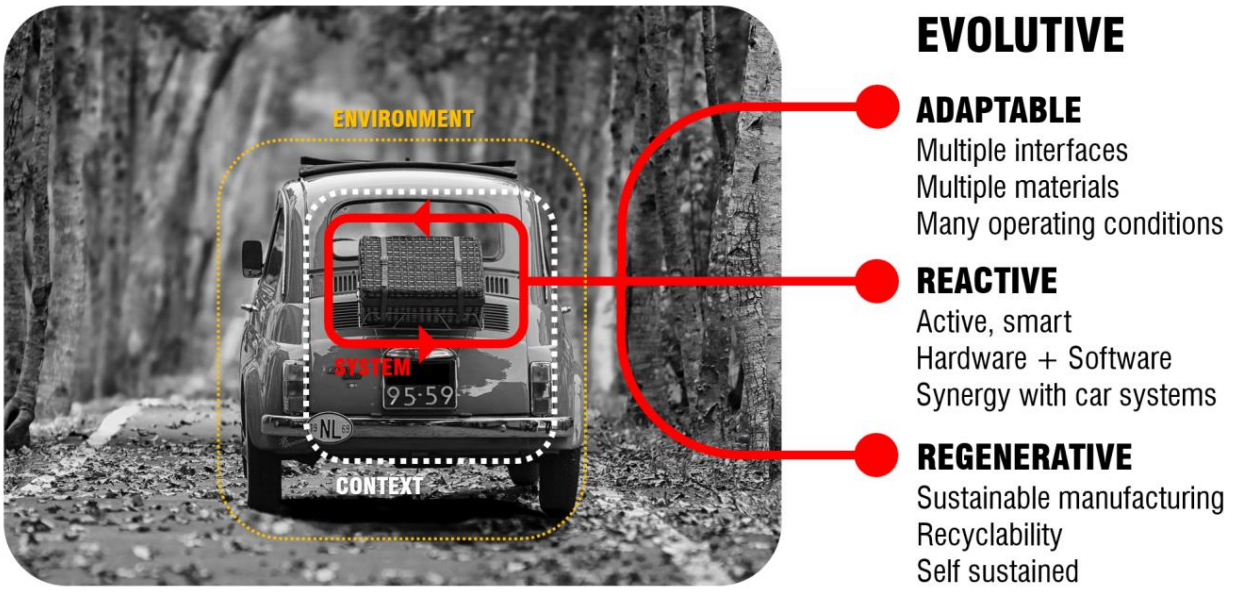



*Figure 129. Examples of eSARD approach applied to an add-on component for an existing car design.*

The development of an external add-on part for the body of an existing car design could be a good example of this approach. For instance, this could be for instance a luggage support add-on for the trunk (Figure 129). The part itself does not have much heritage since is quite unique and not a part of the original design. However, it requires to increase its performance when compared to previous solutions due to increases in speed tolerance, comfort standards, and environmental protection. Furthermore, an evolutive approach applied to this problem would consider the following points:

- **Adaptability** (geometry). The component should adapt to different driving parameters, environmental conditions, and mechanical interfaces passively. Multiple finishing and material options would be part of the trade space.
- **Reactivity** (behavior). This aspect could enable lighting and active aerodynamical control. It should also be trackable with GPS if gets lost, stolen, or falls off the car. Thus, batteries, sensors, and active components are integrated.
- **Regeneration** (substance). Both manufacturing and system complete lifecycle should fully sustainable.

The company developing this part could only be interested in addressing requirements for a specific model. However, when an evolutive approach is applied, both designer and design workflow should adapt for more. Thus, rather than its design being solely applicable to a single case, it is done considering many other likely or feasible possible constraints to find more synergetic and optimized solutions. It is not about over-constraining; it is about stressing towards a better solution.

## 5.3. ARR Development Areas

An eSARD process tackles all three ARR areas (adaptability, reactivity, and regeneration) from a networked perspective by addressing in detail design, implementation, and operations (DIO) of any system, as Figure 130 shows.

The following sections will elaborate the overall design approach of this methodology, which is the main objective of this research. Furthermore, sections 5.3.2 and 5.3.3 present the general perspectives for complementary optimization and implementation efforts (including management) that are also required. These considerations happen concurrently to the design activity, and as soon as possible in the process. However, the complexity and depth required for their full development will be part of future research activities and publications. All these aspects have been researched and tested by the author on different professional practices.

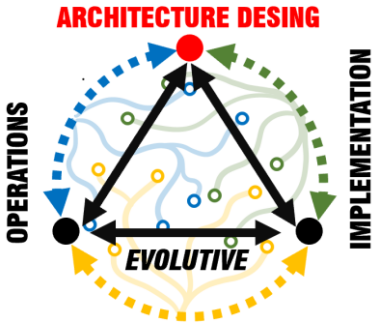



*Figure 130. eSARD networked process.*

### 5.3.1. eSARD System Design (Geometry)

A critical node within the eSARD evolutive design process is to create an initial reference system or seed architecture, so an iterative and concurrent design process can be developed around it. In the case of hardware-based system architectures, that seed system description also includes a descriptive geometry, or in other words a volumetric, morphological, and conceptual description. Such a geometrical definition often starts with a sketch and evolves to a fully detailed CAD/CAM model including parametrized design variables, and sensitivity studies, and among other aspects an applied e-design paradigm (Chang, 2015).

However, in this continuous evolutive process such drawings or models are not static since they constantly change both conceptually and functionally. Regardless the tool being used, in the mind of the designer and at the core of workflow this representation should be considered more of an animated cartoon or a video, rather than a static picture or CAD model. Thus, such conceptual representation is always a snapshot in time which gradually gains more details and definitions of key data points from other activity nodes in the design network such as optimization and implementation (Figure 131). This design seed will always change.

There is not a beginning or end within this approach since a complete system architecture requires all nodes to be defined enough to be complete and feasible. The design process conceptually never ends, and it is meant to be capable of continuing. However, such a process considers the evolution and adaptability of the system, therefore its behavior and implementation are also areas for optimization, scalability, and upgradeability even if they are not initial requirements. This design development process presents several key features such as:

- **Fast**. By default, this is a fast-paced process to make an efficient use of time and other valuable resources, as well as to improve easiness.
- **Easy.** This process should enable seamless synergistic efforts among disciplines, workforce, infrastructure, and schedules considering multiple phases in the lifecycle (including prototyping).
- **Disruptive**. The ultimate goal of eSARD processes is to go beyond previous solutions performance and capabilities (if they exist).
- **Stressing**. It is critical in this approach to stress the design process by scouting connections among initial requirements and disciplines, while considering other complementary ones across phases.

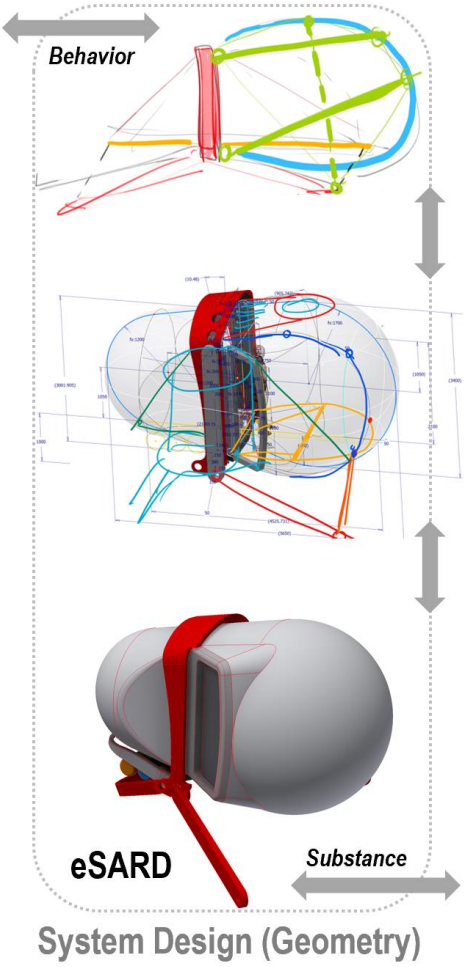



*Figure 131.System architecture geometrical seed, presented as a captured instance within an evolutive design process example.*
*© 2021 Raul Polit Casillas*

The geometrical aspect does not necessarily mean the system needs to be physical. Logical and non-geometrical systems (e.g., software-based) can be addressed within this approach as well. In that case, geometry refers to the conceptualization and logical structure of the system. For instance this could be a system model diagram (Friedenthal et al., 2008) such as data flow, which offers a non-geometrical view of the model created for a system. This initial seed geometrical design (Figure 131) will continue through a process of refinement and detailing, which addresses other foundational aspects related to the behavior and implementation of the system. In other words, we need to know what we draw/model and why.

In a networked process, this all happens concurrently, so all methods should be allowed to explore, capture, and share any knowledge or experience. This aspect is critical as enabling a good communication among all actors in the process, human or otherwise, leads to faster, better, and more efficient efforts in creating this seed geometry.

#### *5.3.1.1. Inputs*

There are multiple inputs required for this design node within an eSARD process. Table 22 presents a summary of inputs, processes, and outputs. Among some of the principal inputs are the following:

- **Requirements**. These could be only quantifiable, qualifiable, or both. They could also be open in nature (e.g., as light weight as possible) or closed (specific value). While there are key design requirements a system architecture design must meet, there are also priorities based on importance and criticality. Thus, requirements could be essential, or desirable, and it is critical for the design process to study and strategize them accordingly. Furthermore, an eSARD process will developed secondary requirements during the process to stress the design, to better explore the trade space of options better, and to address complementary implementation and operations needs that might not be foreseen or included at the start.
- **Constraints**. These limit the design process unless countermeasures can be taken. They include limitations from the standpoint of product, process, and operations. Their nature varies and includes topics such as design, interfaces, assembly, heritage, culture, manufacturing, budget, schedule, and maturity needs, among many other limitations.
- **Drivers**. These are not really an input per se, but rather inherent characteristics of the design process that could influence both designers and workflows. Some of the most relevant were described in section 4.3 (system) and will be elaborated further in sections 5.4 and 5.5.

#### *5.3.1.2. Processes*

This design process will be developed in detail in the following sections. From section 5.7 and on, all details regarding phases, steps, and techniques used within this research will be laid out. Chapter 6 presents a simplified example as well.

#### *5.3.1.3. Outputs*

The product of this initial process is a system architecture seed, a fundamental geometrical and system definition of the system being developed. This [1] addresses the most important gaps at both system and subsystem levels, [2] considers design, implementation, and operations, and finally [3] becomes the foundation for subsequent iterative cycles to bring more details and evaluate alternatives. Such an evolutive seed not only entails a system visualization but also capturing all associated knowledge used to identify architecture maturity gaps (AMGs) and develop systems engineering modeling. These will be elaborated in detail in section 5.9. These outputs could then be digital, virtual, physical, and data based.

Exploring any design challenge at hand through a series of facilitated questions allows one to identify hidden relationships between subsystems and discipline requirements. AMGs are the most critical of the key relationships defining both feasibility and system performance. Upon such gaps, the design (geometry) and system modeling (abstract) processes develop the foundation for further cycles that will increase the maturity and complete the system design. Following points will elaborate and exemplified this phase in the evolutive methodology that is summarized in the following table.

| **INPUTS** | *Description* | **PROCESSES** | *Description* | **OUTPUTS** |
|---|---|---|---|---|
| **ARR Drivers** | *Driving Process / Product* | • eADQN, eAMG, eASG<br>• Geometry sketches & refinement processes<br>• Volumetry & packaging<br>• System definition<br>• Rapid analysis<br>• SE modeling<br>• DE modeling<br>• Styling & customization<br>• PR modeling<br>• Feasibility<br>• CAD/FEA/CAM/BIM<br>• Rapid prototyping<br>• Visualization | *Digital, physical, virtual* | **Geometry** |
| | Adaptability-driven (design) | | | Diagrams, sketches, 3D models, 4D models |
| | *Reactivity-driven (system)* | | *Digital, logical* | **Systems Description** |
| | *Regeneration-driven (resources)* | | *Logical, digital* | **Basic Analysis** |
| **Requirements** | *Driving product* | | *Logical* | **New requirements** |
| | Quantifiable, qualifiable, both | | *Digital, physical, virtual* | **Interfaces** |
| | Primary (client-driven) | | *Digital, physical, virtual* | **Styles** |
| | Secondary (eSARD-driven) | | *Logical* | **Equipment lists** |
| **Constraints** | *Multiple types* | | *Digital, physical, virtual* | **Visualizations** |
| | Product outcome, design process, operations | | *Digital, physical, virtual* | **Rapid Prototypes** |

*Table 22. eSARD system design development inputs, process, and outputs.*

### 5.3.2. eSARD Operative Optimization (Behavior)

Understanding design needs and the complexity behind the system architecture leads to a feasible design path and an evolutive seed geometry. However, this is not enough to complete the maturation of such system. Behavioral considerations are also key complement, becoming the base for the second node within this process, operative optimization (Figure 132). Although this process might not start necessarily by creating a geometry, this helps as an initial input in the process. System behavior aspects are maturity gaps being addressed within this node. Like in any other system architecture, (Pollio, 2018) geometry (design), behavior (performance), and substance (resources) are both related and needed. This operative node depends heavily on the reactivity of the system, and it is more prone towards systems engineering and analysis within the development process while addressing two main areas such as:

- **Operations.** Any system is designed to be used and operated affecting every aspect from system performance to supply chain (Mahadevan, 2010). Understanding operational needs and bring them upfront in the design process (networked workflow) is critical for an evolutive system architecture that addresses the sustainability of resources, the system adaptability, and other subsequent smart management topics. This area includes among other ops-con studies, system functional behaviors, prototyping, functional analysis, model refinement and analysis, etc.
- **Optimization.** The operative side of a system development, as well as its multidisciplinarity requirements bring the need to optimize such system architecture from that perspective. This optimization can happen at many levels such as discreet (e.g., topology optimization to lightweight an structure), multi-objective (Abraham et al., 2005), and multi-criterion (Takahashi et al., 2011) using evolutionary data science techniques. Within this node, optimization parameters are driven by system operations, which include all behaviors and functions the system architecture is capable to perform. Thus, this activity is based on the system design, adaptability, resources management (regeneration).

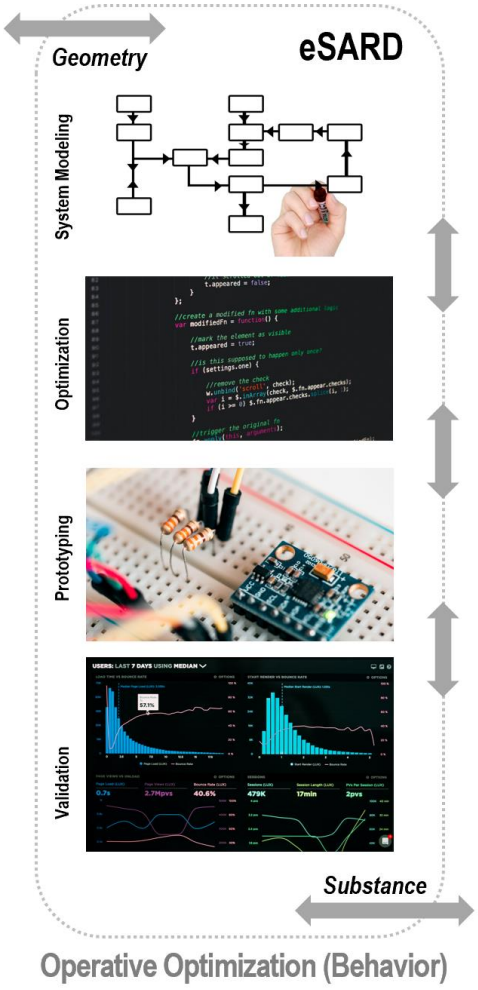



*Figure 132. Operative optimization design node within the evolutive design methodology.*

As previously described in section 5.3.1 the foundation of this operations node within an eSARD process presents a series of basic inputs, processes, and outputs. The following sections briefly develop all these areas.

#### *5.3.2.1. Inputs*

Inputs for this operational node come bidirectionally from both design and implementation activities within an eSARD process. All three are needed almost in all cases, so no input is more important than the other. Such inputs evolve and get more defined along the maturation process. Therefore, these are some of the most critical inputs in this phase:

- **Systems behavior and performance requirements** (ARR). These include descriptions, variables, models, and measurements related to areas such as, system performance levels, scenario characteristic, autonomy requirements, interactivity, environmental constraints, performance curves, validation codes, protocols, regulatory limitations, etc.
- **Overall behavior constraints**. Among others these include power limitations, software restrictions, coding limitations, heritage, culture constraints, translation limitations, sensor capability, onboard computing, etc.
- General **eSAR architecture design and eSARD process drivers**. See section 4.3, 4.4, 5.4, and 5.5.

#### 5.3.2.2. Processes

Following a similar approach to the design node, the operative optimization node requires a series of interconnected steps and products, which are linked to both design and implementation activities. These steps integrate critical areas such as system behaviors, operations, and ARR aspects. These include among others the following:

- **System and Operations Modeling.** The seed geometry within an evolutive architecture represents and includes all volumes, interfaces, styles, and adaptability considerations. The concept of adaptability is directly connected to the reaction of the system, which requires the modeling of complementary systems reflecting non-geometrical characteristics such as behavioral ops-con, structural analysis, analytic requirements, and parametrics, among many more. Such system modeling can be done using multiple tools and methods (e.g., SysML™, UML, manual diagrams, etc.). However, under an evolutive paradigm this process will continuously be changing based on the feedback from both design and implementation nodes. Furthermore, this evolutive design seed model is centered in synergetic connections across requirements, subsystems, and disciplinary analysis, which are also multidisciplinary in nature. The more gaps are studied and connected; the better modeling of the system can be done. This point includes key operational areas such as autonomy, data architecture, sensor architectures, ops-con studies, regulations, coding, and programing considerations, among others. So, from this perspective, a network approach can make a difference connecting cross-cutting areas across the lifecycle. Connection between design and operations become crucial affecting requirements, configuration, interfaces, design, analysis, verification, risk management, optimization, manufacturing, testing, integration, data and knowledge management, planning, reutilization, and decommission.
- **Optimization**. Once key primary requirements have been studied, and secondary requirements identified, subsequent design iterations can explore feasible design strategies, where both geometry and systems behavior are defined properly. This is critical towards improving system performance and assessing solutions beyond existing heritage. This is also the starting point for an optimization process that will continue during the evolutive process. Such process is directly linked to an initial analysis and evaluation of the system activity within its context. There are two objectives within this step: [1] to find better and more optimum solutions that consider key synergetic connections highlighted by the eSARD process, and [2] to provide an assessment of the system performance and closeness. If a system is fully defined by an initial and feasible solution assessing all ARR areas, it can be evaluated and redone.
- **Prototyping**. There are aspects of a system architecture, especially when complexity is high, that cannot be modelled or predicted. Thus, prototyping is as critical as other system functional demonstrations such as behaviors, interactions, and other complex functional topics. This is not only a concurrent tool towards the validation of the system, but also a design tool to discover critical maturation gaps that are not possible to identify otherwise. Prototyping under this approach also becomes a quality control tool for the whole process affecting design, operations, and management. This activity can be physical (e.g., 3d printing, bread boards, COTS - Macdonald et al., 2014), virtual (e.g., computer simulations - Cooper, 2001), behavioral (e.g., user study case), or a hybrid of them.
- **Validation**. The system operational approach and related design paths are continuously detailed and assessed upon the optimization and development process. As a continuation of all prototyping activities, this step has multiple implementation paths. Assessing the validation of the system not only considers subsystem feasibility and other general assumptions, but its whole from a design, operative, and implementation standpoints. Maturity levels are then used to assess system completeness, similarly to aerospace concept maturity levels (CML, Wessen et al., 2013) and technology readiness levels (TRL, NASA, 2016). However, within an eSARD approach, this point also addresses other critical areas across the system lifecycle that are linked to both design and implementation topics.

#### 5.3.2.3. Outputs

Within such networked and concurrent eSARD process, all outputs of the operative node are interconnected with other areas of the process. Furthermore, data coming from instrumented real systems is integrated to optimize future outputs and system optimizations under some design trends. In essence, future systems design and operational optimizations are data-based. New tools such as physics based visualizations (Plowman, 2019) are also changing the way operations and system behaviors can be studied. Table 23 presents a summary including inputs, processes, and outputs for this critical eSARD node that tackles basically all system functional behaviors and optimizations.

| INPUTS | *Description* | PROCESSES | *Description* | OUTPUTS |
|---|---|---|---|---|
| **ARR Drivers** | *Driving process / Product* | • eADQN, eAMG, eASG<br>• eASMs<br>• Operation analysis<br>• Ops-con<br>• Ops. architecture<br>• System definition<br>• Analysis<br>• Simulations<br>• Functional prototyping<br>• Process visualization<br>• Others | *Digital, logical, virtual* | **Sys. Behavior** |
| | Reactivity-driven (system) | | | Diagrams, timetables… |
| | *Adaptability-driven (design)* | | *Digital, physical, virtual* | **Ops. Architecture** |
| | *Regeneration-driven (resources)* | | *Logical* | **Data Architecture** |
| **Requirements** | *Driving product* | | *Digital, physical, virtual* | **Ops. Visualizations** |
| | System behavior-driven | | *Digital, logical, physical, virtual* | **Functional prototypes** |
| | Ops-con-driven | | *Digital, logical, virtual* | **Virtual studies** |
| | Regulatory / culture-driven | | | |
| | Primary (client-driven) | | | |
| | Secondary (eSARD-driven) | | | |
| **Constraints** | *Multiple types, limiting:* | | | |
| | System functions<br>Heritage / culture<br>Design process<br>Operational architecture | | | |

*Table 23. eSARD operative optimization development inputs, process, and outputs.*

### 5.3.3. eSARD Implementation (Substance)

Designing, optimizing, and prototyping is not enough for a complete system architecture design. Thus, the implementation of the system is part of an eSARD approach. This design node is directly related to the regeneration principle pointing towards the use and management of resources (substance). Thus, it entails materialization, manufacturing, resource management, and resource optimization including recycling, repurposing, and regeneration (see Figure 133). The evolutive approach towards systems engineering tackles both method and products, thus it considers system feasibility and resources management of (e.g., workforce, time, tools, etc.) towards implementation, operations, and design of the system. It also includes managerial and programmatic aspects, which are also required to set up an evolutive design workflows and obtain evolutive system architecture designs more efficiently. This implementation node (Figure 130) goes beyond fabrication and includes systems integration and delivery as well. The objective of defining, designing, analyzing, and modeling a system has been already addressed on previous chapters, however implementing such system architecture, and managing all required resources must be part of the development process too. Therefore, this concurrent part of the method is about the substance of the system, a concept that includes materials, workforce, coding, computing power, energy, and other natural, human, and technical resources. Hence, this is about managing, optimizing, and improving the use of those resources [1] across the system lifecycle, [2] its development process, and [3] all system operations within organizations (new, seasoned, or virtual), teams, and professionals.

This node is especially important when resource scarcity, system performance, and heritage are key design stressors for the evolutive system architecture. This effort is also critical when the need of a quantum leap from any previous heritage solution forces to rethink the approach affecting all the way to how the manufacturing of the system is done (Leondes, 2019).

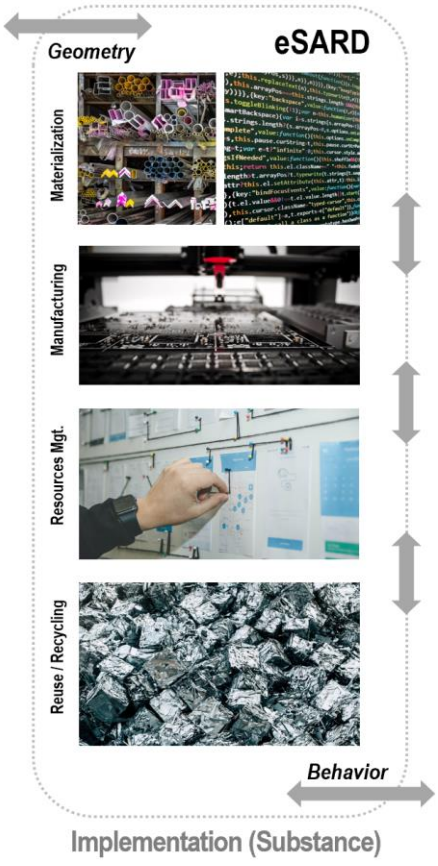



*Figure 133. Implementation node within the evolutive system design methodology.*

#### 5.3.3.1. Inputs

The outputs of the design and operative nodes are the inputs for this node. These also include all relevant information regarding resource utilization, optimization, management, utilization, and implementation techniques (e.g., building, manufacturing, programming, coding, fabrication, etc.) regardless of their human-driven or machine-driven nature.

#### 5.3.3.2. Processes

Multiple interconnected steps and fields are part of this evolutive node tackling adaptability, reactivity, and especially resource regeneration and management (ARR). Among others these processes include the following:

- **Materialization**. Implementing a physical system requires materials, the same way an algorithm-based system needs coding and data. Hence, this point refers to all resources required to turn a system architecture into a physical, logical, or virtual reality. The selection of the right substance is critical and influences manufacturing feasibility, as well as energy assessment, thermal properties, design effort, and analysis, among many more. Similarly, the selection of coding languages and data architectures directly affects cost, schedule, service (workforce), speed, etc.
- **Manufacturing**. If system design and materialization are defined, the next step in the use of those resources is to fabricate, manufacture, build, integrate, produce, code, and release such system architecture. This presents huge implications affecting workforce (Waldeck, 2014), cost, schedule, automation constraints (Wang et al., 2016), technology infusion, and system integration. The increasing infusion of disruptive techniques, such as additive manufacturing (Killi, 2017) is a good example, since their use can greatly conditioned both system design and optimization. Just like material and code aspects, the infusion of manufacturing constraints, requirements, and options from the very beginning not only presents a huge advantage for a concurrent cost, risk, and delivery schedule assessments, but also for future system upgrades and developments as part of family of solutions (species). For instance, once the design of a high-performance window is done, this step will ensure that future modifications can easily be implemented, allowing the repurpose of work and thus lowering the cost. On the other hand, if new manufacturing capabilities are available, it would also be easier to reassess the system architecture accordingly.
- **Resources management** refers to multiple managerial aspects regarding the use of resources during the design, operations, and implementation of the system. For instance, assessing a system implementation from an energy standpoint evaluates the energy: [1] required for system operations, [2] used to produce the system, and [3] utilized to design and optimize the system (e.g., computing power and workforce). This step is directly related to the 3C framework created for this networked activity and is critical towards introducing this methodology in any organization. Both traditional frameworks that are more driven by heritage (slow pace) and innovative clusters being developed along the way (fast pace) could use this methodology if done properly. The management of resources is a foundational characteristic of an evolutive regenerative architecture development.
- **Sustainability (reuse / replenishment / recycling / decommission).** Finally, as a close-cycle economy effort, the evolutive approach tackles the sustainability of a system architecture within its context, as well as other related design methodology efforts. This area is connected to the regenerative evolutive principle within the ARR approach. This is especially relevant under current tendencies of product and service-driven systems (Ceschin, 2013) that require more innovative solutions faster, while influencing their corporate organizations in the process (Kao, 2010). Here several areas are tackled simultaneously including energy, natural resources, environmental footprints, workforce capabilities, knowledge management, data management, system repurposing, and recycling strategy, among many more. In essence, this step is organized in three large areas affecting both systems and processes, such as [1] reusing or repurposing systems, components, and work efforts, [2] replenishing directly or indirectly all used resources by the system, [3] recycling resources, system components, and data, and lastly [3] full system decommission. Certainly, these topics also relate to the operative side of the system, affecting its optimization across its lifecycle.

#### 5.3.3.3. Outputs

This phase presents multiple outputs that require further development. In general, these serve as inputs for other nodes while they tackle all key implementation areas of physical, logical, digital, and virtual systems including trade studies, utilization schemes, implementation plans, tests procedures, cost studies, and organizational schemes, among many more.

The products of this node could be organized across design, development, and system operations around these areas:

- **Using resources**. This includes all necessary products and processes managing the implementing of a system, such as material studies, manufacturing studies, equipment trade-offs, programming schedules, etc.
- **Managing resources**. These include products related to all system operations aspects.
- **Regenerating resources**. From system recycling studies to replenishing resources, these products tackle the full spectrum of analysis, schemes, and design to achieve feasible regenerative and sustainable systems.

Table 23 presents a summary of some inputs, process tools, and output products of this node. The goal is not necessarily to create a full development plan, but to enable a good system design that considers key characteristics for its future development, mass production, or even one-off production implementation plans.

| ***INPUTS*** | Description | ***PROCESSES*** | Description | ***OUTPUTS*** |
|---|---|---|---|---|
| ***ARR Drivers*** | *Driving process / Product* | • eADQN, eAMG, eASG, eASMs<br>• SE modeling<br>• DE modeling<br>• Feasibility<br>• Trade studies<br>• Analysis<br>• Simulations<br>• CAM<br>• Coding / Development<br>• Prototyping<br>• Sustainability studies<br>• Recycling studies<br>• Others | *Digital, physical, logical* | **Resource schemes** |
| | Regeneration-driven (resources) | | *Digital, physical, logical* | **Mfg. Schemes** |
| | *Reactivity-driven (system)* | | *Digital, physical* | **Mfg. Tests** |
| | *Adaptability-driven (design)* | | *Digital, physical, logical* | **Dev. Schemes** |
| ***Requirements*** | *Driving product / Process* | | *Digital, physical* | **Dev. Tests** |
| | Quantifiable, qualifiable, both | | *Digital, physical, logical* | **Org. Schemes** |
| | Primary (client-driven) | | *Logical* | **Cost analysis** |
| | Secondary (eSARD-driven) | | | |
| **Constraints** | *Multiple types, limiting:* | | | |
| | Mfg. Techniques<br>Digital technologies<br>Resource availability | | | |

*Table 24. eSARD implementation and resource utilization optimization (substance).*

### 5.3.4. eSARD Overall Foundations and Uniqueness for ARR

This method tackles many especial design characteristics of evolutive systems (ARR) from a holistic and concurrent approach. Thus, eSARD builds upon other DE, SE, and evolutionary techniques altogether, as Table 25 shows. The evolutive standpoint rises from [1] acknowledging a series of environment conditions and needs for this subset of complex system architectures, and [2] the inspiration of natural evolution mechanism applied to complex engineering design and systems engineering efforts. Therefore, the process associated to the development of these architectures (eSAR) shares those foundational points and builds upon the gaps (section 3.4 and 4.1) and strengths (chapter 3) of state-of-the-art design engineering and systems engineering practices. Furthermore, it also includes unique features and modifications developed during the research activity in support this thesis and an easier applied practice of the method.

Table 25 presents an organized summary of these inputs for a technique capable of addressing multiple ARR needs. However, this classification is not rigid, and these foundational inputs and comparisons are based on the extensive literature review in chapter 3. All these techniques have important impacts across all eSARD phases. They are organized in two areas: [1] those already present in other methods or with a very similar implementation (red) and [2] those unique to eSARD processes (purple). However, these two groups can present multiple connections among them. The ARR classification also relates to the influence of the method over all DOI sectors (design, operations, and implementation).

| ***Technique*** | ***Domain*** | ***Class*** | ***Foundational for eSARD*** | ***Similar / Unique to eSARD*** | ***References*** |
|---|---|---|---|---|---|
| Vitruvius's | DE | *Classical* | Strength + utility + beauty | Evolutive GBS | (Roth, 1994) |
| ***Descriptive Design*** | DE | *Descriptive Design* | Vision-driven | Includes system definition | ***Table 12, DE5*** |
| Asimow | DE | *Descriptive D.* | Geometry + Sketch | Includes logical parameters | (Asimov, 1976) |
| Spiral | DE | *Descriptive D.* | Design loops | Full-cycled and networked | (Evans, 1959) |
| ***Prescriptive Design*** | DE | *Prescriptive* | Synthesis-driven + Analysis | Geometry as a multidisciplinary | ***Table 12, DE6*** |

| | | | | | | |
|---|---|---|---|---|---|---|
| ADAPTABILITY (Geometry) | | | *Design* | | starting point | (Dym, 2013) |
| | Pugh | DE | *Prescriptive D.* | Decision matrix | Dynamic questioning (eEADQ) | (Pugh, 1986) |
| | Cross | DE | *Prescriptive D.* | Concept-to-detail | Includes resources lifecycle | (Cross, 2008) |
| | Pahl & Beitz | DE | *Prescriptive D.* | Interrelationship-driven | Similar | (Pahl et al., 2007) |
| | Eggert | DE | *Prescriptive D.* | Optimization | Similar | (Eggert, 2005) |
| | ***Design thinking*** | DE | *Design Thinking* | Non-analytical factors | Detail-driven | ***Table 12, DE7*** (Curedale, 2013) |
| | DTM | DE | *Design Thinking* | Co-evolution | Co-evolution within systems & assemblies | (Brown, 2009) |
| | Human-centered | DE | *Design Thinking* | Human perspective | Considering team dynamics | (Rosenbrock, 1989) |
| | User-centered | DE | *Design thinking* | Includes the context of the system | Addressing global design stressors | (Norman & Draper, 2018) |
| | ***Innovative Design*** | DE | *Innovative D.* | Idea-driven | Implementation is included | ***Table 12, DE8*** |
| | TRIZ | DE | *Innovative D.* | Driven by design principles | Adaptable principles (eAMG) | (Altshuller, 1984) |
| | OSTM-TRIZ | DE | *Innovative D.* | Network of problems (NoP) | Network of connections | (Fiorineschi et al., 2015) |
| | C-K Theory | DE | *Innovative D.* | Domain-independent | Maturation space | (Hatchuel et al., 2004) |
| | Morphology | DE | *Innovative D.* | Consistency-driven | Maturation-driven | (Ritchey, 2002) |
| | ***Method-driven D.*** | DE | *MDD* | Methodology-driven | System-driven | ***Table 12, DE9*** |
| | Axiomatic | DE | *MDD* | Divide & conquer complexity | Sinergy to tackle complexity | (Farid and Suh, 2016) |
| | DRM | DE | *MDD* | Research methodology | Similar | (Blessing and Chakrabarti, 2009) |
| | **Process-Driven D.** | DE | *Process-Driven* | Outcome defined by process | Outcome-driven process | **Table 12, DE10** |
| | FBS | DE | *Process-Driven* | Ontology-driven theory | System-driven method DBS | (Gero & Kannengiesser, 2004) |
| | MPM | DE | *Process-Driven* | Multidisciplinary method | Similar | (Chakrabarti and Blessing, 2014) |
| | FORFLOW | DE | *Process-Driven* | Product-design-driven | System & architecture-driven | (Rodenacker, 2013) |
| | Concurrent | DE | *Process-Driven* | Simultaneous activity | Similar | (Eastman, 2012) |
| | Set-based | DE | *Process-Driven* | Open designs and simultaneous efforts | Keep the design process open for as long as possible | (Singer et al., 2009) |
| | ***Integrative Design*** | DE | *Integrative D.* | Design + Analysis | Similar | ***Table 12, DE11*** |
| | $IP^2D^2$ | DE | *Integrative D.* | Concurrently data-driven | Similar | (Magrab and Magrab, 2010) |
| | Generative | DE | *Integrative D.* | Algorithm-driven | Enhanced by algorithm | (Shea et al., 2005) |
| | ***Design Tools*** | DE | *Design Tool* | Influence over workflows | Tool agnostic | ***Table 12, To1-4*** |
| | Concept words | DE | *DT* | Concept storytelling | Included | (Lees-Maffei, 2013) |
| | Concept sketch | DE | *DT* | Fast concept communication | Included | (Ullman et al., 1990) |
| | Technical drawing | DE | *DT* | Complex 1-4D views | Included | (Ullman, 2009) |
| | CAD/CAM | DE | *DT* | Complex assembly & analysis | Included | (Rao, 2004) |
| | BIM/BEM | DE/SE | *DT* | Complex assembly & analysis | Included | (Deutsch, 2011) (Clarke, 2007) |
| | Parametrics | DE/SE | *DT* | Multiple variations | Included and qualifiable too | (Kimura, 2001) |
| | MBSE Design | DE/SE | *DT* | MBSE design enhancement | Included & enhanced by MBSE | (Fernandez & Hernandez, 2019) (Dori, 2016) |
| | Evolutionary | DE | *EvoDE* | Evolutionary methods | Included | (Braha et al., 2006) |
| | **Evolutionary Computer Science** | DE | *EvoDE* | Applied computer science | Inspired by it and for hardware | **Table 18, Evo2** |
| | Adaptive Genetic Algorithms (AGA) | DE/SE | *EvoDE / SE* | Both parameters & algorithms change concurrently | Fully adaptable approach | (Sivanandam & Deepa, 2007) |
| | Evolutionary Strategies (ES) | DE/SE | *EvoDE / SE* | Phenotype and genotype are optimized jointly | Each system becomes heritage for new designs and efforts | (Bentley, 1999) (Rechenberg, 1989) (Beyer, 2013) |
| | SCRUM | DE | *EvoDE* | Agile practice with multiple | Similar | (Ockerman and Reindl, |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | iterative spring cycles. | | 2019)<br>(Cohn, 2010) |
| | Evolutionary System Architecture (ES Arch) | SE | *EvoSE* | Not fully formed SoS includes self-evolution, joint evolution, and emergent evolution | Subsequent design cycles using evolutive approach | (Jamshidi, 2011)<br>(Chen and Han, 2002)<br>(Grösser, 2012) |
| | Incremental Iterative Development (IID) | SE | *EvoSE* | Multiple iterative cycles of incremental improvements | Similar | (Blokdyk, 2017)<br>(Larman and Basili, 2003)(Isaias 7&Issa, 2014) |
| | Complex Adaptive Systems (CAS) | DE | *EvoDE* | Based upon stable states outside the equilibrium of complex systems | Applied to hardware-based systems as well | (Yin and Ang, 2008)<br>(Gros, 2015)<br>(Holland et al., 1992) |
| | Evolutionary Design (ED) | DE | *EvoDE* | Evolutionary biology + computer science + design (CAD) to create 3D using GAs | Enhancements done by adding SE and non-quantifiable parameters | (Bentley, 1999)<br>(Kalyanmoy, 2008)<br>(Bentley & Wakefield, 1996)<br>(McCormack, 2008) |
| | Agile Hardware | EvoHR | *Hardware Evo* | Agile software techniques + MBSE + rapid prototyping to delay design freezes | Similar | (Huang et al., 2012) |
| | ***eSARD*** | DE/SE | *Evolutive* | N/A | Based upon synergy gaps<br>Multidisciplinary<br>ARR-oriented<br>DOI-centered | |
| *REACTIVITY (function)* | C&C2-A | DE | *Integrative D.* | Function-based design | System function is key | (Albers & Wintergerst, 2014) |
| | ***SE Theories / standard*** | SE | *SE Standard* | SE practices | SE for design is foundational | ***Table 15, SE1-1*** |
| | Contemporary SE | SE | *SE Standard* | Signal, data, materials, and energy-based practice | Geometry, functions, and substance for ARR systems | (Kossiakoff et al., 2020)<br>(Lapham et al., 2014)<br>(INCOSE, 2015) |
| | SoSE | SE | *SE Standard* | System of SE for extreme complexity (synergism, self-government, reconfiguration, symbiosis, and modularity) | Synergy-based of hardware design system definition | (Badiru, 2019) |
| | ***SE Models*** | SE | *SE Model* | SE methods of practice | Included or inspired by | ***Table 15, SE2*** |
| | FFBD | SE | *SE Model* | Functional flow block diagram | Similar | (Badiru, 2019)<br>(Liu, 2015) |
| | Spiral | SE | *SE Model* | Concentric development | 3D Dimensional | (Kamrani and Azimi, 2010)<br>(Boehm, 1988) |
| | ICSM | SE/DE | *SE / DE Model* | Wheel structured and networked development | Similar & applied to hardware | (Boehm et al., 2012)<br>(INCOSE, 2015)<br>(Haberfellner et al., 2019) |
| | Walking Skeleton | SE | *SE Model* | Rapid development through unfinished solutions | Similar & applied to hardware | (Badiru, 2019) |
| | Agile SE | SE | *SE Model* | Simultaneous fast-paced SE by interconnected cycles | Similar & applied to hardware | (Haberfellner et al., 2019)<br>(Douglass, 2016) |
| | OOAD | SE | *SE Model* | Agile object-oriented analysis and design with small teams | Similar and inspired by it | (Badiru, 2019)<br>(Ramnath & Dathan, 2010) |
| | FBSE | SE | *SE Model* | Function-oriented SE | Similar and inspired by it | (INCOSE, 2015) |
| | MBSE | SE | SET | Multidisciplinary and full-lifecycle Model-based SE Workflows based on interconnected elements | Included & enhanced by | (INCOSE, 2015)<br>(Haberfellner et al., 2019)<br>(Badiru, 2019) |
| | Code-based tools | DE | DT | Programing and scripting | Included | (Barr & Massa, 2006) |
| | Digital twin | DE | DT | Digital replicas of real systems | Enhanced by | (Jones et al., 2020) |
| | Rapid prototyping | DE | DT | Fast working models | Included | (Cooper, 2001) |

| | | | | | | |
|---|---|---|---|---|---|---|
| | ***Evo. Software D.*** | *EVO DE* | EvoDE | Multiple techniques | Similar and inspired by it | **Table 18, Evo3** |
| | ***Evolutionary SE*** | *EVO SE* | EvoDE | Multiple techniques | Similar and inspired by it | **Table 18, Evo4** |
| | Evo. System Dev. Prototyping (ESDP) | SE | EvoSE | Rapid engineering and evolutionary development of software through multiple early working versions | Similar and inspired by it | (Budde et al., 2012) |
| | Incremental Iterative Development (IID) | SE | EvoSE | Iterative and incremental cycles for SE and software development | Similar and applied to hardware | (Blokdyk, 2017)<br>(Larman and Basili, 2003)<br>(Bittner and Spence, 2006) |
| REGENRATION (Substance) | Math driven tools | DE | DT | Multiple tools | Included | (Wolfram & Illinois, 1999) |
| | WBS | *SE* | SE Model | Work breakdown structure for schedule, efforts, and tasks | Similar and inspired by it | (Martin, 1996)<br>(Kamrani and Nasr, 2010) |
| | SEMP | *SE* | SE Model | SE management plan including functional analysis | Similar and inspired by it | (Martin, 1996)<br>(Kamrani and Nasr, 2010) |
| | INCOSE SEFT | *SE* | SE Model | Evidence-based SE Competency framework | Similar and inspired by it | (Liu, 2015)<br>(Badiru, 2019) |
| | Vee | *SE* | SE Model | [Right] Design (requirements)<br>[Bottom] Implementation<br>[Left] fabrication | Similar. It includes multiple networked Vees. | Forsberg and Mooz, 1992) |
| | Integrated Product Development (IDP) | *SE* | SE Model | Process-oriented full lifecycle approach based on continuous integration and design-to-manufacturing. | Similar and inspired by it | (INCOSE, 2015) |
| | **SE Tools** | *SE* | SE Tool | Influence over workflow | Tool agnostic | ***Table 15, SE3*** |
| | Diagrams | *SE* | SET | Various tools | Included | (Karayanakis, 1995) |
| | Matrixes | *SE* | SET | Various tools | Included | (Burge, 2009) |
| | Codes | *SE* | SET | Various tools | Included | (INCOSE, 2015) |
| | Analysis | *SE* | SET | Various tools | Included | (Badiru, 2019) |
| | Simulations | *SE* | SET | Various tools | Included | (INCOSE, 2015) |
| | **SE Languages** | *SE* | SEL | Multiple languages | Language agnostic | ***Table 15, SE4*** |
| | **SE Frameworks** | *SE* | SEF | Multiple frameworks | Framework agnostic | ***Table 15, SE5*** |
| | Test Driven Development (TDD) | *DE* | evoDE | Rapid cycle of testing, coding, and refactoring with small delivery increments | Similar. It is applied to hardware. | (Astels, 2003)<br>(Shore et al., 2008) |
| | Adaptive Software Development (ASD) | *DE* | evoDE | Change-driven approach intertwining concept, development, and management for CAS. | Similar. It is applied to hardware. | (Highsmith & Highsmith, 2002)<br>(Koza et al., 2005)<br>(Yu et al., 2019) |
| | **EVO Hardware** | *EvoHR* | Hardware Evo | Influence over workflow | Tool agnostic | **Table 18, Evo5** |
| | Multi-Agent System (ABM) | *Evo HR* | HR Evo | Agent-based implementation methods include collaborative, concurrent, planning, advanced manufacturing, and holonic manufacturing systems. | Similar. It is applied to hardware. | (Weiss, 1999)<br>(Gräßler and Pöhler, 2017)<br>(Shen, 2019)<br>(Monostori et al., 2006) |
| | Evolutionary Machine Design | *Evo HR* | HR Evo | Evolvable hardware and genetic programming are used to improve and optimize hardware and behavioral system capabilities. | Similar and includes full cycle design and implementation | (Nedjah & Mourelle, 2005b)<br>(Koza et al., 2005)<br>(Bekey & Goldberg, 2012) |

*Table 25. Summary table regarding the comparison between traditional DE-SE techniques and DSE eSARD methodologies.*

## 5.4. Design Objectives

The process to design and develop any new system architecture, independently of the field, must deal with many barriers based upon previously described stressors (chapter 2) as well as key specific system characteristics. This is even more relevant when the process also considers as much of the system lifecycle as possible including manufacturing and operation considerations. The main objective here is to increase system performance and efficiency beyond any existing heritage and from a system-level perspective. This is especially critical when dealing with changing and complex eSARs.

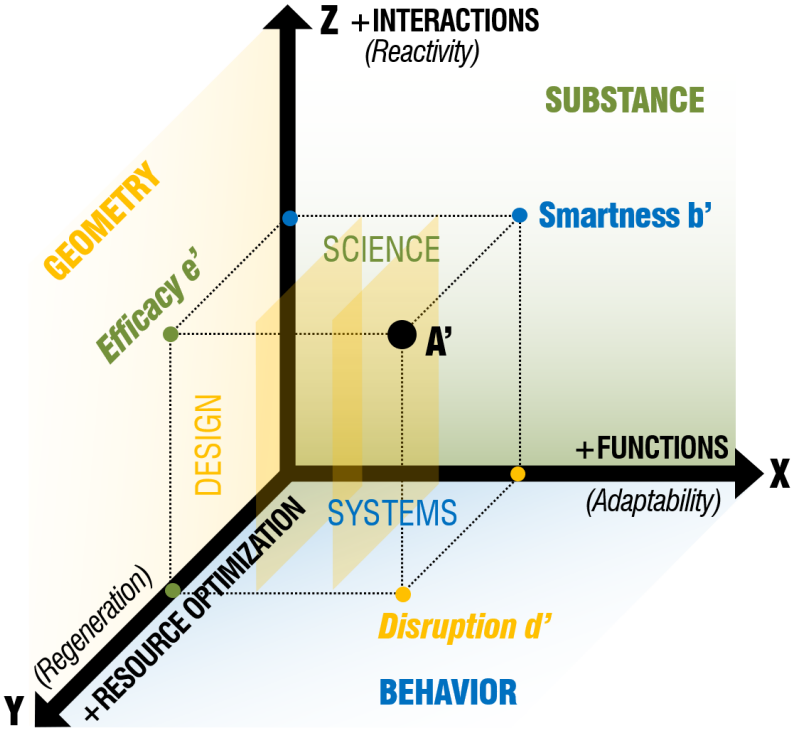



*Figure 134. eSARD evolutive design reference framework addressing balances, process objectives, and design principles.*

To develop the design process section 0 (Figure 125) provided a three-dimensional coordinate system that describes the complexity of an eSAR across three evolutive keystones, such as adaptability, reactivity, and regeneration (ARR). The overall goal of an eSARD process is to either evolve a system design from situation A to B (Figure 125), as well as to efficiently create new systems as close as possible to situation B. While relationships among design drivers within those coordinates can be very complex (section 4.4, Figure 124), such framework provides a great reference towards organizing and assessing objectives. Looking closely at such relationships across that simplified framework (Figure 134) allows to create the following approach: [1] any given family of design solutions (e.g., adaptable solutions enabling multiple functions) could be contained on a plane (yellow) perpendicular to the adaptability axis, [2] such plane is defined by both regeneration and reactivity axes, [3] within that plane multiple relationships can be studied, such as driving forces (e.g., resource use vs interactivity), risk postures, and other related driver analysis (section 4.3). Similarly, other dihedral studies could be done for any specific resource strategy or system performance design within the other two planes.

In summary, a complex three-dimensional relationship challenge can be simplified to study, assess, and develop key design objectives and approaches within the eSARD process. This helps to guide highly interactive and changing design paths. Thus, within a design effort this method looks at the current family of solutions or design path by addressing potential solution from both three-dimensional and bidimensional studies. Table 26 summarizes these relationships within the evolutive design tetrahedron that pursue the following **general objectives** within an eSARD design effort:

- **Best design path (3D).** A foundational objective within eSARD is to find the best path towards a design family that offers the most adaptable, reactive, and less resource intensive solution. Such objective is a three-dimensional assessment within the evolutive framework. For instance, in Figure 135 a design path connecting multiple solutions (purple) should tend towards B enabling solutions with more functions, more interactions, and less resources.
- **Balanced synergy (2D).** Within each reference plane (geometry, resources, behavior) any solution can be studied and optimized based on variables that exist on both axes defining the plane. This 2D study also allows to optimize a solution given a specific family of geometries, system performances, or resource allocations.
- **Optimized design driver (1D).** Within each axis multiple drivers can also be individually addressed and optimized.

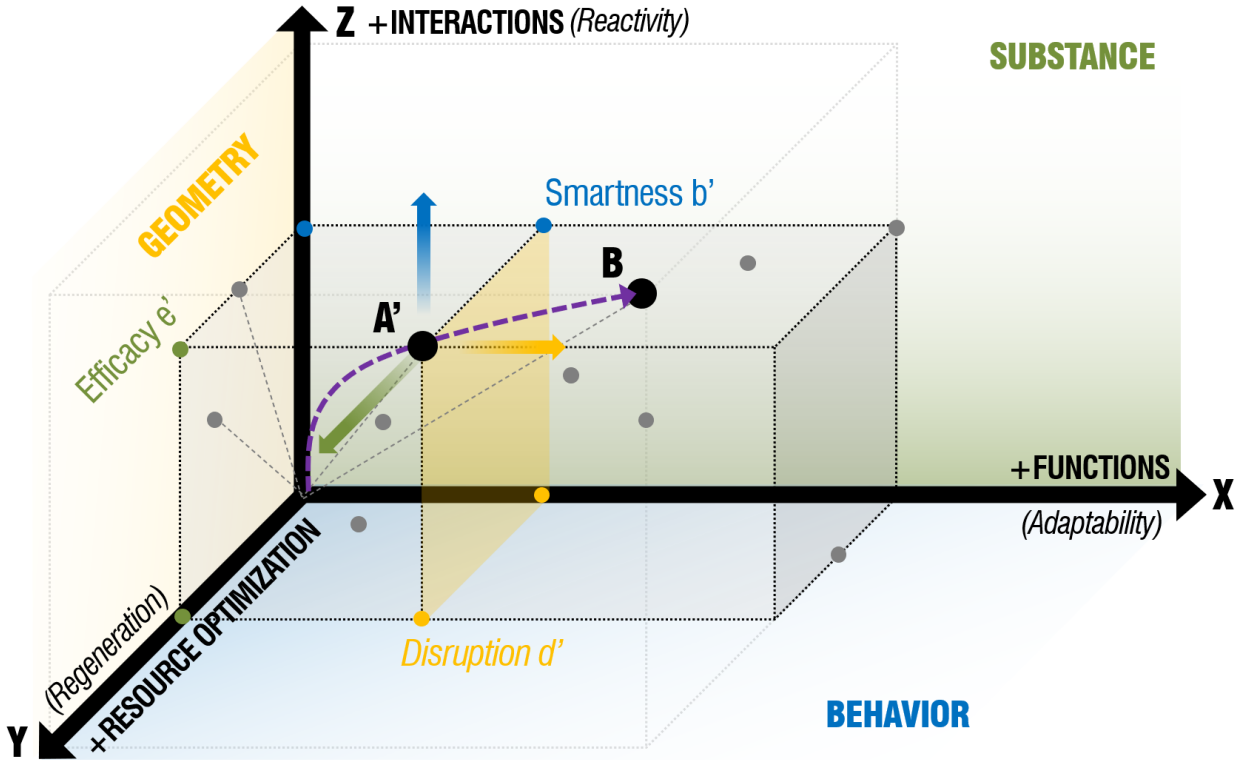



*Figure 135. Assessment of design solutions and paths within the eSARD evolutive design reference framework.*

The next sections elaborate these objectives and relationships to be used both as a guidance approach for eSARD processes, as well as a foundation to implement design principles towards both systems and processes.

| *Objective* | *Key Axis* | *D. Family* | *View* | *Measurement* | *Plane* | *Counter force* | *Studies* |
|---|---|---|---|---|---|---|---|
| **Disruption** | Adaptability | Geometry | Architecture | Functions | YZ *(yellow)* | Heritage | *Trades Optimization Feasibility Style* |
| **Smartness** | Reactivity | Behavior | System Performance | Interactions | XY *(blue)* | Passiveness | *Trades Optimization Reliability* |
| **Efficacy** | Regeneration | Substance | Resource Use Science | Resource Utilization | XZ *(green)* | Cost | *Trades Optimization Efficiency* |

*Table 26. Key coordinates and elements within the eSARD simplified design reference framework.*

### 5.4.1. **Adaptable Design:** Disruption vs Heritage

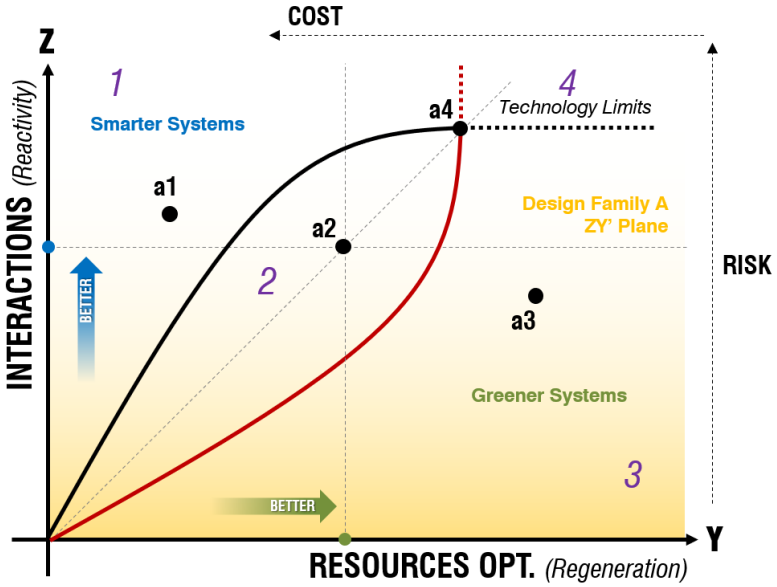



*Figure 136. Interaction (reactivity) vs. resource optimization (regeneration) within the geometry plane (design).*

For a given family of designs in the geometry plane, the balance between system interaction and resource optimization of such solution can be addressed. Figure 136 shows those solutions within the ZY plane of the reference framework. The balance between reactivity and regeneration of the system design presents several key zones:

- **Zone 1 - Smarter systems** are capable of more interactions. The black line separates high-performance active solutions from those more passive solutions.
- **Zone 2 - Balanced** architectures with a good ratio between reactivity and resource use optimization.
- **Zone 3 - Greener area** includes more sustainable and efficient solutions from a resource utilization perspective. The red line delimits low-performance passive systems.
- **Zone 4 - Technology limitation**, natural laws, and design requirements limit feasible and efficient solutions within a family of system architectures. This area presents systems that cannot keep evolving efficiently.

Furthermore, when studying this graph from a risk and cost perspective it brings the notion that more interactions mean smarter systems, but also more possible failures and thus a higher risk. Similarly, a more efficient use of resources such as recycling, repurposing, and even regeneration can potentially reduce cost too. Risk and cost here are indicators of complex relationships behind these solutions, as well as critical subjects in many process and solution assessments.

An eSARD methodology should drive the design process towards zone 2 to reduce risk and improve cost effectiveness. Although, the goal in such endeavor is not to reduce complexity but to manage it. Just like happens in natural mechanisms, complexity here is also a source of efficiency if systems are designed and managed properly and consequently.

At the same time, designing future systems tends to be conditioned by heritage either as a reference that should be surpassed, or as a measurement to its validation regarding feasibility, risk, and cost. Yet agile approaches (Huang et al., 2012) have barely tried to infuse software methods into hardware workflows. Thus, heritage is both an enabler and a barrier in the context of new SE and DE developments. Such past influence runs deep in multiple technical, artistic, and social structures. These condition from Hollywood movie remakes to public space agencies funding allocations. An evolutive approach builds upon system synergies and includes heritage too. This becomes both a process driver and a validation tool, but it should not be a constraint for design, implementation, and system operations. Therefore, if a design disruption is a goal for the process to improve and surpass previous solutions, heritage is its counter force. Of course, these do not only apply to geometrical design drivers and associated variables, but to other evolutive principles and keystones as well.

### 5.4.2. **Reactive Performance:** Smartness vs Passiveness

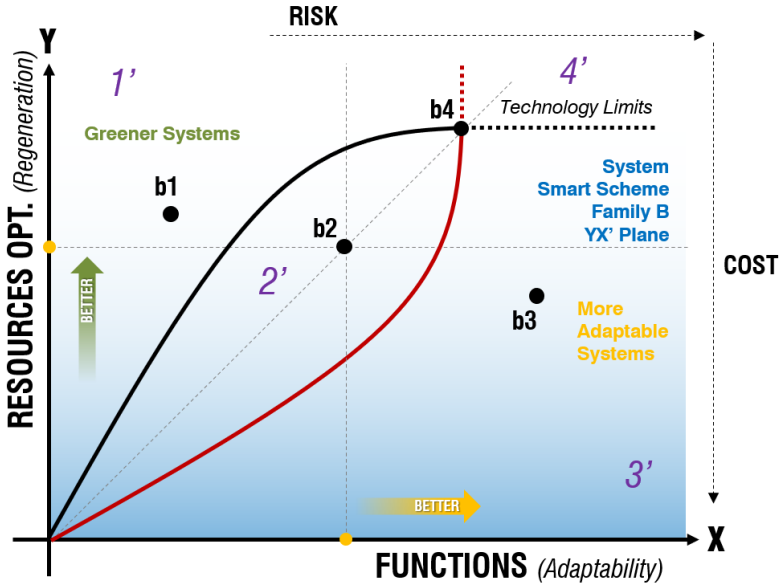



*Figure 137. Resource optimization (regeneration) vs. functions (adaptiblity) within the behavior plane (performance).*

The same design or family of designs can be studied in such 3D reference framework by addressing the balance between functions and resource optimization for any given smart performance scheme. In essence, a family of system solutions presents a series of features that allow them to manage their interactions (hardware, software, or both). Figure 137 presents this perspective from the YX plane where these areas can be assessed as it follows:

- **Zone 1' - Greener systems** capable of better resource utilization. The black line defines regenerative solutions.
- **Zone 2' - Balanced** architectures with a good ratio between adaptability and resource use optimization.
- **Zone 3' - More adaptable systems area** delimits more adaptable solutions capable of performing more functions. The red line defines low-performance and depleting systems from the standpoint of resources utilization.
- **Zone 4' - Technology limitation**, natural laws, and design requirements limit feasible and efficient solutions within a family of system architectures. Similarly, this area delimits systems that cannot keep evolving efficiently.

From a risk and cost perspective, the more functions a system design is addressing, the more evolutive it could be. Following that rational, the more evolutive the system is, the higher its potential risk could be from a systems standpoint (since it can have more elements) and thus it might require better design efforts to avoid failure. However, from an interaction and purpose standpoints, the risk of failure is lower since the system could handle more environmental, design, and use changes (adaptability). Also, a better use of resources from a recycling and repurposing standpoint also means a potential cost reduction, at least in the long term. Risk and cost are still indicators of complex relationships across all evolutive drivers behind these systems. An eSARD process drives the system design towards zone 2 to reduce risk and improve cost effectiveness. Once more the goal here is to manage and optimize complexity as a cost reduction tool.

Nowadays system architectures tend to be more adaptable and reactive due to the addition of sensors that digitalize their use, as well as data-driven design approaches that can improved and updated over time based on their interaction with the environment. Both these tendencies are software-based and apply to many fields, such as smart cars, home appliances, health devices (Krohn and Metcalf, 2020), etc. The internet of things (IoT) is a good example of this growing new paradigm (Lee and Lee, 2015). Furthermore, highly technical areas such as drones, off-road vehicles, and advanced robotics (Merten and Gross, 2008), among others also embrace this approach from a hardware-based perspective that allows them to physically change their geometry. In other words, complex systems are becoming smarter and physically evolvable (Haddow and Tyrrell, 2011), thus design methodologies need to embrace it by enabling less passive and limited solutions.

### 5.4.3. Regenerative Resource Utilization: Efficacy vs Cost

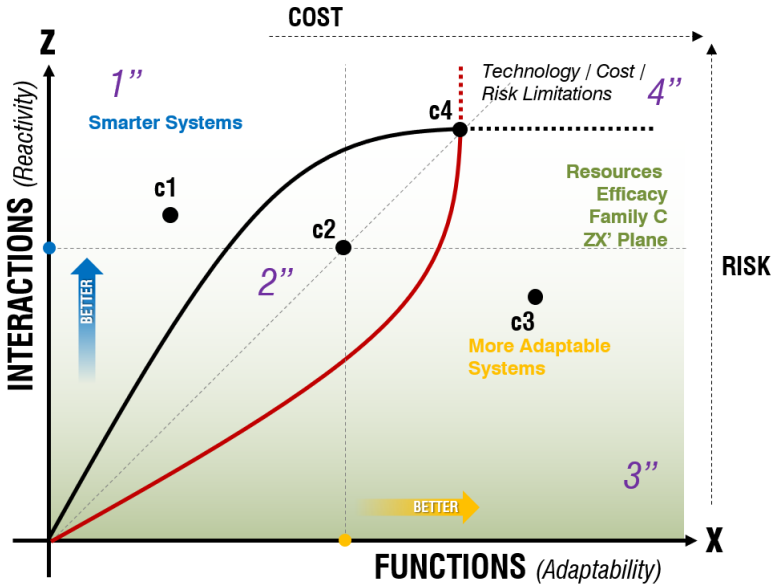



*Figure 138. Interaction (reactitivy) vs. functions (adaptiblity) within the substance plane (resource science).*

Finally, the last objective in an eSARD process is about resource utilization schemes and the science approach behind them. Looking at any given family of resource regeneration and management solutions in the ZX plane we can also study key relationships between reactivity and adaptability (Figure 138). Generally, these areas can be identified:

- **Zone 1" - Smarter systems** capable of better reactivity while enabling more adaptability and resource optimization. The black curve delimits systems with higher reactivity (smarter systems) but a slower growth towards adaptability.
- **Zone 2" - Balanced** system in between reactivity and adaptability for a given resource utilization scheme.
- **Zone 3" - More adaptable systems area** limits solutions capable of performing more functions with less interactions. In essence, these are evolutive and adaptable systems with a more passive design philosophy. The red line bounds low-performance systems from an adaptability standpoint.
- **Zone 4" – Technology, cost, and risk limited area** defines unfeasible, unreliable, and unaffordable solutions. This area encompasses systems that cannot provide feasible evolutionary alternatives efficiently.

As it was elaborated in the previous section, the more functions a solution can entail in terms of design, analysis, and implementation cost, the more such system can potentially improve the return on investment due to a higher adaptability, easier upgrades, and part repurposing. Similarly, more interactivity in the system involves more complexity during the design effort, regardless of this becoming the ultimate benefit towards the system adaptability in the longer term.

An eSARD process should again drive the design effort towards zone 2, to find balanced solutions among reactivity, adaptability, cost, risk, and feasibility limitations. The scheme behind resources utilization during the lifecycle of the system and its development process (eco-devo) presents multiple economical, sociological, and cultural complementary aspects.

Analyzing the projection of a design solution over the resources plane (XZ) in the evolutive framework (Figure 138) reflects the overall balance between efficacy and cost as opposing forces. Often, an efficacy paradigm addressing the use of resources requires initially more costly design efforts, which can be distributed over the lifetime of the system, as well as multiple customers and production elements. However, as chapter 2 presented we are facing times were the use and application of resources will require further and deeper studies due to multiple reasons beyond scarcity. Thus, such balance must be embraced by an eSARD method with the same evolutive principles that characterized a system architecture evolution over time. This is even more relevant when not only the system sustainability, but some level of resource regeneration is the ultimate objective of the process.

### 5.4.4. Overall Conclusions

One of the overall objectives of an eSARD process is to obtain more efficiently a better system performance under multiple external and design stressors as chapter 4 presented. However, while managing such process to go from situation A to B (Figure 125) is the goal, there is an associated cost and risk based upon knowledge, required effort, heritage, and even cultural constraints that include previous heritage, uncertainty, and unknowns. Design (system architecture), implementation (resources utilization), and operations (system performance) represent three main sources of the total cost, which could be identified as a fraction of the area under the graph in Figure 139.

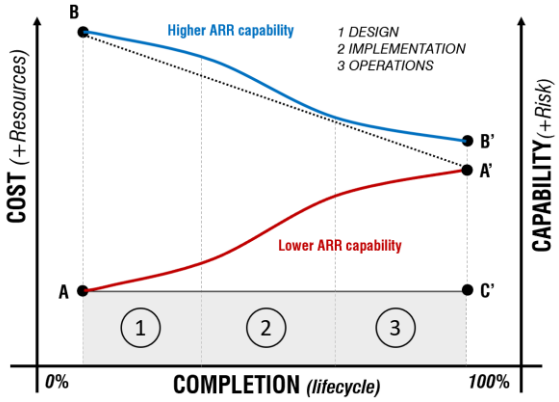



*Figure 139. Relative cost and evaluation of system capabilities among different types of eSARD strategies.*

From an overall perspective, we can study the development process of a system architecture by considering its relative cost and ARR capabilities (adaptability, reactivity, and regeneration) over the full system lifecycle (from design to operations). Thus, Figure 139 shows that architectures with higher ARR capability (blue plot) can do better with less, addressing more system needs (requirements) with less resources. However, they also require more resources for its development, especially if there is no previous heritage. Moreover, these architectures also involve more risk since they mean more complexity. Nevertheless, they also potentially reduce the need for resources during implementation and operation phases. In essence, these systems present a smarter approach in the use of resources but require more complex designs to implement them.

On the other end, system architectures presenting lower ARR capabilities, and more heritage influence can have lower design costs. However, these might need more resources during implementation and operations since the system is less capable as well. In conclusion, increasing ARR system capabilities can lead to reduce the overall relative cost. The eSARD process enables and fosters those objectives by following this development points of an evolutive system:

- **Address and minimize the cost** of development (design), manufacturing (implementation), and use (operations) by enabling alternative design solutions faster, while comparing and validating simultaneously against heritage.
- **Address cultural influence.** The design culture of an organization and even an individual can become rigid because of the relative influence of both heritage and success, especially when it comes to system performance or design tradition. Thus, constraints related to supply chain and institutionalized methods can [1] obstruct new system architectures, [2] conditioned the workforce upon organizational heritage, and [3] increase the general development cost. For instance, an established fossil-fuel-based car manufacturer can become rigid and stagnated without adopting new electric power trains, which affects dramatically future productions and design requirements. If complexity and adaptability are rising needs, the more adaptable product and process are, the better it gets to reduce cost, improve times to market, and increase both customer and workforce engagements, among other challenges.
- **Allow multiple design paths.** An evolutive architecture system architecture is not just a point-design, but rather a family of solutions that can be understood as a system design species. Modern computational-driven design tools such as generative design already allow us to address them as such. For instance, multiple structural solutions simultaneously can be generated based on loads, interfaces, and keep-out design volumes. And the tendency is to enable more disciplines within such design processes, allowing us to look at a point-design as a part of a continuous spectrum of solutions where multiple variables are addressed simultaneously. In other words, an AI or computational network generates thousands of solutions based on many permutations of key variables, so any obtained solution is just an instantiation among many. The eSARD approach embraces this approach, while allowing less technology-dependent paths to keep designs as open as possible, for as long as possible (Huang et al., 2012).
- The process should enable **enough details and enough context**, so any solution can be compared properly. Thus, it must tackle both full systems as well as discreet subcomponent topics requiring more design maturation.

## 5.5. Design Principles

Once the approach and general objectives towards an eSARD process have been established, the next step is to address all design principles driving an evolutive architecture that such eSARD workflow enables. Next sections describe 6these principles based upon all key evolutive keystones (ARR) that eSARD methods embrace, develop, and enhance.

### 5.5.1. Better with less - Kill the Problem by Design (*Adaptability*)

Any complex system architecture is going most likely to operate at multiple levels simultaneously, including mechanical, logical, thermal, and electronic, among others. For instance, the architecture of an automobile must handle mechanically both power transmission and suspension, manage thermally combustion and electrical subassemblies, while running multiple lines of code to manage infotainment and sensor arrays. Within such context of complexity, a known efficiency approach and efficacy principle has been to do 'more with less'. Architecture design principles such as 'less is more' by Mies Van Der Rohe (Benevolo, 1977), topology optimization structural techniques, and the creation of meta-material (Wadley, 2006) are good examples of removing unnecessary components and concepts, while accomplishing more functions with less elements. This presents implications that ripple through the full cycle of design, implementation, and operations. Even from a management and business perspective, the concept of frugality (Radjou and Prabhu, 2014) has been embraced during the last decade due to the increasing level of complexity in products and processes, as well as other challenges related to resources scarcity.

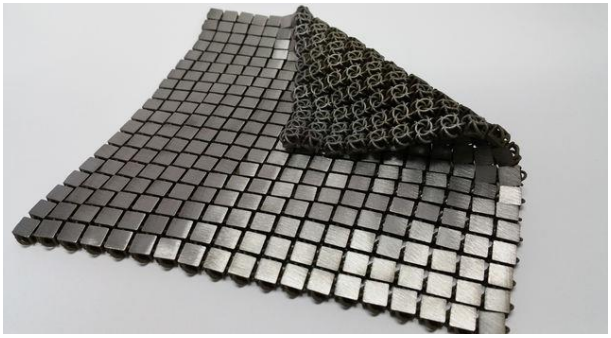


*Figure 140. Multifunctional 3D printed fabric developed by the author, after JPL NASA / Caltech (2017).*

Doing better with less leads to consider multifunctionality from both system and subsystem standpoints. If a sub-system or a simplified assembly can perform more tasks with less components, then there will be implications affecting from manufacturing costs to integration easiness. Moreover, there is a correlation between the number of functions being performed by the system, and the number of disciplines involved in the description and development of such system. For instance, the structure of a building that supports the natural convective airflow must be defined and understood from both structural, thermal, and bioclimatic standpoints. The rise of new manufacturing techniques such as additive manufacturing has enabled in the last decade the conception of very complex geometries at much lower cost. An example of this is the published metal 3D printed fabric (Figure 140) developed by the author (JPL NASA / Caltech, 2017) and capable of performing multiple functions such as thermal management, energy reflection, structural resistance, and foldability. Under these standpoints, any structure can be designed not only to manage mechanical loads, but to perform other functions such installation allocation or thermal management as well. In essence, under this perspective a system architecture will be capable of doing better with less. While this path could rise the risk of hyper-integration, making for instance repairs and upgrades more complicated, the reality is that when we look to nature almost all systems are multifunctional (and therefore multidisciplinary) and highly integrated. Bones in our bodies serve both as structure and blood cells production system, the skin serves as a sensor while it provides thermal control and protection, the trunk of a tree provides supports and enables nutrient transport, etc. All these examples of highly efficient systems follow the evolutive principle of better with less.

Therefore, an eSARD process needs to address and be driven by such principle as well. This is possible by tackling synergies among functions and subsystems, which allows to concentrate design efforts on those disciplinary gaps among them. Traditional divide-and-conquer design methods split a larger problem into smaller problems that become components, sub-systems, and disciplinary questions, which are later assembled into a system architecture. At the same time, multiple disciplines tackle such subdivided items in parallel as multiple techniques showed (chapter 3) across DE and SE domains.

However, the geometry, behavior, and substance of a system are linked among multiple disciplinary perspectives such as thermal, mechanical, and architectural design, among many more. If such relationships and interdependencies among such functions are understood, then both product outcomes and methods can substantially become much more efficient and faster to be obtained. Figure 141 compares graphically both traditional and evolutive approaches. So, to develop

system A multiple disciplinary topics must be addressed. These topics will evolve and increase definition over time by using more analysis, design evaluations, more variables, and more technical details. In a traditional approach as section 2.9 explained, these steps mostly run in parallel, and their iteration becomes the design scheme behind the process. However, within an eSARD approach connections and links among those disciplines are first identified. For instance, mechanical and thermal performances are directly related, so they directly influence the final geometrical design of the system. These links are shown in Figure 141 through blue circled in between disciplines. These links become eSARD starting points since they allow to both optimize the system and to discover new paths within the trade space. The system is defined by finding enough and adequate connections, especially those regarding key questions that ensure the feasibility of the system. Then the objective is to reduce all design loops among disciplines, while getting faster and better system solutions that address those multidisciplinary questions simultaneously. This is a mindset for the architect and a workflow rule driving tools and models.

Designing towards doing 'better with less', means to find a way to '*kill the problem by design'* from a process standpoint. This principle builds upon all previous general objectives of efficiency and aims to the system efficacy. When designing any system architecture, constraints and requirements set up objectives and barriers for such system. However, if the challenge is too complex this often implies a trade-off among those constraints and an identification of the margins in the system that the design aims to optimize. Thus, such approach affects both system design and design processes.

The goal though is always to find a solution capable of conquering constraints, as well as to infuse enough adaptability into the system towards future changes or new paths. For instance, if the objective is to design a mobility train for an off-road lightweight vehicle, such design must meet specific terrain specs and the process should also push for solution capable of working in any terrain. This is done under this principle even if it is not needed right away. However, this does not mean that extending efforts and increasing cost of such design process regardless of whether the customer requires it or not. It means that stressing the initial design effort by considering more constraints (e.g., other terrain constraints) will help in finding better and more resilient solutions. In other words, the system will be fully designed to meet certain terrain specs, but other terrain and mobility needs will also be considered during the early design phase. This will help to identify better design strategies or paths, by enabling the discovery of key disciplinary connection (blue circles) that could remained unknown otherwise. In essence, this mechanism pushes the limits beyond all initial specifications, and expands the boundaries of the foreseeable trade space from the beginning of the design process. Even if all objectives are not met, the new direction being infused in the process ensures a better result by pushing any system design pre-conception systematically. Such results affect multiple design areas by enabling the following improvement in both product and process:

- Better system designs with more ARR capability (adaptability, reactivity, and regeneration).
- Easier upgrades and better system adaptability due to those extra design considerations.
- More reliable implementation methodologies since tougher requirements imply more research and analysis.
- Faster and more resilient design cycles that are driven by over constraining the effort to push for more efficacy.

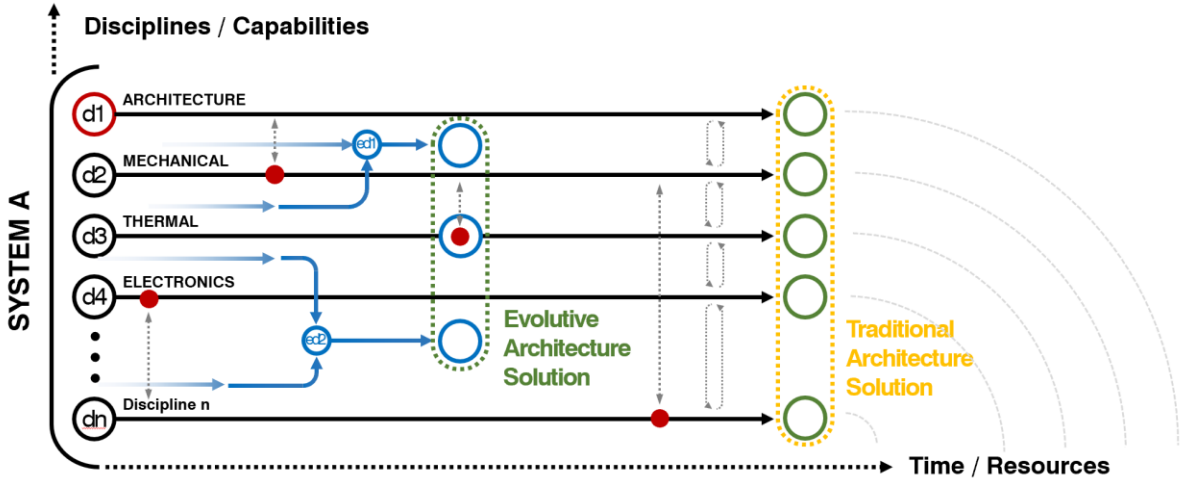



*Figure 141. Multidisciplanry and concurrent eSARD cycles versus traditional parallel design approaches.*

### 5.5.2. **Smarter with less -** Continuous Solutions and Operations (*Reactivity*)

The second principle behind an eSARD methods relates to the evolutionary principle of continuity, as well as the consideration that any system is a temporal instantiation within a continuous evolving process. This applies at three levels:

- **System**. Any system architecture could present variations, upgrades, and updates while it becomes new heritage for upcoming solutions. As such, any design process needs to address the system architecture design, as well as other implementation and development aspects. For instance, this is not about how a motorbike will perform (e.g., power, mass, design, and style), but how is going to be made (e.g., manufacturing steps, quality control, vendors, etc.).
- **Process**. If the solution can change during or after its development, an evolutive design methodology should also be adaptable enough to enable a better exploration of the trade space, and any required change in the design strategy.
- **Operations**. Finally, how the system is going to operate or be operated is part of any eSARD approach.

As previously explained, an evolutive approach considers that any given point solution is not only an instantiation, but also a member of a family or species of solutions. Thus, as a design process, is not only about the geometry, behavior, and substance that needs to be considered, but also all parametric variables, relationship algorithms, and variables, among other components. In essence, an evolutive system architecture is both hardware (phenotype) and information (genotype).

For instance, the design process for a prefabricated and lightweight balcony structure could start with the requirement of being as compactable as possible for a cheaper and easier transport. This leads to multiple structural solutions and multiple materials being considered over the design development. Furthermore, as the previous point presented, this type of design processes then considers other design stressors to reinforce solutions and find better alternatives, beyond all initial requirements. Such extra constrains include interface considerations, integration easiness, morphology, etc. In the end, the final decision could be driven by cost and material availability for instance. However, the same design process can be replicated for a different solution as a template if done properly, which will help both present and future endeavors by reducing design cost to create new products and offer new services. So, if the system architecture development has followed this principle, the final design will allow to integrate customized details, new complementary solutions, and changes in dimensions easily. The pressure to be able to add, update, and upgrade solutions in any part of the process, conditions the system architecture and all its disciplinary standpoints. In essence, considering more allows to design better with less.

Figure 142 represents graphically a continuous eSARD process approach with a series of concentric cycles that address multiple disciplines (black paths), as well as design parameters, variables, and aspects (red dots) used by those disciplines. As previous section described, the solution of the system architecture is in essence a network of those parameters, and the design process to achieve it needs to explore those networks to assess both the full system design and all related operations at every design cycle. In the process, some of those disciplines will be fused, becoming a single track as section 5.4.1 presented. Across multiple cycles, this process could be visually summarized as a three-dimensional helix with a spline axis advancing towards the most refined and better solution.

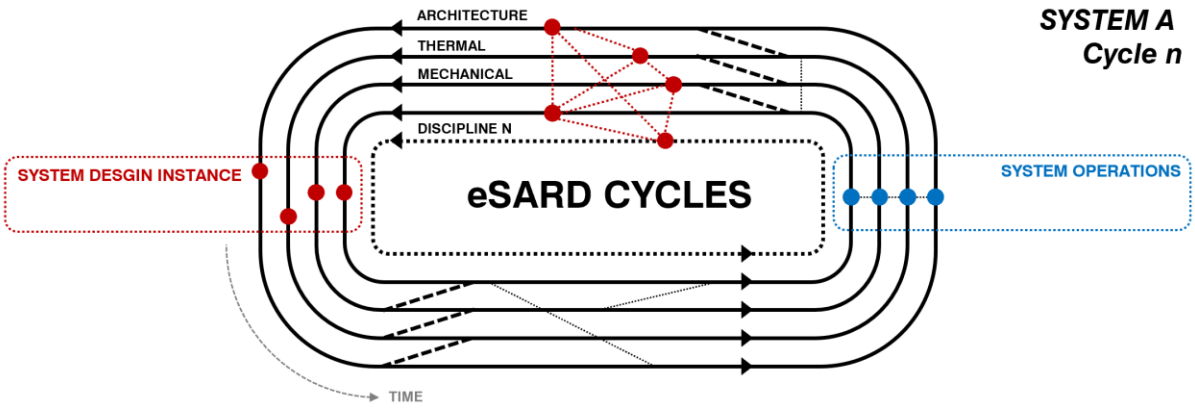



*Figure 142. Consecutive design cycle within an eSARD process addressing multiple networked solutions.*

### 5.5.3. **More with less -** Resource Utilization Lifecycle and Optimization (*Regeneration*)

Finally, addressing the utilization of resources across the full system lifecycle leads to the last overall design principle behind an eSARD method, to do 'more with less'. This also applies directly to the utilization and optimization of resources across the development process that include among others the following areas:

- **Workforce**. This includes number, expertise, and compatibility of people required in the design effort.
- **Computing power**. More complexity in terms of operations and design leads to assess the capability and scalability of any available computing power today. This not only addresses variables like the number of operations per time, but also the effort and feasibility to use such power including access, protocols, programming, technicians, etc.
- **Energy** is a critical resource that applies to products, processes, and system operations. Often this is an area that is not addressed completely (Cody, 2017) across the full lifecycle, but it drives dramatically the feasibility of a solution, especially under current or future conditions defined by the scarcity of resources.
- **Time/Schedule**. Project and market schedules are a very relevant aspect of a design process that go beyond technical considerations. Furthermore, life span and time operational constraints must be considered as well.
- **Upgradability**. If any evolutive solution is an instance within a continuous process, the capability of the system to be updated and upgraded over time needs to be addressed too. From future system interfaces to changes in use, this resource is critical to obtain good solutions in the present that enable and simplify future developments.
- **In situ resources utilization (ISRU)**. The use and management of available resources that can be physical (e.g., materials, space, air, vegetation, solar energy, etc.), logical (e.g., open-source code, communication infrastructure and protocols, etc.), and digital (e.g., web infrastructures, telecommunication networks, etc.).

Thus, cost is a relative consequence of these points, and it must be addressed from technical, human resources, ecological, programmatic, and monetary standpoints. However, it is not a resource by itself within this approach. Therefore, a design process aiming to develop an evolutive solution needs to consider these resources over all phases in the design lifecycle. This affects both the future system performance and its relative cost across these three areas:

- **Design** includes all resources being used in the research, conceptualization, and design processes.
- **Implementation** includes resources for the construction, manufacturing, fabrication, and development of the system.
- **Operations** considers all resources required to both operate and use the system over its lifecycle.

Thus, the objective here is not only to minimize the consumption of resources, but to maximize their output. For instance, the sustainability of an electric boat must consider all energy and resources used during [1] the design effort, [2] its implementation, [3] its lifetime, and all the way to its [3] decommission and recycling. Figure 143 shows graphically this point. For a given system architecture (red dot) there are multiple resources utilization and management ARR aspects.

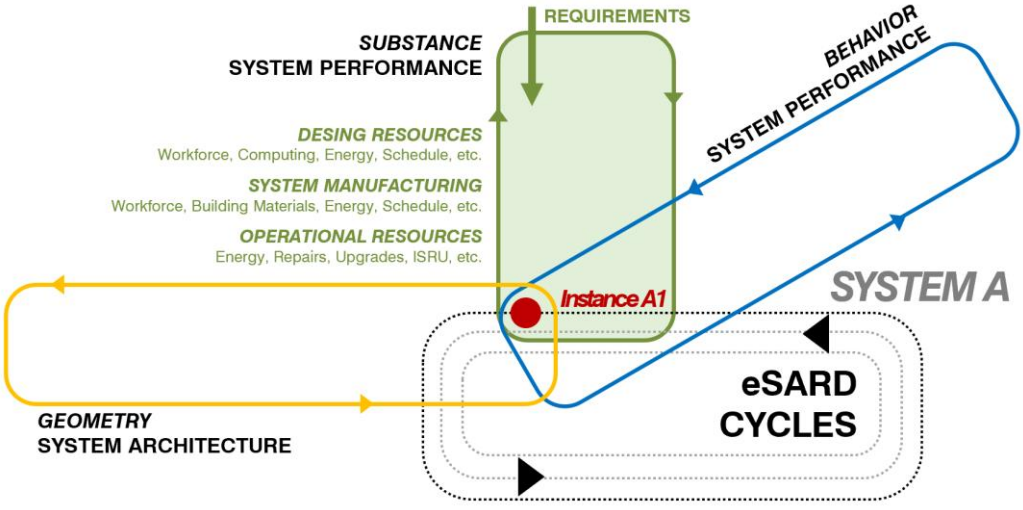



*Figure 143. Resource utilization lifecycle within an eSARD development for an instance design (A1).*

## 5.6. Evolutive Design Helix Model

Chapter 4 laid out both rational and foundational principles regarding the definition of an evolutive system architecture or eSAR. Moreover, chapter 3 identified gaps across state-of-the-art SE and DE techniques, including those applying evolutionary principles. And finally, the previous section presented what the subsequent evolutive design methodology must pursue in terms of objectives and principles from a combined DE and SE perspective.

The next step then is to develop the eSARD process, which presents a networked nature addressing all three key activity nodes such as [1] **system design**, [2] **system operative optimization**, and [1] **system implementation (DOI)**. The first node in such method represents the main objective of this research, but all of them are intimately intertwined since they must be developed concurrently. This section defines specific details of the eSARD workflow, including framework, milestones, tools, dynamics, and routines for highly adaptable design system engineering (DSE) methodology.

Figure 144 presents a graphical description of the **eSARD helix model (eSARD_he)** where three sectors (triangles) are distributed over a spiral scheme describing different design gates (milestones), routines (tools), and paths used to cover all three ARR areas of a system (adaptability, reactivity, and regeneration). The representation of a single system at any given time is presented by the spiral in Figure 145. ARR vectors (red arrows) create the structure defining DOI sectors, phases, milestones, etc. for any given system. Multiple iterations within the same system are represented by translations in the sector similarly to incremental SE techniques or IID (section 3.3).

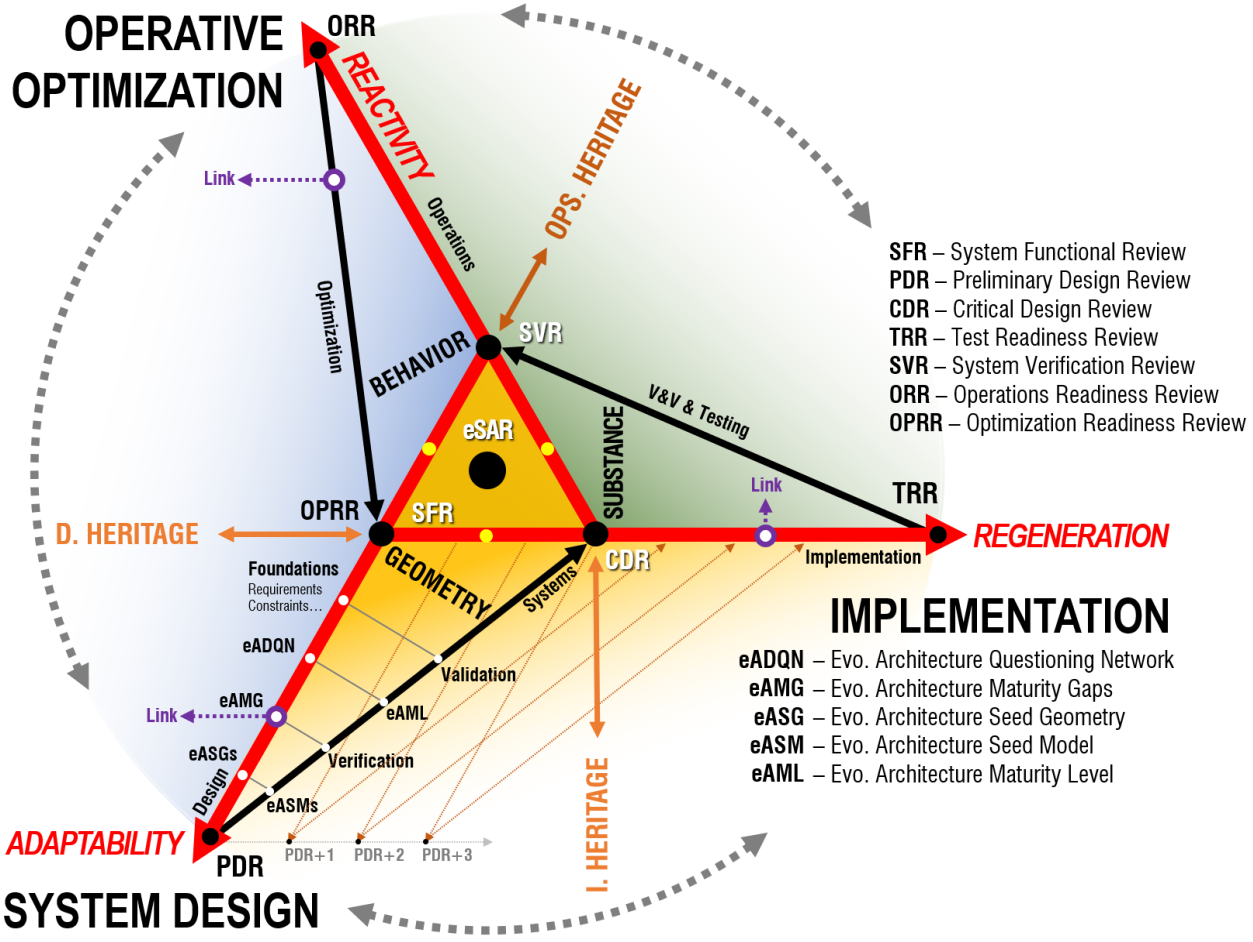



*Figure 144. Evolutive system design networked process, presenting all three ARR activity nodes addressing design, operations, and implementation.*

Considering these continuous changes in multiple instances within a common species (evolutive continuous design), the evolution of the system could be represented as a three-dimensional helix as Figure 145 shows. Such helix can also be referenced within the evolutive framework described in section 5.4 (Figure 134). Within such framework, any point design (e.g., purple B) represents a family of instances or solutions. The eSARD 2D spiral (Figure 144) process can be applied to each one those points, with the objective to describe, develop, and manage a new solution that is referenced in this coordinate system. Thus, the 2D spiral graph represents the process at the solution level, while the 3D helix represents the work at the species level, and both scales can be compared and connected within such evolutive framework. 2D and 3D spirals are networked processes, so within them all related variables, milestones, routines, and tools are linked across their workflow. For instance, manufacturing validations at the implementation sector can also condition design maturity gaps and vice versa. Since everything happens simultaneously, having correct data is key. The flow of milestones helps guiding and validating the process, while enabling traditional methods and tools to be infused in more heritage-driven organizations.

The iterative nature of more traditional design approaches (Johnson and Gibson, 2014) is also part the activity in each of these sectors, as well as all interactions among them. However, this method includes other aspects of the lifecycle such as sustainability or decommission. ARR keystones (adaptability, interactivity, and regeneration) are directly related to the three sectors describing an evolutive system architecture: **geometry, behavior, and substance** (GBS). The method behind this approach is related to other design theories as section 4.1 presented based on chapter 3 conclusions. These include SE and DE techniques such as systematic DE (energy, material, signal - Pahl et al., 2007), FBS (function, behavior, and structure - Gero and Kannengiesser, 2004), and other evolutionary methods shown in Table 17. Nevertheless, this approach is not about creating just a general design process, but rather about creating a holistic and adaptable method that addresses all the special characteristics of hardware-based evolutive system architectures across its full lifecycle.

Within this workflow, relevant heritage inputs also become an essential aspect serving as validation, context, and comparison for new systems. Although, such inputs can never limit any design effort. So, heritage is independently assessed since the objective of an eSARD method is to develop an evolutive architecture solution that most likely exceeds previous system performance while often becoming the first of a kind. The next section elaborates in detail these design steps.

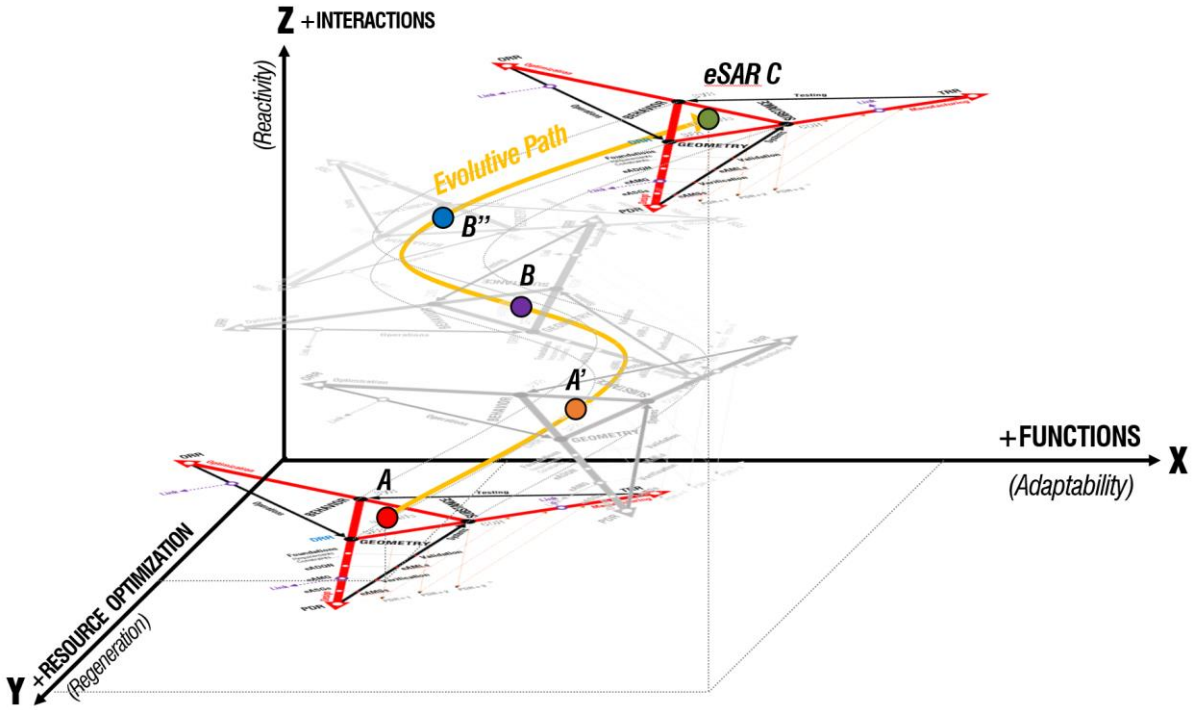



*Figure 145. eSARD helix model of design species sharing a common design path within an evolutive framework.*

### 5.6.1. eSARD_he components

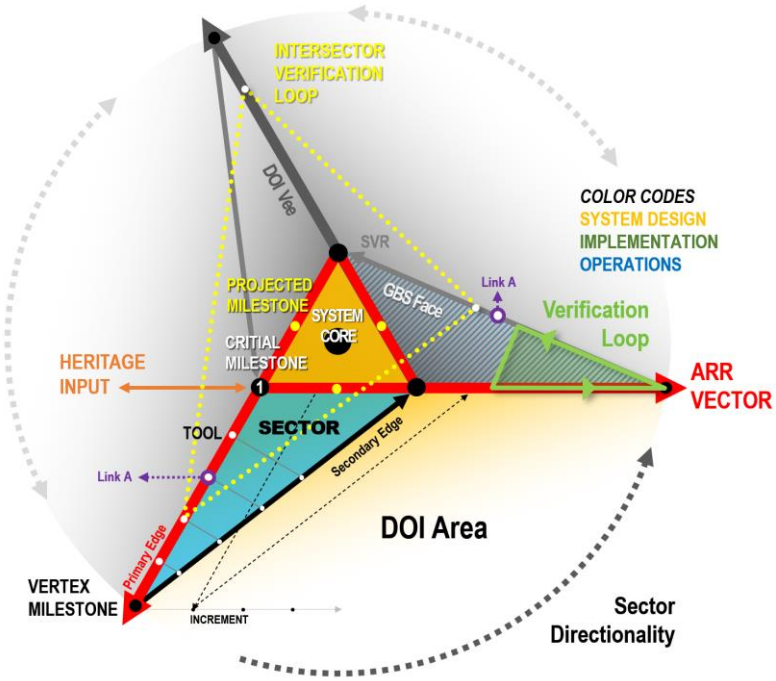



*Figure 146. Elements of the eSARD_he scheme within the 2D evolutive spiral.*

Among the most relevant and critical elements within an eSARD 2D spiral (Figure 146) are the following:

- **SARR vector** (red arrow). There are three red arrows distributed around the central triangle. These represent the three ARR areas (adaptability, reactivity, and regeneration) which are base edges of an evolutive tetrahedron. Each vector relates to key design drivers associated with ARR principles (Figure 124).
- **System core** (red central triangle). This central core is graphically defined by vertices such as SFR (DDR), CDR, and SVR critical milestones. These define system geometry, substance, and behavior (full cycle) milestones.
- **Sector** (teal triangle). Each triangular DOI sector (design, operations, and implementation) is defined by the primary and the secondary edge. At the vertex of each sector are critical and regular milestones. Each sector starts from the system core (base of the evolutive tetrahedron). These sectors represent a series of paired activities and tools following the direction of the arrows. Such tools increase the maturity and feasibility of the system architecture. For instance, the system design sector is defined by design (DE) and systems (SE) edges, which are defined by the SFR, PDR, and CDR vertices. Each vertex is a tipping point providing a change of direction. Sectors can move towards the right at any DOI sector in response to incremental changes (e.g., PDR+1, PDR+2, etc.) or new system versions.
- **Sector primary edge** (red arrow within a DOI sector). This is the first edge that coincides with the ARR vector on each sector. It starts at the system core from one of the critical milestones. This often holds primary (but not sufficient) tools (e.g., routines, activities, models) regarding the first family of processes. For instance, in the system design sector this edge holds system engineering (DE) tools that are complemented by those on the secondary edge.
- **Sector secondary edge** (black arrow within a DOI sector). This one starts from the vertex milestone similarly to the bottom vertex within a classic Vee model. The nature of the tools in this second edge is complementary to the primary edge. For instance, this would be the SE edge complementing the primary design edge (DE) in the design sector.
- **DOI vee** (three DOI triangles). These are defined by primary and secondary edges across design and system lifecycles. Unlike the classic vee model, here there are three interconnected vees representing: [1] geometry (design + systems), [2] substance (implementation + testing), and [3] operative optimization (optimization + operations).

- **GBS face** (blue angled pattern triangle). These are defined by the surface area of each sector triangle representing a face of the evolutive tetrahedron. They incorporate all relationships among GBS design drivers and tools.
- **Vertex milestone** (small black dot). This is the tipping point between primary and secondary edges of each sector. It represents the middle maturation level between the initial and final milestones of the sector. For each sector we could identified the following vertex milestones:
  - **Design, PDR** (Preliminary Design Review). This is the transition from system design (DE) that starts with SFR (concept & requirements) to systems engineering definition (DE) ending with CDR (implementation readiness).
  - **Implementation, TRR** (Test Readiness Review)**.** This marks a transition from systems implementation (e.g., manufacturing) starting with CDR, to system testing and verification ending with ORR (operations & optimization).
  - **Operations, ORR** (Operations Readiness Review). This marks the transition from system implementation (e.g., OPT) starting with SVR, to system operations (e.g., OPS-CON) ending with DRR (system recycling).
- **Critical milestones** (large and small black dots). These represent all mayor system maturation level milestones across the full lifecycle shown in the eSARD-he system definition process. The most relevant are these:
  - **SFR** (System Functional Review): includes requirements, constraints, and functions.
  - **PDR** (Preliminary Design Review): SFR + System design with basic analysis and implementation.
  - **CDR** (Critical Design Review): PDR + Full system and implementation definition (geometry).
  - **TRR** (Test Readiness Review): CDR + All implementation details, trades, and disciplinary studies.
  - **SVR** (System Verification Review): TRR + Including all tests and verification activities (geometry & substance).
  - **ORR** (Operations Readiness Review): SVR + All functional operations tools.
  - **OPRR** (Optimization Readiness Review): ORR + System optimization + recycling (GBS).
- **Projected milestones** (yellow dots). These are the projection of vertex milestones upon the core triangle. These provide a summary of the full system definition across all the ARR areas.
- **Tools** (white dots). The eSARD_he graphic is a way to visualize and check all efforts required to mature a complex evolutive system architecture. Such efforts involve a series of tools or activities. Such routines include dynamic questioning (eADQN), maturity gaps (eAMG), system architecture seed geometries (eASG), and implementation paths (eAIP), among others. The next section will elaborate each one of them in detail.
- **Link** (purple arrow). In between any given milestone or tool there can be links or connections among them. Practically this means the outcome or the parameters of one is strongly interconnected and often bidirectionally conditioned by each other. Beyond such connections among models and tools (networked process), these links highlight critical connections that are tightly coupled with other system design efforts at hand.
- **Increment** (black dash line). Variations, improvements, and changes in the initial conditions can affect both system design and outcome objectives, which leads to the need for new and distinctive design cycles. Hence, new sectors are created and displaced towards the right to represent such new operations within each sector.
- **Heritage input** (orange arrow). Heritage within an evolutive approach is used to compare against a validated solution, as well as a building block for the new system at hand (e.g., specific technology and precious subsystems).
- **Verification loop** (green line triangle). In developing and maturing the system, this process follows the direction established by multiple edges within all sector vees. However, before the system has achieved enough maturation for the next critical milestone within such process, there must be a verification process among tools on both sides of the vee. This is in essence a quality and design objective control process.
- **Intersector verification loop** (yellow dotted triangle). Similarly, this loop checks values, status, milestone development, and other critical activities, among multiple different sectors and edges (at least three or more).
- **DOI sector area** (colored circular sector). This represents the activity area that is more related towards tackling system design, operations, and implementation, respectively. Conceptually extra operations happen here.
- **Sector directionality** (circled dotted arrow). Quick visual aid that references the direction in which the design activity on multiple sectors is happening. It could clockwise, counterclockwise, or bidirectional (network).
- **Color code** (yellow, green, blue). The use of colors helps transmitting information faster, using for instance yellow for design (geometry), green for implementation (substance), and blue for operations (behavior).

### 5.6.2. System Design (Geometry) – eSARD_he

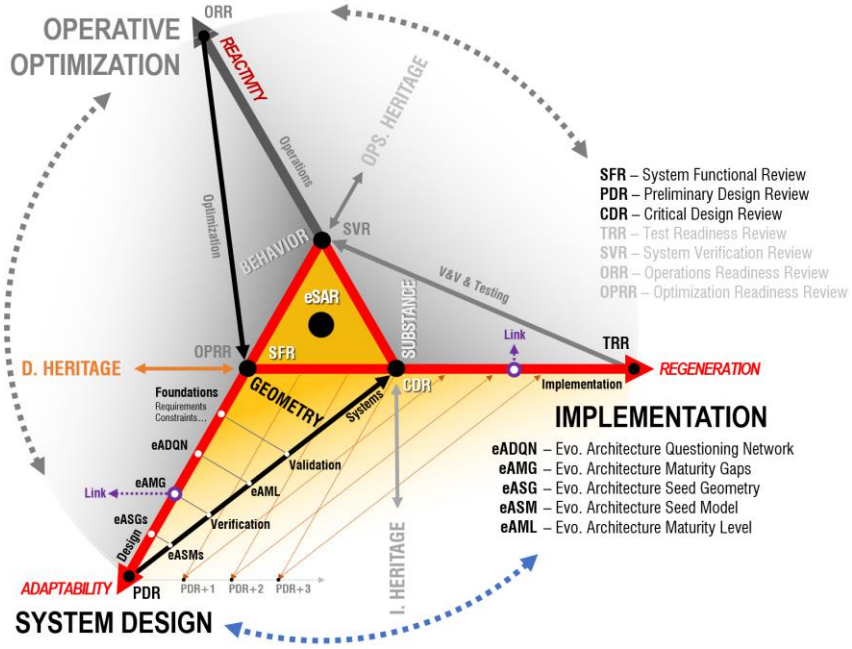



*Figure 147. System design sector within the eSARD helix diagram based on the ARR evolutive tetrahedron.*

Within the eSARD helix model, the design sector is usually the first one to be tackled and the one this research is concentrated on, so this diagram describes multiple steps and tools used within it. However, as previous sections already explained, design, implementation, and operative optimization (DOI) happen concurrently. This is being implemented differently depending on all the design resources available. Two concurrent workflows can summarize them:

- **Fully concurrent**. Available design capabilities in terms of workforce (people), computer power (connected computer/s, cloud computing, linked tools), etc. allow concurrent DOI processes using connected tools. This is possible individually or in teams but requires a previously set up infrastructure. Concurrent design facilities are common today (Shen, 2019), however in an evolutive approach these need to allow the infusion of new models and variables. Hence, a closed, concurrent, and static loop that impedes new variables and models does not work.
- **Discreetly concurrent**. In this case, the limitation of resources can force to split a network process into a series of interconnected and discreet design cycles, which later would feed each other sequentially. Under this approach, the design sector happens first, and its outputs serves as inputs for the implementation sector. Consequently, implementation and design activities feed the operation sector. There can also be inputs in the process that come from pre-activities in any of those sectors. These are the purple dots in Figure 147. In essence, while conducting the design development there could be breaks in which critical aspects of implementation and operations are addressed. This is still a networked process, but the schedule is delayed and segmented across all nodes. This approach allows for a single person to use the eSARD approach even without the support of advanced computational capabilities.

Therefore, within a fully concurrent workflow the goal is to speed up the process. These next key objectives for the design eSARD helix sector support this acceleration and they can be summarized as it follows:

- **Definition**. The objective is to mature a system concept so implementation and operations can be feasible. This means going from a system functional review (SFR) that establishes primary system requirements (as well as some preliminary eSARD secondary requirements), to achieve enough maturation so the system architecture is ready for a critical design review (CDR) level, including the full development and assessment of critical design parameters.

- **Adaptability**. This process should be able to make corrections and changes at any given time. Thus, while the sector might look like a very strict path it can be a very flexible model. The objective is to achieve gradually more system maturation, while using an adaptable and networked process to address feasibility from all DOI views.
- **Robustness**. The final system design should be robust enough against primary requirements, which are provided by the customer or defined by the challenge at hand. This should also be reinforced by secondary requirements as a product of the eSARD process itself. These allow to explore more refined solutions within the trade space, enable future upgrades much easier, reduce cost, and overcome operations and implementation constraints later on.
- **Feasibility**. Such designs must be feasible when considering all ARR design drivers and DOI technical details.

Next sections will explain in detail the set of milestones integrated within this methodology, while they also provide details regarding key tools (e.g., eADQNs) that will be further elaborated in section 5.7 and beyond.

#### 5.6.2.1. Foundation - System Functional Review (SFR)

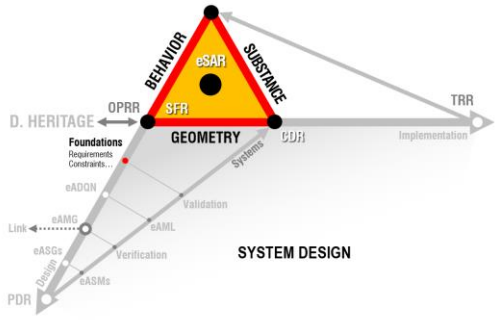



*Figure 148. Foundation and eADQNs for SFR*

Adaptability, reactivity, and regeneration principles and drivers (ARR) must be assessed to define all DOI areas of an evolutive system architecture. This mean that technical details and key requirements defining its geometry, substance, and behavior (GBS) need to be collected, studied, and developed.

Thus, understanding client requirements, needs, wishes, and constraints is a critical first step. This will be done through a series of questioning techniques (eADQNs) which allow not only to identify and challenge primary requirements, but most importantly to find synergies among those requirements, subsystems, and discipline constraints (section 5.3) leading to new secondary requirements.

This initial networked and less structured activity is represented in Figure 148 by the triangle at the core of the diagram. The main objective of these dynamic questioning activities is to discover most critical gaps within the system, which the design process should tackle to provide feasible solutions. Often this process might require multiple iterations. For instance, rather than mass reduction it could be that the packaging scheme is what makes the system design feasible, bringing a much higher level of performance against other solutions. The outcome of this step allows the process to be ready for a systems functional review or SFR (NASA, 2007) among other things. Nevertheless, this is just used as a milestone within more traditional and detailed SE and DE methodologies (Martin, 1996). The real objective of this process would be to increase the maturation level of the system.

#### 5.6.2.2. Design - SFR to PDR

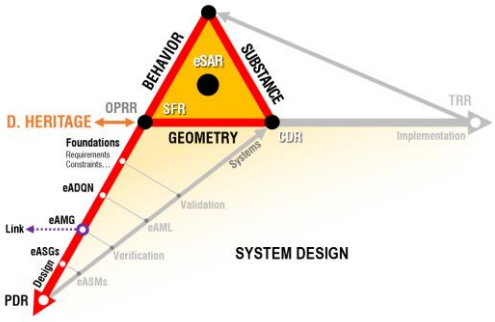



*Figure 149. Milestones and tools from SRF to PDR (eSARD)*

Questioning tools such as eADQNs (section 5.7) allow to identify where the design process should concentrate upon. This is done by identifying and defining maturations gaps (eAMG). Among all gaps in a system architecture requiring new solutions and developments, AMGs are the most critical ones. They are synergetic in nature, and the eSARD method uses them to speed up design processes (section 5.5.1). SE techniques such as walking skeleton (Table 15) and rapid prototyping (Table 12) techniques use this approach in similar ways. However, the approach of building fast upon critical points is applied to the maturation of the concept. These gaps become the focus of the design activity including among others: [1] definition (e.g., geometrical, and logical), [2] preliminary evaluation (e.g., basic analysis), [3] feasibility, [4] open options (e.g., trade space of alternative design paths), [5] DOI open questions, and [6] key technical topics (e.g., manufacturability constraints). Thus, the main goal of this part of the process is to quickly produce a concept that is mature enough to address all DOI questions and

ARR topics at the level of a product design review (PDR). The final tool used on this first branch is the creation of seed geometries (eASGs). These are representations of the system with combined geometrical and logical information that are not fully finished but represent most relevant and critical characteristics of the new system architecture. These three-dimensional representations are one of the most critical tools of the eSARD since they combine many SE and DE topics within a common space and time reference framework. This means that at the PDR point there are no fundamental laws of physics, economy, and style, among other critical topics that must be addressed. While the system still requires a lot more maturation it already presents a solid foundation for a further development. Similarly to a SE V-model (INCOSE, 2015), the PCR milestone is at the vertex of this section where the first edge of the triangle relates to system design fundamentals. Although this side of the V here also tackles key related DE aspects, so at a PDR level the system is defined and feasible. The design branch tackles both quantifiable and qualifiable variables.

The input of heritage solutions is a very useful and fundamental part of this workflow. This is especially relevant in the beginning of the process and when critical elements such as the infusion of new technologies or the integration of critical subsystem are needed. This works as a design data point, a comparison tool, and validation input that could be utilized across the whole process. This method allows to assess the feasibility of both system foundation and performance.

#### 5.6.2.3. Systems - PDR to CDR

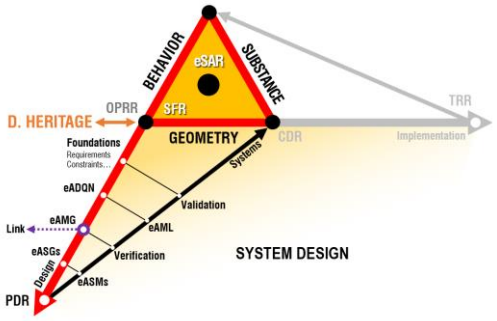



*Figure 150. Milestones and tools from PDR to CDR (eSARD)*

The other side of the design triangle after the PCR milestone is about systems modeling and validation. Similarly, to the development of seed geometries, a seed model (eASM) is also created and related to a three-dimensional model while capturing more complex logical and analytical relationships. These two complement each other working as basic tools addressing multiple solutions (e.g., parametrically, or generatively) and bringing more detail into the design process. See Figure 150 for reference.

Then, the next step is to assess the level of maturity (eAML), which is a working milestone (not a review) that keeps evolving, while enabling other comparison and quality control processes. During the design modeling steps, key analysis can be happening concurrently and ideally with a multiple disciplinary standpoint. This allows for the system solution to be initially validated, becoming the last stop towards a critical design review level (CDR). This last part completes the sector addressing the geometry of the system and its connections to the other DOI sectors. While the overall diagram shows a flow of natural steps, in reality all these steps happen under a networked paradigm of multiple connections among them. Such links respond to quantifiable and quantifiable requirements, as well as parameters affecting both system and process needs.

#### 5.6.2.4. Continuous design - Multiple CDRs

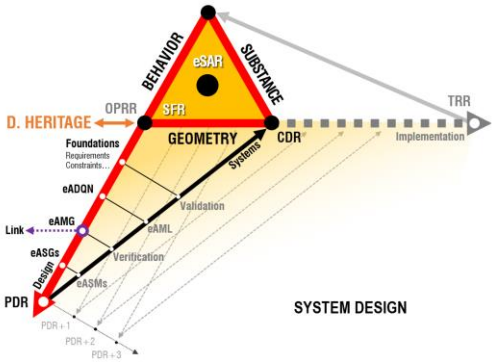



*Figure 151. Milestones and tools for multiple CDR (eSARD)*

Once the evolution of the system is aiming towards the required maturity level for a CDR based on requirements, new discoveries and secondary requirements made through the eSARD process could challenge these initial assumptions. As a networked process this can also happen due to bidirectional inputs coming from the implementation and operative sectors.

In the case of an initial system design evolving into a different version (Dori, 2011), the previous CDR level system definition (vertex of the sector) shifts to the right repeating the same process towards PDR+1, PDR+2, etc. This is similar to incremental SE methodologies such as IID (Forsberg, 2020) and walking skeleton (Badiru, 2013). This does not mean that all previous work must be redone. In fact, the development of evolutive models and designs assumes this will always happen so incorporating parametric and reusable methods to

repurpose previous work is part of this process. This is a critical point in the way of thinking required for an eSARD. Thus, it is not only about achieving the best system design but to perform such work assuming sudden changes at any time, so methods and results must be adapted accordingly and efficiently. This continuous design workflow not only happens on this sector addressing geometrical consideration, but simultaneously and sequentially on the other sectors. This translation towards the right, must preferably happen when at least the system definition already includes both seed geometry (eASG) and basic system modeling (eAMG).

#### 5.6.2.5. Towards Implementation - CDR to TRR

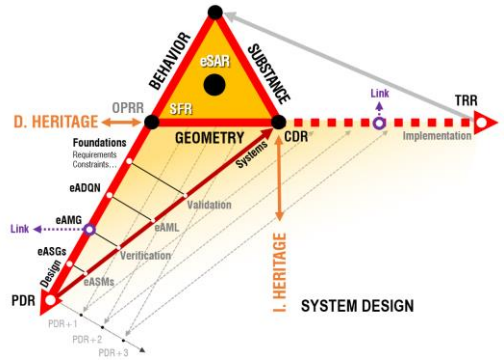



*Figure 152. Milestones and tools from CDR to TRR (eSARD).*

The final step within this sector is to prepare the transition towards both implementation (resource-driven) and operation sectors. In the beginning of these sector processes, specific heritage inputs for these sectors must also be considered. They are a great quality control tool to check the feasibility of the system design as well the influence of inputs coming from other sectors.

Along these lines, it is also relevant to highlight that links across the design network process can be presented at any point since they relate to any other step across the design effort. This process has a direction flow but inputs and links among tools and steps are concurrent and networked in nature. These links are shown with purple connections (Figure 152) representing connections among steps and milestones.

Even if this is a networked process, iterative cycles are still critical to obtain better and more refined solutions, as well as to enable more detailed designs by diving deeper into multiple disciplines. Concurrently and sequentially, these design cycles not only enable more refinements, but also a better exploration of trade, design, and maturation spaces. Thus, the more links are created the more efficient and capable the process can be. The eSARD methodology builds upon systems synergies while matures designs by using multidisciplinary connections and checks. In essence, this is what the natural evolution mechanism does when many small variations (mutations) of an organism lead to a more fitted system that is continuously been tested against the environment. However, such iterations might not happen equally across all steps and milestones. The end of a sector is the beginning on the next one, so in this case CDR efforts include implementation aspects that lead to the beginning of such sector and the test readiness review (TRR). Thus, DRR and SFR are the end of a design sector and potentially the beginning of a new design effort. The next section will elaborate a bit more these other sectors.

#### 5.6.2.6. Overall methodology

eSARD merges DE and SE efforts into one and relates to multiple gaps referenced in section 3.4 and 4.1, such as:

- **Sinergy**. This method is based on synergetic gaps (eADQNs and eAMGs) among solutions, components, and processes rather than discreet ('divide-and-conquer') traditional disciplinary methods (section 2.9 and 3.4).
- **Continuity** is enabled through a networked approach, synergies, heritage inputs, and continuous design cycles.
- **Qualification** can be tackled since this design framework allows both geometry and logical parameters.
- **Full cycle**. From design to decommission the eSARD process not only can address all phases of a project but it also encourages it by tackling gaps and links across the full lifecycle to develop a more robust design.
- **Flexibility**. Design and systems engineering methodologies are combined within a highly adaptable workflow that is tuned to the needs at hand. Constant changes are expected and even encouraged for concept validation.
- **Disruption**. Easier paths for new solutions are enabled through better connections among discreet disciplines (eAMG), geometrical-logical frameworks, heritage inputs, and the easiness to quickly change paths (Figure 145).
- **Fast pace**. To speed up this method, synergetic multidisciplinary topics (eADQNs + eAMGs) are used as starting points for the process, tackling retrospectively and concurrently more disciplines for a more complete system design.
- **Connectivity.** Not only the system design is being addressed within this networked and linked approach, but also its context, external stressors, environment constraints, and key interactions with other systems.

### 5.6.3. Implementation (Material)

The next networked sector within the eSARD tackles implementation, which refers to physical systems (manufacturing, construction, fabrication), logical system (coding, programming, development), or both. This research is mainly centered in the overall eSARD process and the system design sector. However, as an introduction there are a several relevant generic constructs within this sector that influence other DOI sectors and the overall process.

The main objective of this sector is to address the 'substance' of the system, which is how the solution goes from a thought concept to an implemented reality (physical or otherwise). For instance, how the system goes from a three-dimensional CAD model that only exists digitally, to a physical part made of steel or plastic is the real foundation here. While this approach could apply as well to software-based systems, its ultimate objective is to address complex hardware-based evolutive system architecture that also require software (coding) to address system behaviors, achieve more system adaptability, and enable better capabilities. This implementation sector deals with details such as vendor constraints, material tolerances, manufacturing tests, code troubleshooting, programming verification, UX testing, among many more areas. These tasks happen regularly, however by integrating them into the overall development process not only better and faster solutions could be obtained, but they can be better optimized in terms of cost, capabilities, and time to market.

Chapter 3 concluded that most DE and SE techniques do not tend to consider both design and systems engineering simultaneously. Within such scenario, development workflows, workforce efforts, and tools are combined so a final product can be implemented. But if such level of connection and detail is infused from the very beginning into the design process, the final system can be optimized better and faster. The evolutive approach and its ARR principles are based upon such early connection similarly to natural mechanisms. Any species is constantly getting tested against the environmental

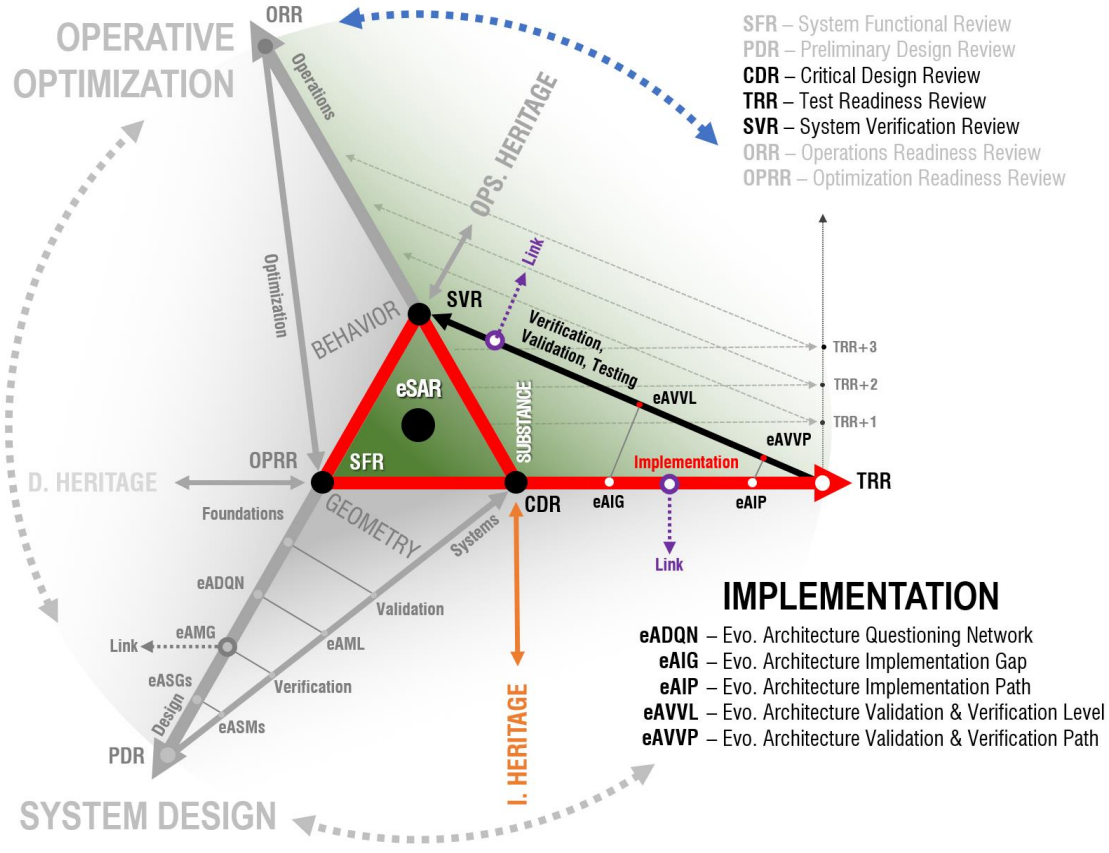



*Figure 153. Implementation sector within the eSARD helix diagram based on ARR tetrahedron.*

conditions, and any change or improvement in its adaptability can only be assessed when the system (organism) validates itself against the environment. As Table 25 introduced early, the objective of this methodology in general is to bridge areas of development (DOI) that often are not really connected in the design process. SE methods might tackle implementation aspects regarding product developments, but the reaction speed and the system capability to provoke and embrace critical change while avoiding expensive implementation issues early in the design phase, is often very limited. For instance, if cost and lead time constraints of a manufactured metal part in the system are infused early in the design process, production cost issues could be avoided more easily, and they can even improve the system design with a more robust approach. Here this is addressed by considering all disciplines at once.

This sector tackles multiple implementation steps and other detailed processes such as manufacturing, testing, Q&A, etc. Paths, tools, and milestones are identified in Figure 153. Among some of them these are the most critical:

- **CDR** (Critical Design Review) is the initial milestone. The beginning of the implementation sector is the delivery of the design and systems sector within this networked process that is defined by a spiral flow. Previously, both implementation and operation constraints were only considered at a high level with eADQNs. Thus, the feasibility and likelihood of methods, materials, and data structure among other aspects were not fully addressed yet, but now this type of details are key. Reviews only state the level of maturation that the system design needs to obtain.
- **Implementation edge** (CDR to TRR). From the CDR, the primary edge is implementation, so all tools used in the process are focused on making the system design real (physically, digitally, and virtually) including these ones:
  - **eAIG** (Implementation gaps). Overall design efforts often consider subjects such material selection, manufacturing techniques, and overall cost estimation, to name a few. But there are more key details that must be addressed and validated. eAIG tools prioritize gaps among them in this implementation path for the system.
  - **eAIP** (Implementation paths). Once key gaps have been identified, then an implementation strategy must be developed to address them. This could lead to changes in many other steps within the design sector (network).
- **TRR** (Testing Readiness Review). This is a vertex milestone pointing towards a level of maturation in which the system design is fully developed, and all implementation and development studies (e.g., materials, cost, workforce, feasibility, etc.) have been done. This process prepares testing and verification aspects of the final solution.
- **Testing edge** (TRR to SVR). The secondary edge on this sector (substance) includes these tools among others:
  - **eAVVP** (Verification and validation paths). The path towards the final verification and validation of the system addresses key gaps in these areas. At this point, this should be detailed enough from both geometry and substance standpoints. Along the way multiple eAVVP and eAVVL are be created and linked with other tools.

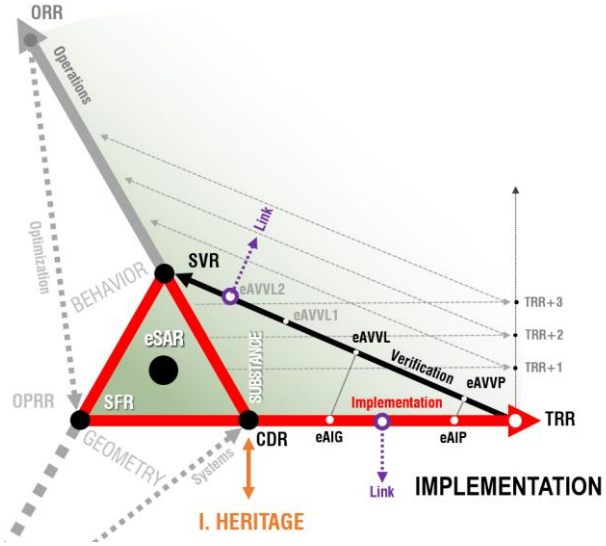



*Figure 154. eSARD incremental versions and heritage inputs.*

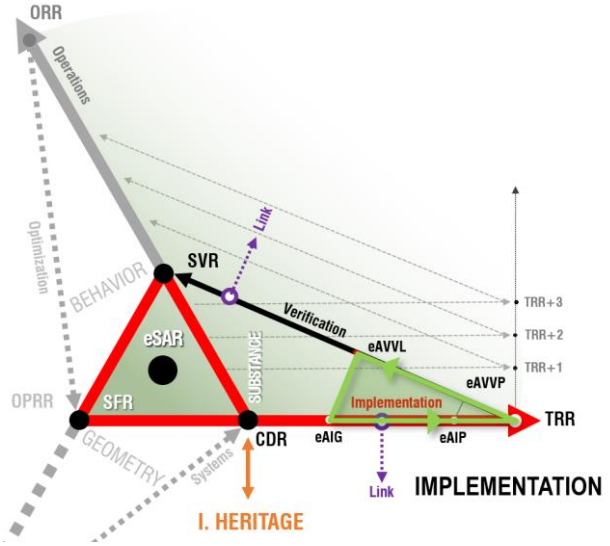



*Figure 155. eSARD verification loop (implementation sector).*

- **eAVVL** (Verification and validation levels). Just like with the maturity gaps, this is a tool that assesses the feasibility of the system, while it compares among systems, processes, and solutions. This tool implies gradual improvements in the validation and verification strategy as middle steps towards the SVR milestone.

- **SVR** (System Verification Review) is the final and critical milestone in this sector. Here the system architecture solution includes design, systems, implementation, and verification at the highest maturity level.

Regarding the eSARD helix workflow within this sector, there are six basic maneuvers, that could be summarized as it follows:

- **Sector path** (**Error! Reference source not found.**). The workflow in this sector starts with the CDR and tackles first i mplementation aspects (increasing details), while it prepares testing. TRR is the vertex milestone and here the eSARD approach addresses the verification of the system including testing, consumer studies, and many other validation tools.
- **Heritage** (**Error! Reference source not found.**, orange arrow). While at any given point the infusion and assessment o f heritage solutions is be very useful, this is especially important in the beginning of workflow to assess feasibility at the smallest scale.
- **Links** (Figure 156, purple arrows) allow to make connections among sectors, routines, tools, and milestones. They allow through different level of maturation along the process. The spiral scheme (triple vee) presents here a guide to follow and organize such processes, but these links make a critical difference in within this networked approach.
- **Verification loop** (**Error! Reference source not found.**, green triangle). This allows to check correlations among i mplementation aspects and verification tools, milestones, and levels of development at any given time in the development process.
- **Increment translation** (**Error! Reference source not found.**, grey dash lines). As previously described, different d esign strategies lead ultimately to changes in these milestones. This means that different and incremental cycles within any sector are represented by sequentially translated sectors moving towards the right side of the primary edge. Each one represents a critical variation in the sector, while tools and milestones within it could also be linked.
- **External input loops** (Figure 156, green dotted circle). Implementation processes and techniques (e.g., programming and coding) can be very complex, including many external open and close inputs. Thus, operations such as assessing manufacturing cost require independent cycles, which are summarized graphically by these loops.

After this high-level introduction to the structure, tools, and milestones of the implementation sector, there are several conclusions with regards to this sector and its influence upon the overall eSARD process such as:

- **Organization**. The eSARD approach is not only about addressing the full design process of a system, but also about tackling the culture of the organization behind it. Since the goal of this process is to enable disruption, higher system performance, and overall efficiency the 'how' is as relevant as the 'what', hence the adaptability of this workflow.
- **Early failure**. One goal of the eSARD process is to enable fast failures at both system design and detail level. This allows to find faster and more efficiently a successful and more robust system architecture design. By addressing both overall architecture trades and detailed studies simultaneously it is easier to identify most relevant gaps. Thus, this also allows to ensure the feasibility, adaptability, and robustness of a family of solutions and many of their possible future outcomes.
- **Qualification**. The processes to verify and validate single projects, as well as these families of solutions

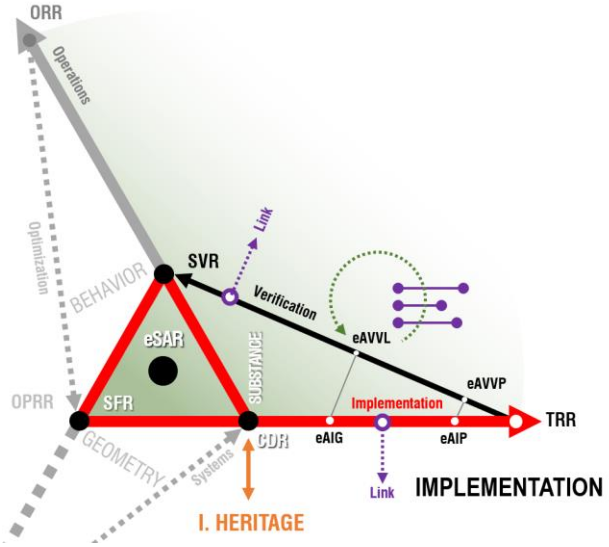



*Figure 156. eSARD external input loop (implementation sector).*

(qualification) can be very complex, especially with one-off designs and the infusion new technologies.

- **Disruptions** are enabled by the networked nature of this approach, using three interconnected sectors (vees) that are constantly being updated and linked among them. Incremental changes are addressed by more traditional design cycles (sector translations), as well as interconnection loops among all of them.

### 5.6.4. Operative Optimization (Behavior)

The last sector to be explored in this networked spiral eSARD is related to the behavior of the system and tackles directly both operations and general system optimization. An evolutive system architecture is ideally a highly adaptable system, thus its behaviors condition not only its design but the use of resources. The eSARD_he presents a network approach because all these aspects are intertwined. The operations edge of this sector addresses the following topics:

- **System functions and capabilities**. From highly reactive to passive systems, an evolutive system presents some level of integrated adaptability and smartness leading to a system that behaves or functions in a certain manner. Examples of this are the physical adaptability of a system architecture (e.g., rear active spoiler on a sport car), data created by sensors and other integrated electronics (e.g., smartwatch readings), and the interaction between systems and environment (e.g., robotic mobility platform that adapts to the terrain). So, regardless is this a fully autonomous system, a manned operated device, or highly a passive object there is an operations scheme behind the system that must be considered since affects design constraints, requirements, planning, and cost across the full lifecycle.
- **Environmental interaction**. Such operative mode is also driven by relationships between the system and its environment, which are defined by both direct interaction with it, as well as indirectly by all resources required in the process. This point affect both system operations and optimization directly and is at the core of this sector.
- **Design stressors**. As chapter 2 presented, the practice of designing complex systems must address more and more a series of key stressors affecting the nature and capability of a future system architecture (e.g., resource scarcity). This again affects how the system is operated, and furthermore it conditions its overall optimization.
- **System timeline and lifecycle**. This sector addresses all these points across the full lifecycle of the system and among all sectors simultaneously. Nevertheless, it is crucial to understand the system timeline as it relates to its behavior and capabilities, as well as the most relevant needs and challenges specifics to its lifecycle. Such temporal evolution needs to consider as well multiple system alternatives, heritage possibilities (species), and link among tools.
- **Optimization.** This is a very broad topic out of the scope and length of this research. But it is important to highlight that the optimization of the system touches all aspects and steps in the development of a complex system. However, this point is partially implemented across the process (e.g., eADQNs) from each step perspective, and later is completed as whole in the last edge of this sector. All ARR general characters and specific GBS details are tackled here across all DOI sectors. System optimization here is also continuously changing due to external conditions.

At this point, all key trades, details, and development work regarding ARR principles and DOI areas have been addressed. Geometry, substance, and behavior defining aspects of the system have also been addressed. Therefore, the next complementary step in this continuous cycle is the optimization of all three areas altogether. Such system optimization activity is related to other design studies which are often performed independently around the eSARD.

The operative optimization sector is in essence about how the system is going to perform over time and throughout different situations during its lifetime. This opens the path not only to system improvements (optimization), but also to enable a better design process that considers direct feedback during system operations. Evolutive architectures in particular and smart systems in general, are not anymore about how the system was designed when they are ‘unboxed’. They are really about the data they will generate over their use and how data can be used for the system improvement and evolution. Therefore, design processes such eSARD need to incorporate this feedback into the workflow. This helps both the system design at hand, while it closes the loop to future ecosystems, system upgrades, and overall discreet improvements based on data. Within this sector that tackles operations and optimization, several critical tools and milestones can be identified as Figure 157 presents. Among some of the most relevant elements and milestones in this sector these are critical:

- **SVR** (System Verification Review) is the initial milestone of this sector process. At this level of maturation design, implementation and validation details have been addressed concurrently in other sectors too.
- **Operations edge** (SVR to ORR) reflects activities and tools assessing behaviors that are system functions enabled

by both design and substance schemes. For example, a robotic arm will move depending on the mechanism design, energy schemes, and environmental conditions. Thus, the system behavior determines the capability of its geometry to adapt to its environment under different tasks. This edge includes among others the following tools:

- **eAOG** (Operations gaps). These tools study where system maturations are required from operations and support architecture standpoints. This also includes external workforce aspects required to assemble the system on site.
- **eAOP** (Operations paths) complement the previous point by creating solution paths and alternatives to assess and 'kill those gaps' by the operation scheme design. In essence, they are the solution approach for each eAOG.

- **ORR** (Operation Readiness Review). This vertex milestone sets the maturity level by which all operation aspects are defined, while they complement design and implementation schemes concurrently addressed in other DOI sectors.
- **Optimization edge** (ORR to OPRR). This edge relates to all optimization tools tackling operations, implementation schemes, and designs since the three of them are interconnected across the final system definition. The goal here is to assess and implement both system-level and detail-oriented optimization schemes for the system at hand. There are two main families of tools within this sector that are aligned to ARR design drivers (section 4.3):
  - **eAOPG** (System optimization gaps) highlight where the system can be optimized from a full DOI perspective. This requires external activities to assess solutions under different environmental conditions, changing constraints, etc.
  - **eAOPP** (System optimization paths). Similarly, these develop a solution scheme to address each eAOPG.
- **OPRR** (Optimization Readiness Review). Final milestones within this sector establish a level of maturity and optimization where the system has been stressed to the maximum. This is also directly related to the very first milestone (SVR) since they both close the full cycle of the system. As such, OPRR can be understood as a direct input for new SVRs towards future systems or variations within the same family of solutions.

Within this sector, possible workflow maneuvers are the same as those described in the previous section. Among others these include (Figure 157) heritage inputs (orange arrow), input loops (blue dotted circle), verifications loops (blue triangle), and increment translations (grey dash lines). These last ones could be related as well to major changes in the system design due to external factors and system behaviors across different chronological phases in the lifecycle.

Finally, this last sector brings up several key conclusions about the eSARD process in general. These are specific to this sector and unique to the evolutive design methodology (eSARD_he) approach, including the following one:

- **Connectivity**. When assessing operations, evolutive solutions may present a higher level of connectivity with their environment. This includes the use of sensors, data structures, communications protocols, etc. In essence, the external environment influences the system design, its behavior and performance.
- **Reactivity**. An evolutive solution is dynamic, thus the process should implement the same level of flexibility towards its functions and other aspects across all DOI areas.
- **Evolutive ARR**. All these activities happen simultaneously. Moreover, they all reference each other while advancing the development of their specific maturation. So, even if the graphic presents a 2D diagram, this process follows a three-dimensional spiral when all multiple solutions, situations, and levels of maturation are considered within the eSARD process.

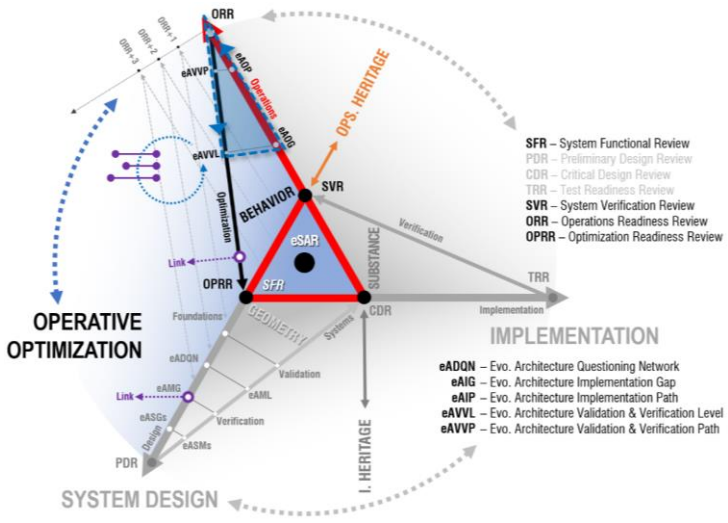



*Figure 157. Operative sector within the eSARD helix diagram based on the ARR tetrahedron.*

### 5.6.5. Design Verification Loops

Verification loops presented in Figure 158 showcase the rationale behind all connections among tools (design activities) and milestones (maturation levels). At any given time, multiple of these links can be referenced to check parameters, relationships, constraints, etc. This graphical representation means that models such as DE, SE, Implementation, Ops-Con, etc. will share data and interact among them to accommodate key design changes. At the end of a full iteration cycle, all milestones (both critical and vertex) describe the maturation level of the system (Figure 159) from a DOI perspective, while they define the solution towards all ARR needs. This helix scheme tackles the complexity of an evolutive system, while it also offers a platform to further the design, detailing, and scope of such initial connections.

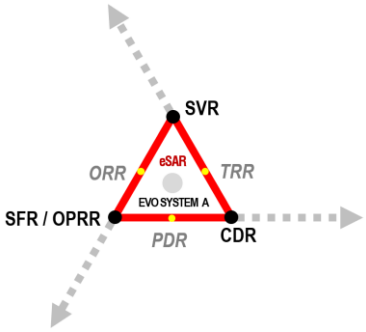



*Figure 159. Summary of key ARR system design milestones in the eSARD process.*

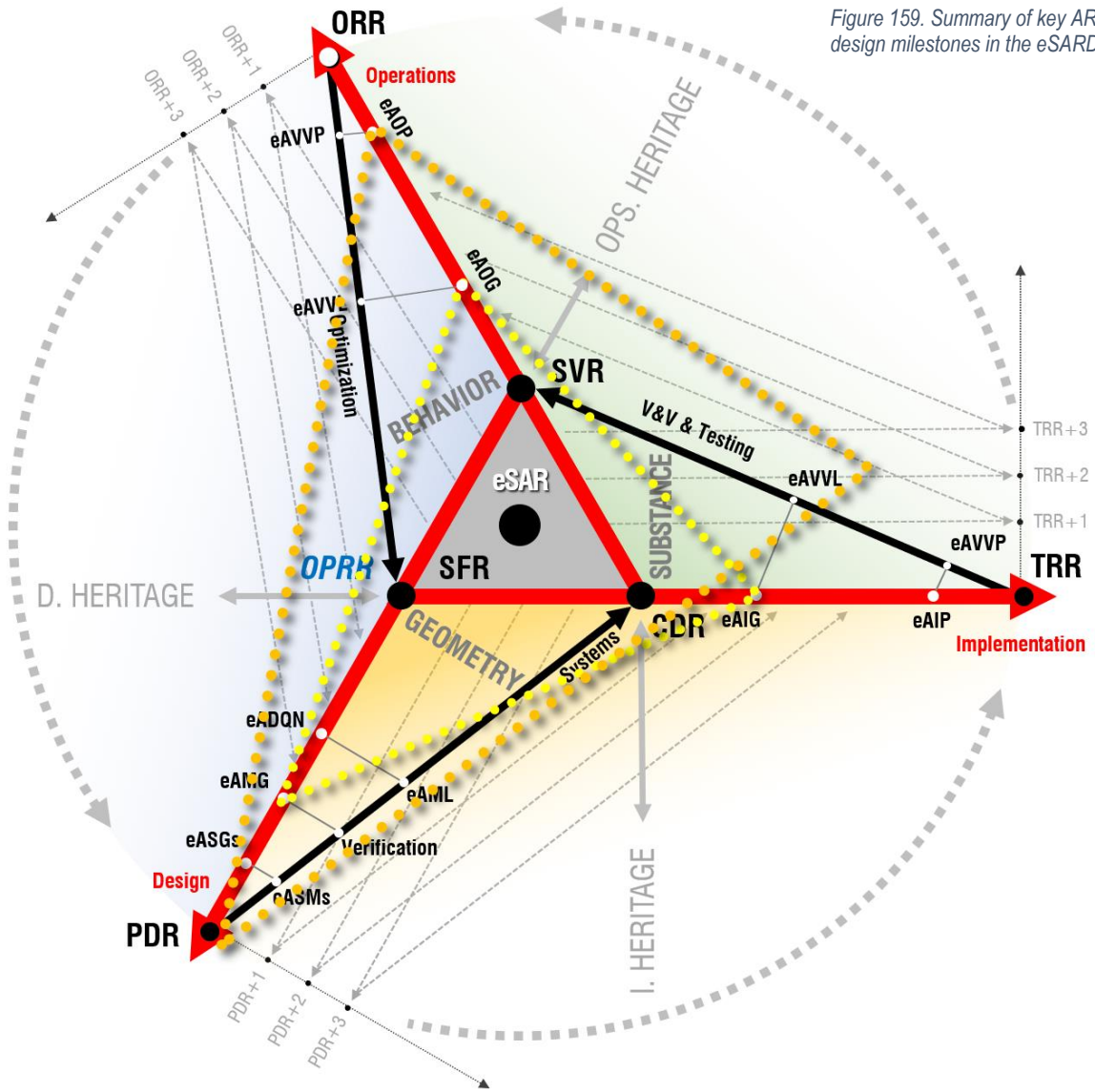



*Figure 158. Multiple verification loops within the eSARD_he representation of an evolutive system design activity.*

### 5.6.6. eSARD_he 3C Framework (Concurrent, Collaborative, Communicative)

Communication is one the most critical aspects within an evolutive design methodology. Following a communication theory approach (Sekhar, 2005), information is transferred among people, tools, and models using a process that is often driven by a disciplinary perspective. In that regard, it is crucial that the right amount of information at the right speed can be transferred properly and bidirectionally, even more if the process being served has a networked nature. Thus, the eSARD process not only tackles how tools should talk to each other, but how professionals and teams visualize, share, and interact among each other regarding critical design topics. Therefore, this is not only about the data rate, but its quality and the facilitation process that is also required. An evolutive methodology (Figure 160) can have virtual, physical, and hybrid workflows and workspaces. This requires an information infrastructure based on three main practice points (3C) such as:

- **Concurrent.** Any model (mathematical, conceptual, etc.) and discipline activity (e.g., thermal design, mechanism definition, architecture layout, etc.) need to be developed concurrently and bidirectionally, exchanging at least key parameters with regards to most relevant figures of merit for the system at hand. This applies to both computer models (e.g., data-driven analytic models such as databases, excel, etc.) as well as to professionals sharing a common source of information truth. Multiple techniques have been developed over the years for product design, aerospace, etc. (Eastman, 2012). In an analogy to the music industry, these are multiple instruments sharing a common but also constantly under development score. In Figure 160 this is shown as grey bidirectional lines, between multiples disciplines and agents (e.g., D1), as well as common in-development scores (black line boxes). This is also done traditionally in concurrent engineering where key parameters and figures of merit are captured in a general model that is later shared among interconnected agents. Thus, critical data must be captured, transferred, and shared, in a way that can be curated and utilized. The starting point of this method begins with a system architecture sketch underlying basic geometry, subsystems, and even initial system behaviors. This will be tackled in detail in following points, but it requires a clear and potentially evolvable framework to capture and define objectives, goals, constraints, requirements, and designs, among others critical topics.
- **Collaborative**. Not only models need to be connected among them, but the design process itself is being co-authored as well. This means that all activities and disciplines involved, should advance the design or development in parallel while they relate to each other in the process. Collaborative methodologies are a growing tendency in advance design nowadays (Safavi, 2016), because in combination with other approaches they can be very powerful. Following the music analog, besides a common score, a live and constant revision of the activity of other musicians is needed as well, just like a jazz jam session uses instant feedback among players. Constant changes, interactions, and directions need to have a media to be captured as well. In Figure 160 these are represented by red dotted lines.
- **Communicative**. Connecting models and sharing information is critical. However, sue to the potential complexity of the system architecture at hand there must be a curated and facilitated guidance toward what to communicate and how to do it. This also includes the design path within a full trade space. In previous examples, if models and people are like instruments in an orchestra using a live and changing scored while becoming aware of the interaction among musicians, there must be also one or several orchestra directors guiding the process. The goal of this last point is in essence to provide a mechanism to reject non valid paths quickly, create new ones, and validate 'on-the-fly' other alternatives. In other words, this is about communicating properly to fail fast and thus achieve better results faster. The system architecture keeps evolving and changing quickly based upon such dialog, which is a tension between discipline inputs and a facilitated guidance. However, the solution is neither unique nor static. In Figure 160 this is conceptually represented by a meandering yellow line with multiple dots that represent different evolutive states of the system. Among other goals, this approach aims to quickly obtain a good solution within a family of solutions.

Beyond these pillars there are other critical aspects that need to be developed for a successful eSARD environment. However, these are outside the scope of this research and part of an ongoing research. These are the summary highlights:

- **Workforce team dynamics**. How teams, personalities, and backgrounds interact is critical for the success.
- **Tool agnostic approach**. The eSARD approach is independent from any tools or technique, software or otherwise.
- **Multiagent.** This 3D framework connects designer, team, and machine-oriented hybrid environments.
- **Scalability.** This process is designed to be scalable and adaptable to both new design objectives and capabilities.
- **Culture.** A part of the associated field of research is about how to infuse and merge new techniques.

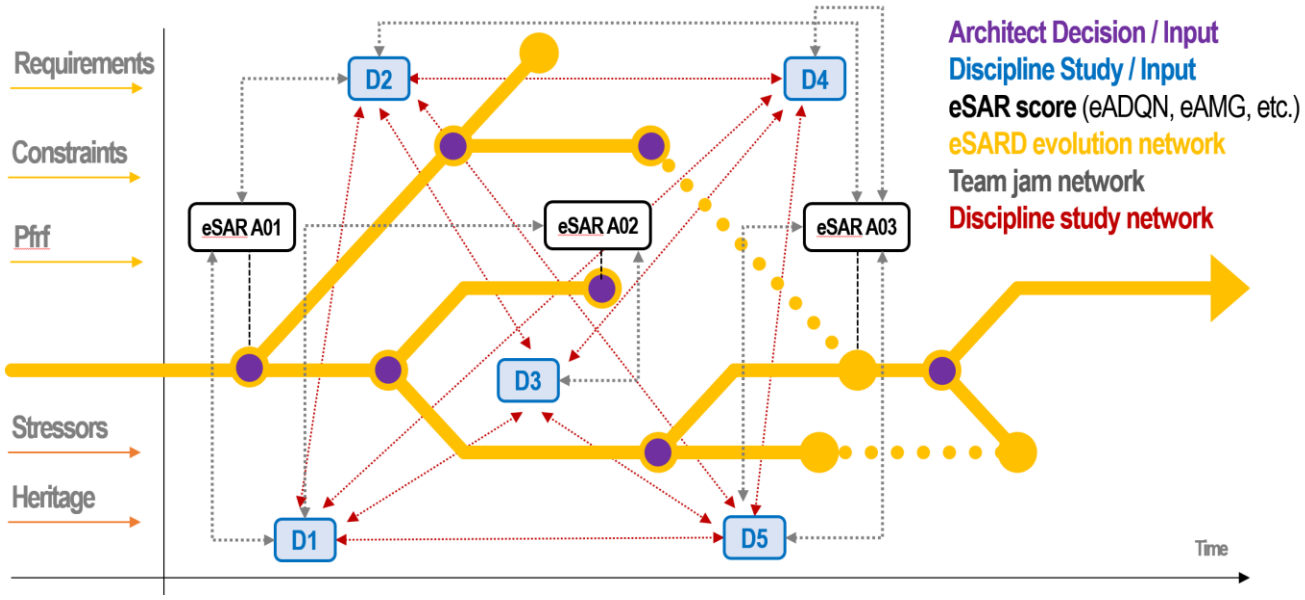



*Figure 160. Workflow within the eSARD 3C framework including disicipline inputs, workforce activity, and eSAR develeopment paths.*

## 5.7. Evolutive System Design Workflow in Detail

During the first part of this chapter principles, objectives, and general procedures for an eSARD method have been introduced and defined. The goal of this methodology is to support the development of complex system architectures in general, and specifically address all special ARR characteristics of an evolutive hardware-based system (chapter 4).

The rest of the chapter is dedicated to study in detail some these tools that were highlighted in the eSARD approach. Since the objective of this research is to concentrate on system design aspects, next sections only develop in detail those tools directly related to such sector, although the approach is quite similar in other sectors. The evolutive design process has a networked nature, and all tools and maturation activities within each sector must happen concurrently while sharing bidirectionally information among them. Once more, the goal here is to efficiently create a system that addresses a balance among all GBS details, tackles all DOI areas, and considers all ARR principles of a complex evolutive system architecture and all its subsystems and components.

As section 5.6 described, within the system design sector the maturation of the system architecture goes through different milestones, reflecting incremental levels of detail, feasibility, and overall evolution of the system such as SFR, PDR, and CDR. These milestones also serve as a reference to assess how much of the system definition needs to be developed, but they are not a design tool by themselves. In the context of an eSARD process, these do not necessarily mean they are a formal review, but rather a checklist of sorts. Section 5.6 described a series of specific tools, routines, activities, and exercises created for an eSARD approach, which are shared with other SE, DE, and DSE methods. These tools can help designers, teams, design workflows to achieve those milestones more efficiently, but they are never static in this context.

Within the system design sector (yellow) there are several groups of tools that allow the following actions and goals:

- **Design objectives**. To design efficiently the eSARD approach is based upon making the right question to achieve better solutions faster thought synergetic connections among requirements, disciplinary studies, subsystems, etc. Identifying seed questions that make a system architecture feasible is therefore the goal of these tools:
  - **Evolutive Architecture Dynamic Questioning Network** (eADQNs) identify key gaps in the design process.
  - **Evolutive Architecture Maturity Gaps** (eAMGs) establish the design starting point due its criticality.

- **System design**. This is the actual detailed design including all DOI necessary elements. Evolutive seed architecture geometry (eASG) is the first and most important tool in this phase complementing other DE techniques.
- **Evolutive architecture seed model (eASM)** is a logical modeling of the system complementing eASGs by pairing SE models with geometry under different DOI perspectives. This also complements other DE techniques.
- **Evolutive architecture maturity levels (eAML)**. Previous tools are engaged in multiple 3C cycles while interacting with each other under such networked perspective. Thus, either manually or using automated workflows, the definition of the system follows different DOI views that will keep increasing. Maturation and design progress allow assessing, managing, comparing, and directing all design efforts from both technical and managerial perspectives.
- **Inputs**. As section 5.6 described there are multiple inputs within an eSARD design process including the following:
  - **Requirements and constraints** (primary) are present in any design process. They can be open or close (fixed).
  - **Heritage** inputs include previously validated solutions, methodologies, technologies, components, etc.
  - **External inputs** are specific to the system architecture or the field of development (e.g., workforce topics).
  - **Implementation** inputs include previous or concurrent topics of this sector that can precondition the design.
  - **Validation, verification, and testing**. Similarly, V&V & tests need to include user experience, special technical needs, risk mitigation strategies, and redundancy approaches, among many other topics.
  - **Operations** critical inputs include all inputs from the DOI sector that can or will affect the system design.
  - **Optimization.** This set of activities in the design process is directly related to all other aspects across the system GBS. The goal is finding a balance among opposing design forces, prioritizing attributes (e.g., performance), identifying hidden figures of merit, reducing risk, and managing complexity, among others. These require a dedicated chapter outside of the scope of this thesis, but chapters 5 and 6 will address some basic foundations.

Table 27 shows below a summary of such tools, groups, and steps within the DOI design sector that next sections elaborate in detail. These are steps within the eSARD model that serve as a general design platform towards evolutive design methodologies and other future developments in general.

| eSARD - Evolutive System Architecture Design Methodology | | | | | |
|---|---|---|---|---|---|
| | **eADQN** | **eAMG** | **eASG** | **eASM** | **eAML** |
| | *Evolutive Architecture Dynamic Questioning Network* | *Evolutive Architecture Maturity Gaps* | *Evolutive Architecture Seed Geometry* | *Evolutive Architecture Seed Model* | *Evolutive Architecture Maturity Level* |
| **Objectives** | Requirements<br>Assessments<br>Heritage<br>Trade space<br>Design space<br>Gap identification<br>Design paths | Feasibility<br>System closure<br>Comparison<br>Optimization<br>Foundation | Maturation space<br>Geometrical relationships<br>Optimization foundation<br>Compatibility | Non-geometrical relations<br>Parametrics<br>Compatibility<br>Optimization<br>Multiphysics | Evaluation<br>Comparison<br>Management<br>Organization |
| **Tool Types** | Pen & paper<br>Mind-maps<br>WIKI<br>Discussion Capt.<br>Other | WIKI<br>Knowledge Mgmt.<br>System models<br>Databases<br>Other | Hand sketch<br>Evolutive sketch<br>Parametric CAD<br>Generative CAD<br>Other | Adaptable SE<br>MBSE<br>Data bases<br>Other | WIKI<br>Visuals<br>Knowledge Mgmt.<br>Other |
| **Product** | Design objectives | New design requirements | System geometry design & develop. | System model design & develop. | Assessment Validation |
| **Base for** | **eAMG** | **eASG** | **eASM** | **eAML** | **Design Cycle** |

*Table 27. Summary of most relevant eSARD tools and models within the DOI system design sector.*

## 5.8. Dynamic Questioning Networks (eADQN), Exploration and Foundation

Understanding critical initial needs and requirements of a system architecture is a necessary step in any design methodology. Customers and technical efforts provide requirements and stakeholder expectations at different levels. However, as detailed as those might be, within an evolutive process they should always be considered as open in nature, since this methodology is based upon underlaying DOI interconnections among them. Under this light, it is assumed that each complex system has a series of key attributes and technical characteristics at the core of any feasibility and maturation activity. Such links can be upgraded or improved at any time by newer connections, complementary heritage solutions, or new disruptive design paths. These are technical design principles, often multidisciplinary in nature, that enable and fulfill the system architecture teleology. Here are some simplified examples used as an introduction towards what an evolutive design approach is addressing to maturate a complex hardware-based system architecture:

- At the core of a radical new architecture for a formula one car trying to achieve a better system performance next season, it might be the mass reduction of joint components using topology optimization techniques. It might seem the key question here is to reduce the mass of every component, using such new technique. However, it could also be that the combination of a simplification in the number of components, the easiness of integration, the manufacturing response, and the structural stiffness is what really holds the key. Achieving this solution then does not respond only to a mass reduction approach, sparking new feasible solutions that require to bound the research differently. So, the question is now how such disruptive solutions can be done with a practical design workflow.
- For an emergency shelter on a desertic area to be self-sustainable it must improve its energy management. However, rather than just carrying solar panels, ac-units, and batteries because portable insulation capabilities are limited, the answer could be about finding a design approach that improves thermal inertia and passive energy management with less components. Such system architecture is enabled by a multidisciplinary technical approach that drives all system design efforts. However, proven heritage solutions are not clear unless they come from other fields. Thus, one question here is how to handle the lack of heritage when creating a validated ultra-performance new system design.

The exploration of gaps and the generation of design knowledge related to the system architecture at hand is the first step towards developing a feasible and complete evolutive design strategy. eADQNs are at the core of it.

### 5.8.1. Nature and Definition

Evolutive architecture dynamic questioning networks (eADQNs) are a series of inquiries regarding requirements, expectations, heritage, and preliminary design paths among other topics to understand key gaps in the multidisciplinary maturation of a system architecture. These questions poke at architecture ARR needs considering design, behaviors, and substance areas at a system, subsystem, and component level. Such needs are required to have a complete, mature, and feasible system solution. But these do not provide design solutions, they rather identify gaps the design process must developed later. This first construct within an evolutive system design process (eSARD) has the following characteristics:

- **Driven by disciplinary gaps (DOI)**. These inquiries start from a disciplinary standpoint addressing design needs and gaps in the system. For instance, how does the system manage heat? What type of structure would it have?
- **Networked nature**. These gaps are interrelated in the context of a system architecture design, as previous sections explained. The deeper those gaps are, the more detailed those questions are, and the more multidisciplinary they become. For instance, manufacturing techniques affect the design process directly, which also affect the number of components and condition the integration of such system. These details affect all GBS fields and topics.
- **Broad spectrum**. eADQNs tackle all ARR areas affecting an evolutive system architecture and all DOI topics.
- **Structured**. These inquiries are organized around the three GBS pillars describing any complex system architecture.
    - *Geometry* (descriptive principles) includes questions addressing structure, volume, size of subsystems, design principles, configuration, etc. For instance, what subsystems does it need? How small can they be? What type and shape of structure does it need? How compact is the system while being stowed?
    - *Behavior* (operative principles) relates to inquires addressing how the system works and what it needs to do so. For instance, what function does it perform? How does it identify the target? What is the control approach?
    - *Substance* (component nature) finally addresses the materialization of the system. Questions such as these are

key: what materials is it made of? What coding language will be used? What kind of manufacturing technique is used? How does this affect the integration? Does it provide enough chemical compatibility?

- **Dynamic**. None of these questions are static nor constant in time or importance. During the design process and the creation of this inquiring network some gaps will be solved or addressed, giving less priority to all associated questions. At the same time, the more in depth the design process goes, the more connections will appear defining new questions and sub-questions consequently.
- **Variable depth**. All inquiries with an eADQN will refine details through derived sub-questions and refinements of the question itself. For instance, the first round of questions might address general manufacturing techniques while at the end of this network process (DOI sectors) they could only tackle specific alloys, post-production methods, etc.
- **Variable nature.** The type of connection between questions might also vary over time. For instance, an initial structural relationship addressed by questions might evolved into a thermal-driven approach. Furthermore, these inquiries can address requirements that are quantifiable, qualifiable, or both. These do not have to be purely alphanumeric in nature, including other sources such as graphics, images, videos, 3D models, etc.

Therefore, such inquiries are meant to question the system architecture design as the objective of an evolutive design process. They do not provide solutions, they are used to point out through multiple design cycles, the most critical gaps among all topics required for the system solution to be feasible and implementable. These mark the beginning of the design process, considering the following areas:

- **Stakeholder expectations** include needs, wishes, non-technical constraints, etc. (NASA, 2007).
- **Requirements** at any level should be considered as open (primary and secondary) within the design process, since one of the objectives of an evolutive approach is to exceed them in terms of performance or heritage.
- **Heritage** includes direct heritage (e.g., older versions or generations of architectures with similar purposes) and indirect heritage (e.g., subsystems used in other designs), as previous sections already presented.
- **Preliminary design paths**. While the process should not be bias towards a specific approach, high-performance organizations and individuals might quickly form initial ideas about the path to follow, which can be problematic.

All these points could be used to question system requirements as well as to identify key DOI areas.

### 5.8.2. Objectives

Thus, the objective of eADQNs is to explore concept, trade, and maturation spaces with a network of interrelated inquiries. These are oriented towards finding critical gaps in the feasibility of a system architecture. Specifically, such objective can be summarized as a combination of the following functions:

- **Identifying critical synergetic links across disciplines**. The evolutive methodology is based in finding links among multidisciplinary gaps driving the feasibility of a system design and its concept. Instead of a 'divide-and-conquer' approach the goal within this research is to find the most critical connections, which within a complex system are always multidisciplinary in nature. eADQNs allow to constantly map the context where the design effort takes place, as well as to focalize the beginning of a design networked process. Therefore, this function is about finding gaps and links between gaps through questions without considering any design solutions yet.
- **Establishing problem limits and metrics**. These inquiries allow to characterize design challenges and initial solutions based on all identified gaps. They also provide the bases for new metrics behind the system performance. The more critical these inquiries are, the more a design effort is defined, and the easier it is to evaluate a solution against requirements, expectations, and heritage. Nevertheless, to simplify the process these metrics are set up against key figures of merit, which are derived from the study and exploration of the design challenge at hand.
- **Prioritizing eAMGs**. Since these inquiries address subsystems, components, and system characteristics that are interrelated among them, they also allow to understand which maturity gaps are more critical. Ideally, at the deepest level of system description, one or more critical gaps represent the difference between being able to obtain a working and feasible system architecture against requirements or not. Creating a design strategy that offers feasible solutions to such critical gap most likely will have a ripple and multiplying effect through systems, subsystems, and components.
- **Promoting disruption.** A feasible system-level architecture is indeed the end goal of the process. While the evolutive approach is based on synergies, the objective of questioning networks is to identify gaps where those synergies could

make the biggest impact. At the same time, looking into conceptual subsystems or parts for the system architecture allows for newer and disruptive ideas to be infused more easily. Such new components can also be validated more easily by using indirect heritage solutions and technologies coming from other fields.

- **Developing an open framework.** Finally, there are no strict rules to create these questions networks, neither what specific tools must be used. The objective is not the network itself, but the exploration of the system idiosyncrasy considering quantifiable and non-quantifiable aspects, which is influenced by the type of system and its context.

Thus, the main objective of eADQNs is not to provide solutions, not to map all the subsystems and requirements, but rather to identify critical gaps from multidisciplinary and holistic standpoints. Furthermore, this approach allows to enable new design paths by looking at synergies among the idiosyncrasy and morphology of the system architecture and its concept.

### 5.8.3. Foundation

Key design methodologies towards the innovation of complex systems include TRIZ theory (Terninko et al., 1998), which was developed by Russian engineer Genrikh Saulovich Altshuller (Altshuller, 1984). This method is based on the identification of 40 systematic system principles (Figure 161) and some balanced contradiction principles among them (Altshuller, 2002). Furthermore, system evolution laws addressing key elements, energy transmission, and component rhythms (Zouaoua et al., 2015) are critical as well. However, such approaches also present limitations when parametric and multidisciplinary connection must be considered (Fiorineschi et al., 2015), even is OSTM-TRIZ methods are used. This last one uses networks of problems (NoP) to map problems, solutions, and contradictions based on TRIZ principles, which are used to divide a complex problem into parts. This approach also presents limitations towards considering heritage needs as well as enabling new disruptive connections. This is relevant to mark the evolution of this approach towards addressing efficiency, as new methods like TRIZEE show (Sheng and Kok-Soo, 2010).


*Figure 161. 40 principles of the TRIZ method. (FotoSceptyk, CCA 3.0, 2016)*

Similarly, systematic inventive thinking or SIT approaches problem solving and idea generation with a method that pursues easier and less complex techniques (Horowitz, 1999) by focusing on a close-world condition. If all components required for the new concept to be created are identified, the final process under this approach becomes a matter of putting the puzzle together. The development of eADQNs follow a similar principle by poking at both the maturation space and all required gaps for a feasible new disruptive architecture, however this is not a close approach so the infusion of new ideas and changes at any time is enabled and encouraged.

The foundation of the C-K theory (Hatchuel and Weil, 2002) is more flexible than TRIZ and presents a new dialog between concept and knowledge spaces enabling 'crazy concepts'. In other words, it allows to bring new knowledge into the process through conjunctions (Hatchuel et al., 2004). This is successfully applied to product design. In this context eADQNs point out towards both c->K and K->C operations within this theory. However, in the evolutive approach both questions themselves and all connections between them point out to both C-K operations, while the rest of the methodology allows to

tackle and provide solutions at any multidisciplinary level to deliver a working system at the end of the process.

At the same time, axiomatic design methods (Farid and Suh, 2016) could be applied to the definition and exploration of such descriptive questions towards more simplicity and reliability in the system design. However, there are fundamental differences between these two methods. Questioning networks within an eSARD approach are developed to find synergetic connections among any type of requirements, even if they are independent from each other. Axiomatic methods specifically present complications to study non quantifiable requirements as well as to find alternative solutions for complex systems designs. This is the same case for other statistical methods such as Taguchi Methods (Nair et al., 1992) and Six Sigma (Pande and Holpp, 2001), which can be used to complement the creation of questions as well as to refine requirements.

In many ways, the evolutive design approach uses similar design research methodology methods (Blessing and Chakrabarti, 2009) to DRM, since eADQNs provide a framework for the design activity that helps focusing the effort. While the evolutive framework remains open towards the infusion of all these multiple techniques, the purpose of eADQNs is not to define success itself (requirement) but to address what is missing for the system architecture to 'close'. Nevertheless, success factors are connected to maturation gaps, and so are the network of questions to the evolutive reference model since they are all tools to achieve system maturation. Thus, these are processes in DRM terms applied to evolutive hardware-based systems. DRM is a recommended approach especially toward the refinement of requirements and figures of merit behind any complex architecture, but this is not part of the scope of this research.

The evolutive approach is similarly to Zwicky's morphological design, but it is not reductionist method. Thus, it is not trying to break something complex into parts, but rather embrace the complexity of the system by looking at its relationships. Cross-consistency assessments (CCA) could be used in order to address non-quantifiable relationship (Ritchey, 2002) among inquiries that address subsystems, requirements, and figures of merit eliminating illogical combinations. However, the evolutive approach provides a structured framework, which is especially designed towards a given architecture and allows to use geometrical descriptive questions, as well as numerical, alphanumerical, and even graphical descriptions.

This approach is tool agnostic, and Model-based system engineering (MBSE) techniques are an ideal complement to capture the logical model behind such dynamic questioning. This is especially relevant if both quantifiable and qualifiable variables must be captured. Languages such as SysML™ (Friedenthal et al., 2008) among others, allow to capture both types of requirements, as well as structural, behavioral, and other parametric aspects of these questions. However, these tools tend to be complicated to apply when geometry (quantifiable) and aesthetics (qualifiable) are also involved.

The development of eADQNs, just like the rest of the evolutive approach, provides an open methodology that can partially integrate other design research methodologies within it. The following points elaborate this method in detail.

### 5.8.4. Trade, Concept, Knowledge, and Maturation Spaces

Multiple design engineering methods encapsulate the design process into a dialog between concept and knowledge space (e.g., C-K theory, Hatchuel et al., 2004), or between the expected, external, and interpreted world (e.g., FBS ontology design methods, Gero and Kannengiesser, 2004) to name a few. In essence, the framework for the design process is defined in between what we would like to accomplish, and we can do. In other words, it is in between what we can describe and what we can measure. The goal of this evolutive approach is not necessary to provide a universal design methodology, nor a description for ontological quantitative design protocols (Kan and Gero, 2017), but rather to provide a highly adaptable and fast-paced design framework that is especially oriented towards hardware-based systems [1] driven by ARR principles, with [2] little heritage, and [3] pursuing much higher system performance levels. This approach as previous sections described is based upon synergetic multidisciplinary gaps within a framework to conduct such activity. Therefore, several operative and conceptual spaces could be identified in the design of a complex system using this method such as:

- **Concept space / Expected world**. This is the area where ideas exist and the realm within the designer's mind. It is made of logical fixed parameters and predominantly non-quantifiable variables. This is the function field within Gero's FBS approach (Gero and Kannengiesser, 2004) and the system teleology. This space is about what it could be.
- **Knowledge space / Interpreted world**. This is the area of quantifiable parameters with different sources truth and rules derived from design efforts. It is the behavior realm within FBS models, and the space of what has been done.
- **Trade space** encompasses all likely design possibilities for a complex system architecture, which are balanced among a selection of critical characteristics that often are in contradiction with each other (e.g., TRIZ). This is the

area explored by brainstorming activities (De Bono, 1993) and being studied by systems engineering methods across many technical fields (NASA, 2007) worldwide. This is the world of trade-off options and possibilities.

- **Real space / External world**. This area represents reality, which is outside both concept (designer vision) and knowledge spaces. This area is highly connected to the trade space through proven heritage solutions. This is the world of what it is, as well as the realm of the empirical and proven heritage across all scales related to the system architecture.

However, there is another area in between concept and knowledge space, which is reinforced by the real space and balanced by the trade space. This is the maturation space proposed by this research (Figure 162) where the evolutive method takes place.

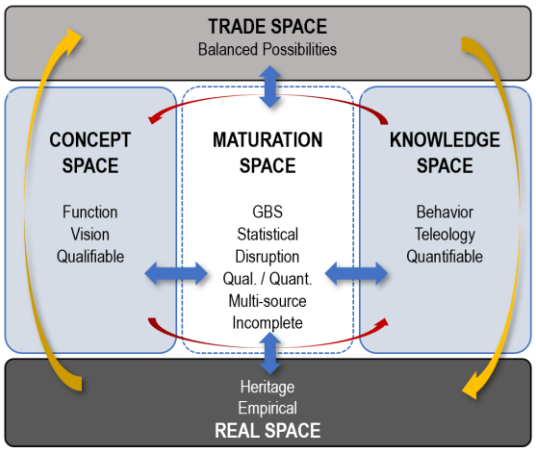



*Figure 162. Evolutive maturation space.*

**Maturation space** is the fluid junction between all the above spaces. This is the area where all gaps serve as a foundation to turn a system architecture into knowledge, and all visions, trades, and heritage solutions that are used to complete them coexist together. Therefore, maturation space is the realm of uncomplete, statistical, disruptive, and innovative elements. In this space where the designer balances vision with options (often unexpected), and the design process addresses both quantifiable and quantifiable variables. eADQNs and maturity gaps (eAMGs) reside within this space. These are variable and less permanent from a more classical design engineering perspective. Thus, maturation space is chaotic and adaptable by nature. Hence, a design effort within this space is not about completeness, on the contrary it is about finding design keystones necessary for the system concept to mature. This ensures system feasibility in the real world with the backup of heritage solutions (if they do exist) and trade-off explorations, following a vision based on system architecture objectives.

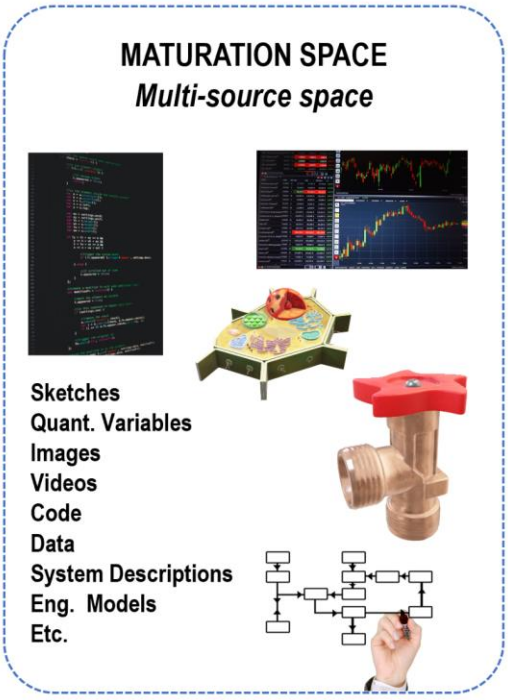



*Figure 163. Maturation space as a multisource information and design framework.*

The development of eADQNs is addressed on a maturation space where eAMGs exist. Hardware-based system architectures without heritage and pursuing dramatic improvement in system performance require new approaches to address critical technical gaps. Thus, the goal of design inquiring efforts is not to have a fully defined system, but rather to find critical system gaps in a fluid way.

It is also important to remind that an evolutive approach has three main nodes within a networked approach: design, optimization, and implementation (DOI). These can influence initial assumptions or deign paths, so they are key in a transitional environment.

The current increase in data volume and new types of information is also important towards the consideration of such space. Nowadays information affects both designer decisions and driving algorithms through quantifiable data, images, videos, sketches, technical descriptions, CAD models, and MBSE models among many more sources and concurrent efforts. However, many tools and frameworks today can jeopardize any approach that is not adaptable enough (Figure 163), since the speed at which digital information and workflows advance is quite high. Furthermore, since agility is a design stressor addressed by this evolutive approach, it is often not possible to format and normalize such sources of data and information. Therefore, the fluidity of the design framework is critical when it comes to data sources. This is something that eventually will be automated, yet it remains tool agnostic from a methodology perspective.

## 5.8.5. Questioning Approach

The approach towards developing initial eADQNs within a maturation space can be summarized though the following points (Figure 164). A more detailed example is provided in chapter 6 for a more extensive explanation.

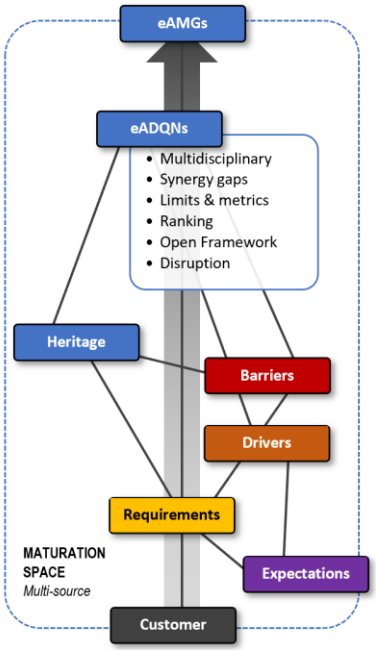



*Figure 164. eADQNs workflow within the eSARD evolutive design node.*

### 5.8.5.1. Challenge

The first step is understanding **objectives, goals, and initial challenges** for a design effort. This requires interactions among customers, representatives, stakeholders, and designers. It is also crucial to address initial maturity points that must be captured initially. This conditions all next steps in the process and enables the methodology to be properly addressed under ARR precepts.

### 5.8.5.2. Requirements, Expectations, and Heritage

Design requirements, customer expectations, available heritage information, and initial design paths are the beginning of this methodology, but it is not part of the scope of this research to define them. Plenty of publicly available bibliography explains how to address them, for instance the NASA SE handbook (NASA, 2007). Nevertheless, it is important to note a series of key points to be delivered and properly captured within an eSARD approach such as:

- **Requirements** could be quantifiable, qualifiable, or both. Thus, any approach must consider that not only analytical answers are going to be managed, since requirements for a complex system architecture need to address all major disciplines involved in the process. As a reference Table 28 shows some basic areas requiring further definition for a generic electromechanical system. Not all of these may be known or even available when dealing with radical new architectures, hence the objective of eADQNs is to find initial gaps within them as a preparation for future events. Any requirement provided within an evolutive effort must be considered as open since it may vary along with the advancement of the design process. For instance, initial power consumption can be eventually surpassed by a new and more efficient design path.
- **Stakeholder expectations** must be considered many times as non-quantifiable variables, which can also tune any provided requirement. These are critical to understand the context of a system architecture, therefore without them such solution cannot be successful (Eder and Hosnedl, 2010). The main goal of an evolutive process if to achieve system feasibility, hence expectations such as the concept of operations, initial technical expectations, product support strategies, and measurements of effectiveness and success, among others are critical in this endeavor.
- **Preliminary design paths** can be considered as well as a subset of any system expectations by either the customer or the designer. While the activity in the design sector of the eSARD process starts once initial gaps are identified, it is important to see their relative value. An experienced designer familiar with the topic and a strong technical culture within an organization most likely might have some preliminary ideas about how to proceed. Under this approach these must be used to question such initial thoughts and find missing gaps among them in the maturation space.
- **Heritage** inputs can be critical, but they are never a restrictive component in an evolutive process. Heritage provides validation points that help addressing gaps and feasibility at both system and subsystem levels. However, radical, or disruptive architectures might not necessarily have direct and useful heritage as a building block for a new system design. There are two types of heritage being considered within the evolutive approach such as:
  - **Direct heritage** relates to the full system architecture by either scope or behavior. This means there is a possibility to measure system effectiveness. For instance, a combustion engine car can be considered as direct heritage for an electric vehicle, since it provides a useful power mobility reference.
  - **Indirect heritage** relates to subsystems and component levels independently. For instance, this can be a type of valve used in the oil industry that has a similar application towards developing a hydroponic food farm.

| | REQUIREMENTS | | | EXPECTATIONS | HERITAGE |
|---|---|---|---|---|---|
| | GEOMETRY (G) | BEHAVIOR (B) | SUBTANCE (S) | Stakeholders | Direct/Indirect |
| Architecture-based | Layout [qual] | Modes [qual/quant] | Exterior materials [qual/quant] | Style [qual/quant] | UX [qual/quant] |
| | Shape [qual] | | | | Design [qual] |
| | Volume [quant] | Deployment [qual/quant] | Interior materials [qual/quant] | | |
| | Configuration [qual/quant] | Packaging factor [qual/quant] | | | |
| Mechanical-based | Mechanism design [qual/quant] | Deployment [qual/quant] | | | |
| | CT Regime [quant] | Interfaces [qual/quant] | | | |
| Thermal-based | Conduction path [quant] | Thermal cycle [quant] | Material insulation [quant] | Policy and support [qual/quant] | Passive systems [qual/quant] |
| | Coating [qual/quant] | User comfort [qual/quant] | | | |
| | Energy gains [quant] | | Thermal inertia [quant] | | |
| Power-based | Solar array surface [quant] | Actuators [qual/quant] | Energy sources [qual/quant] | Outreach [qual/quant] | |
| | | Power Modes [qual/quant] | | | |
| Mechatronics | Packaging [qual/quant] | Functions [qual/quant] | | Local vendors [qual/quant] | Types of motors [qual/quant] |
| Physics-based | Structure strength [qual/quant] | Tribology [quant] | | | |
| Chemistry-based | Exposed area [qual/quant] | Decontamination [quant] | Corrosion [qual/quant] | | Material catalog [qual/quant] |
| Electronics-based | | Modes [qual/quant] | Waterproof [quant] | | Components [qual/quant] |
| | | Programming [qual/quant] | | | |
| Data-based | | Data flow [qual/quant] | Data architecture [qual/quant] | Tool compatibility [qual/quant] | |
| | | Coding [quant] | | Open source [quant] | |
| Implementation-based | Manufacturing | Integration [qual/quant] | Post processing [qual/quant] | | Proven technique [qual/quant] |
| | | Reparability [qual/quant] | | | |
| Operations | Assembly size [qual/quant] | Autonomy [qual/quant] | Recyclability [qual/quant] | Governance [qual/quant] | |
| User-based | Simplicity [qual/quant] | Easiness [qual/quant] | Aspect [qual/quant] | | |
| Workforce | | Disciplines [qual/quant] | | Diversity [qual/quant] | |
| Optimization | Mass reduction [qual/quant] | Function reduction [qual/quant] | | | Previous studies [quant] |

*Table 28. Some system-level types of requirements organized by discipline for a generic hardware-based system architecture.*

#### 5.8.5.3. Figures of Merit as Questioning Drivers

All previous inputs up to this point are studied to provide an initial set of figures of merit. These are critical as overarching metrics to evaluate the progress and success of a future system architecture design. While they might change over time, they are foundational to start the inquiry process (eADQN) since they provide guidance about what the starting point should be. In essence, these are questioning drivers and a very powerful tool to address where the most important gaps (eAMGs) are found. For instance, the total mass of the system, the energy required for the system to work, its final volume, or the style ensuring acceptance can be examples of these figures of merit. They are both quantifiable and qualifiable figures that represent initial objectives for the design process, and they are obtained through an assessment of both requirements and expectations (DOI). After each design cycle, the system architecture (ARR) gets more detailed, and its performance is closer to achieve those figures of merit or even to surpass them. However, the evolutive design process (eSARD) concentrates on identified eAMGs which also provide internal connections among tools, milestones, subsystems, and figures of merit (unknown or otherwise). Thus, these are also temporary like anything else on an evolutive process.

These figures of merit also serve as a quick quality control tool for the design outcome and its path. If the design strategy being explored does not provide solutions closer to these drivers, then changes need to be made. This allows to make decisions even when the full system design is not finished, providing a great research opportunity for data-driven methods to both optimize solutions as well as to predict future issues. Within the eSARD method the objective is to fail fast.

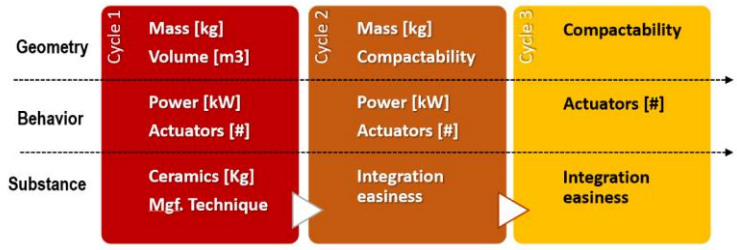



*Figure 165. Example of evolution of figures of merit over a design process.*

These figures of merit should map three key GBS pillars describing any complex system: **geometry, behavior, and substance**. The first is related to all drivers that are applied to the descriptive principles behind a system architecture (section 4.3). For instance, the volume of the final configuration refers to system compatibility, thermal performance, and mechanism complexity. Behavioral figures of merit are related to functional aspect of the system architecture, including data-rates, functional power modes, autonomy topics, etc. Finally, substance figures of merit are about materials, manufacturing variables, coding, and any resource used to implement the system. The initial selection of these often depends on the nature of the system, the design experience of the team, the capability of the workflow, and the cultural heritage of an organization. Chapter 6 presents a detailed study case, and Figure 165 shows some initial figures of merit and their evolution in time regarding the previous captured requirements (Table 28).

#### 5.8.5.4. Constraints

Requirements and expectations are also limited by design constraints and stressors (chapter 2, section 4.1). These barriers can be provided by the customer or developed during the evolutive process. They are also critical since they limit the design effort, and often help enforcing some customer expectations as well. After subsequent design cycles, some of these questions might also bring awareness of other constraints which might not have been initially identified.

#### 5.8.5.5. System Gaps and Questioning Networks

Developing questioning networks is at the core of this initial phase in the eSARD process. However, here are considerations, guidelines, and process descriptions that must be considered. The overall objective here is to identify gaps in the system architecture that are necessary for the system to close, so it becomes implementable, fully functional, and thus feasible. However, the final goal again is not to create a full system description but to identify major rifts in the completion of the system. Components and subsystems among others are also part of the process since they can have feasible known solutions (indirect heritage) that do not contribute the other system gaps.

The development of foundational inquiries is the first step in the eADQNs process. These are mayor questions based on design objectives, requirements, and identified drivers, which are meant to start the questioning process around all three GBS basic areas. Then, three initial lists are started with a series of questions addressing those major gaps such as:

- **Discipline requirements gaps** include which requirements are met from a discrete discipline standpoint and how that is done, for instance: what is the energy generation system? What kind of structural light weighting is needed?
- **Systems gaps** address unsolved (design) or present conflicts (balance), for instance: are there any other unknown gaps? Do they represent important heritage conflicts? How deep have SE and DE efforts been advanced?
- **Future gaps.** Under all evolutive design principles the system architecture is studied from a continuous standpoint, even if the objective is just to create a one-off solution. The more questions are addressing possible upgrades or updates (even if they are not required), the better will be the exploration of the design space.
- **Disruptive gaps.** Similarly to C-K theory (Hatchuel and Weil, 2002) it is critical in this process that 'crazy' or disruptive possibilities can be possible. This is done by allowing 'what-if' inquiries that could potentially 'kill the gap by design' while following evolutive principles (section 5.5). The more gaps a design approach can tackle with less components the more efficient it becomes, but such disruptions can only be addressed after diving into all design limitations.

Geometry, behavior, and substance subjects drive the development of these series of questions addressing systems gaps. This approach explores from high level issues to all design details that cannot be foreseen beforehand. Thus, such approach must consider the following overarching scales in terms of gaps:

- **Highest level gap at any scale,** from systems-level topics affecting full architecture to small scale details.
- **Highest level of required innovation** to answer design, technology, or process gaps.
- **Highest lack of heritage** affecting subsystems, components, structure principles, and methods.

These questions can be developed by a design facilitator in a group (chief architect), a single designer on its own, or by a SE effort using multiple techniques (including automated models). The next section will describe this in detail, but general engagement rules for such process are summarized in the following points:

- There are **no limits** for the number of questions that can be made.
- All **GBS** details should be considered to address all DOI aspects within the system design.
- Each inquiry needs to be identified, captured, and organized which is critical to create and explore connections.
- **Complementary techniques** such as TRIZ can be used to formulate inquiries in any eSARD process.
- Recording these structures allows to **repurpose and reuse** them for future upgrades or upcoming efforts.
- Recording decisions and selections criteria allows **tuning future algorithms, training** workforce, identify errors, etc.
- The process is **agnostic** of any specific toolset to create, capture, and operate eADQNs methods and processes.
- Gaps can be represented by **multiple sources**: alphanumeric, analytics, images, sketches, Eng. models, etc.
- eADQNs can also be used as **decision trees** capturing the rationale behind a system design path.

The outcome of this process is a structure set of dynamic questions (eADQN) as Figure 166 shows graphically. These questioning networks will present different types of structures depending on the nature of the problem, but this is out of the scope of this thesis. This approach was initially designed to be used by low-tech and human-powered workflows (with only one or more people involved), as well as to serve as a foundation towards more automated infrastructures such as those powered by machine learning frameworks (Herbst and Karagiannis, 2000; Tamke et al., 2018; Xin et al., 2018) and hybrid human-machine workflows (Demartini, 2015).

These structures in general also present interesting training opportunities as well. While these can be captured using MBSE tools and other SE techniques, a series of graphical representation principles are provided with the goal of facilitating the communication within a 3C evolutive environment, as well as to enhance the study and review of the process at any given time before, during, or after the activity. They also help identifying commonalities among studies to model automated process later. Figure 166 shows several basic representation rules to develop a three-dimensional descriptive maturation space for an eADQN process that include these:

- Vertical direction (Z) relates to the depth of the detail. The lower the questions the more detailed the inquiry becomes.
- Each horizontal plane (XY) for every vertical level (Zi) relates to the proximity among concepts and ideas.
- Size scales the importance of the question, so the bigger it is the more important it is.
- Links between questions could be done with lines, colors, and symbols, among others while allowing overlaps.
- Colors can be used to reflect relationship and types as subsystems, disciplines, DOI aspects, GBS topics, heritage connections, etc.

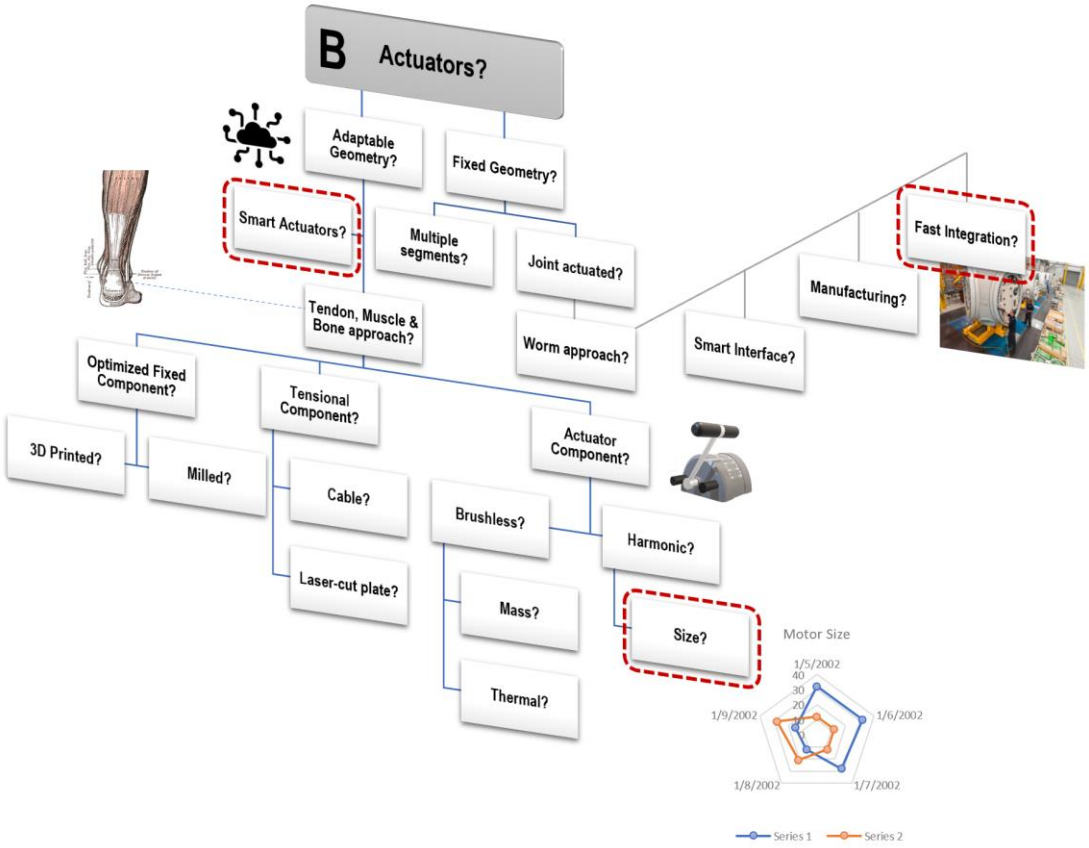



*Figure 166. Example of eADQN created for the discussion regarding a generic electromechanical actuator that includes behavior, geometry, and substance topics.*

### *5.8.5.6. eADQN Generation*

Therefore, the most critical aspect of the eADQN technique is how to efficiently make the right questions to identify those critical gaps. Previous sections have explained the process, the context, the rational, main objectives, and even the representation of those questing networks. But complex systems in general, and evolutive system in particular (ARR) are characterized by several relationships among GBS points. As such, an eADQN model is not just a list of questions, is also a network of gaps and relationships. To start creating this network, which will continue evolving and growing over time, the eSARD methodology presents a double-level information approach. This means to assess overarching topics based on previous gaps, creating links among them, and evaluating their relevance all of which will be constantly changing.

The first level of information (or system incompleteness in this case) is about addressing the most critical questions (nodes) within that network. This means they are tackling key overarching DOI areas as it follows:

- **Missing or undefined elements** such as: components, elements, parts, etc.
- **Critical disciplinary knowledge**. The feasibility of a system design depends on multiple and concurrent disciplinary studies and assessments: which ones are done? Can they be done? Which ones do seem the most critical a priori?
- **Applied or needed technologies** including those critical, new, untested, problematic, or presenting key constraints.
- **General DOI topics** such as system integration easiness, relative cost, workforce needs, etc.

The second level of information is about connections among those key question nodes, which follow subsequent **GBS details** for their implementation. In essence, this level questions the most critical gaps and relationships among them. Thus, it is about the scope, relevance, and criticality of the connection too. Such criterion is based on these basic principles:

- **Synergy**. Is there a technical, programmatic, or technology connection across inquiries?
- **Complement**. Do such inquiries point out to other complementary topics?
- **Dependent**. Are there any inquiries presenting dependent or critical relationships?
- **Causality**. Does the inquiry relate to gaps with a cause-effect relationship?
- **Areas**. Does the link connect gaps among multiple GBS areas?
- **Levels**. Does the gap tackle multiple system levels across the system architecture design?
- **Epistemology** (optional). Is the question related to probabilistic, counterfactual, or mechanistic links?

Beyond these selection and exploration principles, there are other overarching principles driven by the environment, the culture, and the technical context of the system, which also help identifying and assessing these links such as:

- **Relative Cost.** What is the balance between opposed forces? What is the monetary cost?
- **Availability.** Is this connection conditioned by the ecosystem of the system? Has it been done before?
- **Applicability.** Are different links equally applicable? Does the context or environment condition the link?
- **Feasibility.** What is the level of feasibility for this connection? Has this been proven before in the process?

However, these principles are not the only ones. Just like other links and nodes, these will keep evolving even within the same design effort, so these criteria should be adapted. The goal is to dig enough into the details of the concept, that the most critical gaps (eAMGs) can be found.

This questioning network mimics some basics points in the natural evolution process (section 3.3) affecting the interconnectivity within the complex system, such as: [1] the importance of small but critical changes in the overall system, [2] the balance among species overarching principles. The more critical and fundamental these are, the greater impact they can have in terms of efficiency and efficacy for systems and processes.

Therefore, addressing such gaps will have a ripple effect throughout the system architecture since their impact ratio is much higher. Such gaps are often hidden and not obvious because they are not only driven by initial system requirements, but also such deepest idiosyncrasies of the solution system. This is like sequencing a specific DNA code, if it can be decoded then its essence is understood, and therefore the organism can potentially be fully studied, compared, replicated, and even modified. Although, the relationship among such genes and its context requires interaction and more study.

Since those quintessential links are connecting gaps across sub-systems (e.g., batteries) and disciplines (e.g., thermal engineering), they also present a unique opportunity to infuse and evaluate innovative solutions. This is a powerful mechanism that can help tackling both extremes of the design spectrum when dealing with [1] new complex systems without any heritage, as well as [2] systems requiring upgrades with a very strong and rigid heritage. There are two cases then.

- **New systems (no direct heritage)**. If the new system architecture is very innovative and does not have heritage, then all critical gaps most likely will be present a higher level. They will have less detail and the infusion of concepts or technologies will characterize the effort. For instance, this can be seen during the development of the television (Abramson, 2007), when a mechanical approach was first taken due to the lack of heritage. Its failure lead later to a completely new type of architecture using analog and then digital electronics instead. How pixels were going to be captured, transmitted, and implemented was the core idiosyncrasy of such system architecture, requiring higher levels of adaptability and reactivity. Designing for the unknown is complicated, but also quite feasible if multiple options are kept open for as long as possible, its context addressed, and enough connections between system gaps, requirements, and context explored, captured, and studied in detail.
- **Upgrades (strong direct heritage).** On the other hand, if the system at hand is understood as a variation or improvement of a system architecture that already presents a solid and proven heritage, then such connections will present a lower level of detail, and the infusion and disruption might have less consequences. In this case, indirect heritage coming from other fields can be infused more easily. Following a similar example, an upgrade in such system could be a new TV architecture that is data-driven (Wi-Fi), so it is a cloud-based solution that only needs an internet connection rather than an analog antenna served by physical radio stations. Thus, all operations are done with an app, and there is no need for buttons or other physical interfaces. Such design tackles both subsystems as well as critical business and programmatic topics. However, these innovations at a subsystem level can also have huge effects in the overall performance of the system architecture, even if its fundamental nature is not changed by them.

## 5.9. Maturation Gaps (eAMGs)

Once the questioning network has been started addressing both system gaps and links, the final step is to rank them, since these are the seed for all system maturity gaps that become the starting point of an eSARD. Evolutive architecture maturity gaps (eAMGs) are the most critical inconsistencies jeopardizing the feasibility of the system architecture. Thus, the objective of this method is to identify and provide criteria towards where the design process should focus to achieve system closeness more efficiently and with the right maturity level. All this is driven by feasibility, system performance, and system innovation especially with regards to all ARR principles behind the evolutive approach. eAMGs help organizing design efforts while providing the biggest impact across the system. So, often this assessment can only be done by diving into the design effort and challenging the resilience of the concept through multidisciplinary questioning.

Therefore, the same ARR principles used to find these gaps are considered to rank them. However, other overarching criteria to compare them and most importantly to put them in context. In general, the more points, requirements, and disciplines these gaps can address the better starting point they will become. Furthermore, the broader and more detailed these connections are, the more efficient and easier to provide disruptive solution the design process will be. These principles can be quantified with a matrix to assess and compare them using multiple SE techniques, such as a variation of the house of quality or HOQ (Figure 59). In essence the more synergetic and foundational these gaps are the better starting point they will be as well. Chapter 6 will present a full example applying all these principles towards the development of a deployment subsystem for a portable habitat.

A summary of this criteria to assess their synergetic potential is presented in the following points and showcased in Table 29 presenting both identification and ranking principles. The more influence a gap has on these points the higher it will be ranked, and the more critical that gap becomes. Among other assessment points some of the most basic are these:

- **Requirements**. This reflects the criticality of the gap towards a system meeting its requirements. The objective of the design process conditions both the answer and the process behind it.
- **Constraints**. This is about the influence of the gap to both reduce and manage key system constraints.
- **Disciplines**. The more disciplines a gap addresses, the more is going to influence the system architecture, which means more synergy, complementation, dependency, and causality among standpoints.
- **GBS figures of merit.** It is very important to assess the importance of a gap regarding how many GBS areas (geometry, behavior, and substance) it tackles directly (section 5.3, 5.4, 5.5), as well as what key variables and parameters it affects as well. This influences many areas and applicability principles within the list presented in the previous section. This is critical due to the interconnection among design principles and system solutions.
- **Heritage.** The lack of heritage could contribute to the importance of the gap since this can condition if there is a baseline to compare and validate the system design.
- **Closeness**. The relevance of the gap towards the completion and closeness of the system is also critical. This tackles account causality, detail levels, and other epistemology principles across all scales and system levels.
- **Programmatic**. This point addresses how non-technical aspects are weighted by this gap, such as cost, organizational, policies, culture, customer feedback, perception, communication, and marketing, among many more.
- **Unknowns.** Does the gap relate or connect to other potential families of solutions that are very complicated to assess as feasible, applicable, or available systems? Does the gap highlight too many potential unknows?

Among all other identified gaps, **evolutive architecture maturity gaps** (eAMGs) are those being ranked higher (Table 29), meaning they are more critical and thus foundational for the system architecture at hand. These gaps do not allow for the system design to close, presenting essential disconnections among subsystems or within manufacturing uncertainties to name a few. However, at both subsystem and architecture levels these gaps are multidisciplinary in nature and most likely will continue changing during the development process. Once those synergetic connections are explored, evaluated, and ranked a selection of the most relevant gaps is done. These are multidisciplinary in nature, and their development could speed up the design process significantly. Therefore, these become eAMGs and they represent the most critical gaps in the feasibility and maturity development of a system architecture. These are also very relevant towards the system implementation and optimization. eAMGs are the starting point of the design path to mature the system. Some theories and methodologies previously explored in chapter 3 pay attention to the design activity itself (Gero and

Kannengiesser, 2004), as well as all underlining principles behind complex systems (Altshuller, 2002). However, within the eSARD approach the goal is also to address both from a synergetic standpoint, with the clear objective of producing a feasible system architecture in the most efficient, fast, upgradable, and scalable way possible. Therefore, this approach is about holistic views across levels, disciplines, and scales.

However, the nature of the system at hand as well as its cultural and technical context heavily influence these questions and connections. This approach relays on being open and flexible to better explore the maturation space while assessing missing gaps and topics. By centering the effort on those critical connections, new solutions can be infused, since the bias towards the heritage option is removed at the level of the system. Such process is dynamic and allows the designer (person) or the workflow (machine) to concentrate the effort on what 'breaks' or enables the system design. In other words, potentially answering those critical questions with specific but creative solutions closes the system, matures the concept, and enables its implementation all together.

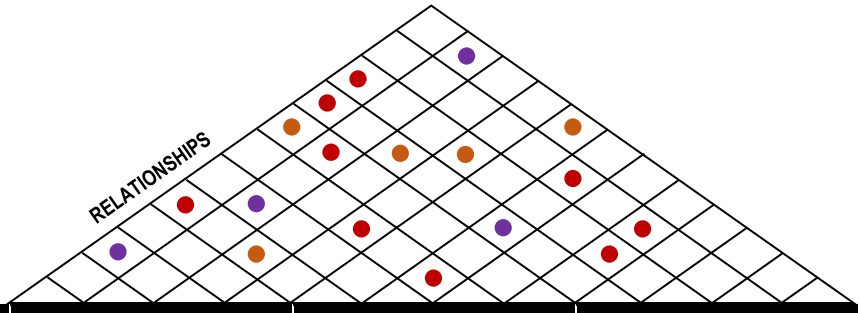
The approach of this research is based on research, observation, proven practice, and some of the state-of-the-art design methodologies as a foundation. Nevertheless, it is specifically oriented towards evolutive hardware-based disruptive systems architectures with ARR characteristics, such as system adaptability, reactivity, and resources regeneration.

It is important to remember that an evolutive architecture design (eSARD) methodology can be used to finish a system as well as to mature a concept. This approach means that the design effort could be oriented not towards addressing every detail of the system architecture, but to develop a strategy or design path only in response to those eAMGs. The alternative is trying to identify all identified gaps that could be needed to address requirements, which often cannot be realistic.

For instance, among the many requirements for a lightweight bike chassis, structure joints and the manufacturing technique can drive the feasibility of the architecture. As previously stated, chapter 6 will present a more detailed study case, however following the previous example of a generic electromechanical device Table 29 shows an analysis of key gaps used to identify eAMGs. On Table 29 the following areas are being addressed:

- **DOI Gaps** (top). They are organized by GBS principles. The goal towards identifying eAMGs is to explore and select gaps based on GBS details while considering all DOI general aspects of a system architecture. Understanding these areas helps a chief architect or facilitator to explore needs and options, going beyond requirements and suggestions provided by customer or former practices. The broader and deeper the search can be, the better. A selection of these is found at the top part of Table 29 and organized by their tendency towards the GBS area.
- **Selection and refinement criteria for gaps** (mid-top left). This presents a summary of different criteria used to assess the importance of such gaps towards closing the design and relationship among them. Specific selection parameters are later tuned by other refinement aspects (e.g., relative cost) that consider overarching solutions and relationships between them. These results are a first high-level assessment of the importance of each gap.
- **Ranking parameters** (left). These are applied to specifically assess the relative weight of all gaps into the system architecture design. Among some techniques the use of weighted numbers is recommended to assess criticality (e.g., 1 for lowest and 10 for highest). Once more, all parameters used here can be changed based on the system design.
- **Weighted importance** (bottom). The result of ranking these parameters is a number by which gaps are compared.
- **Relative importance** (bottom). Beyond such number, selection and raking criteria leave room to finally assess from a chief architect's perspective or a workflow process what the starting point for the process should be.
- **System Importance** (left). This provides a nuance to weight the selection parameters based on different criteria.
- **eAMGs** (top, red). The final selection of eAMGs on a specific design round are highlighted in red.
- **Legend**. This is applicable to Table 29 as it follows:
  - **Selection:** red (high), orange (medium), purple (low)
  - **Ranking:** 1 for low criticality and 10 for high criticality

Nevertheless, none of these parameters and numbers are absolute. Getting deeper into the development process will help identify other eAMGs, as well as to change the relevance of certain gaps that can be addressed and surpassed by new design strategies. The more this exploration and maturation process is developed, the more critical eAMGs will be found. Therefore, this not only becomes a tool to design, redesign, and upgrade a system architecture, but it is also a tool to validate heritage solutions within new contexts or applications.




| | | | GEOMETRY | | | | BEHAVIOR | | | | SUBSTANCE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *High Med Low* | **DOI GAPS** | **System importance** | **Packaging factor** | Mass optimization | Dust proof interface | Sensor occlusion | System reshaping | Automated triggers | Dust operations | **Autonomy integration** | Metal alloy | **Power replenishment** | Programing options | Corrosion in opening |
| **Selection** | **Synergistic** | | ● | ● | ● | | | ● | | ● | | ● | | |
| | **Complementary** | | ● | ● | ● | | ● | ● | ● | ● | ● | ● | ● | ● |
| | **Dependencies** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | |
| | **Direct causality** | | ● | | ● | ● | | ● | ● | ● | ● | ● | | |
| | **Multiple areas** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | **Many levels** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| ***Refined*** | ***Relative Cost*** | ● | ● | ● | ● | ● | | ● | ● | ● | ● | ● | ● | |
| | ***Availability*** | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | ***Applicability*** | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | ***Feasibility*** | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | *Partial Total* | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Raking** | **BSG topics** | | 9 | 1 | 1 | 3 | 2 | 3 | 1 | 8 | 2 | 9 | 1 | 2 |
| *1 Low \| 10 High* | **Requirements** | | 4 | 4 | 5 | 1 | 7 | 1 | 8 | 7 | 1 | 9 | 2 | 1 |
| | **Constraints** | | 3 | 10 | 4 | 2 | 3 | 8 | 2 | 8 | 2 | 5 | 8 | 3 |
| | **Disciplines** | | 9 | 2 | 2 | 2 | 1 | 2 | 10 | 10 | 2 | 9 | 3 | 1 |
| | **Heritage** | | 2 | 1 | 1 | 4 | 2 | 1 | 1 | 1 | 3 | 2 | 2 | 4 |
| | **Closeness** | | 9 | 3 | 3 | 5 | 9 | 3 | 1 | 4 | 1 | 10 | 1 | 3 |
| | **Programmatic** | | 2 | 5 | 1 | 2 | 2 | 1 | 2 | 9 | 3 | 4 | 2 | 4 |
| | **Unknowns** | | 5 | 2 | 3 | 4 | 1 | 2 | 3 | 7 | 8 | 2 | 1 | 1 |
| | *Weighted Importance* | | **43** | 28 | 20 | 23 | 27 | 21 | 28 | **54** | 22 | **43** | 28 | 20 |
| | *Relative Importance* | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |

*Table 29. Identification, comparison, and ranking of system design gaps and eAMGs within and eSARD approach.*

## 5.10. Design Strategy and Seed Geometries (eASGs)

The identification of eAMGs as the starting point of the evolutive design process leads to the development of a design strategy that tackles those gaps, while it keeps considering other operations, optimization, and implementation aspects. Such design strategies address primary and secondary requirements, as well as key synergies among eAMGs that can be combined and dismissed over time. Such process is done using a networked flow within a 3C design environment (concurrent, collaborative, and communicative) that was presented in section 5.6.6.

Therefore, the next step within an eSARD design activity is to create a seed concept that addresses those initial eAMGs. Such concept will act as a seed to [2] explore multiple design paths synergistically and addressing all DOI related topics, and [2] to lead subsequent design cycles to further mature the system and include other aspects such as the use of resources. Like any other step within an eSARD approach, these are fluid and will change over time following the evolution of the process and the system. This seed concept development often has two parts or tools:

- **Evolutive sketches** addressing overall concepts and relationships as a visual starting point for the design.
- **Evolutive geometries** provide detail definition at each design cycle while serve as technical documentation too.

The following sections explain in detail what these tools are based upon, what they can be, how they can be used within an evolutive approach, and how they can be complemented by other system design engineering methodologies.

### 5.10.1. Evolutive Seed Sketch (eSSs)

An evolutive sketch (Figure 168) is a visual concept definition that helps focusing design efforts while addressing one or more eAMGs as the starting point of the evolutive process. This graphic and conceptual design tool is fast, facilitated, multidisciplinary, it changes continuously, and it uses multiple types of information and graphic techniques. Beyond a pure concept sketch approach, an evolutive sketch serves these multiple key objectives:

- **Geometry**. Seed evolutive sketches create a geometrical and reference construct, used as a critical common ground that tackles GBS details of a system architecture. Geometry here does not mean only shape, volume, or spatial relationships, but also logical and temporal aspects. Thus, this objective is critical for multiple disciplines to start thinking about the design approach. For instance, until there is a geometry sketch, mechanical, thermal, and packaging related disciplines cannot start assessing solutions. This is crucial when dealing with hardware-based system architectures. Hence, evolutive sketches facilitate, create, and show information regarding:
  - **Volumes and shapes** that are presented using and hand drawing, diagrams, pictures, etc.
  - **Relationships** of any kind addressing GBS details and DOI topics (e.g., block diagrams).
  - **Configurations** using a very descriptive approach through symbols, colors, diagrams, etc. (Figure 167)
  - **Evolutionary states or phases** that are snapshots of the system evolution in time.
- **Disciplinary synergies**. These sketches develop design paths that start with eAMGs, but they are also used to tackle and study those gaps. Along that approach, one of its objectives is to create a framework for all disciplines and practices involved in such development, so they start sharing information and interacting among them and to provide a seed that serves a starting reference. Such framework includes among others: design approaches, structural parameters, logical schemes, space-time references, and procedural flows, among many more.
- **GBS initial details**. In addition to synergies and geometrical aspects, these evolutive sketches need to address details of the maturation and implementation of the system. Providing a reference and visual a representation of these details (e.g., location of sensor, materials, etc.) enable a fast track towards filling or detailing other gaps.
- **Associated DOI processes.** Similarly, the sooner topics such as manufacturing or resource utilization are addressed within a sketch, the stronger the design path will be. These sketches present a balance between specific GBS details and overarching DOI principles in all key questions since both scales are needed to gradually mature the system.
- **Figures of merits.** Combining all these points withing eAMGs points out to a series of figures of merit that become the most definitory parameters of the system and should be tackled by the sketch. An evolutive sketch will change quickly and those figures will do as well. For instance, these could be the total mass of the system, the final volume of a compacted assembly, and maximum power require by a system architecture, to name a few.

*Figure 167. Examples of concept definition elements used within a fast evolutive sketch (sketch by Raul Polit Casillas, 2009).*

Nevertheless, such sketch is not necessarily handmade, and it can very well be a conceptual computer model capable of addressing, capturing, and connecting key points within a design path. Any technique to create such sketch must include the integration of multiple types of information (Figure 167) that can be summarized with these points:

- **Media.** Sketching the concept should use a media that is easy to share, communicate, and even collaborate upon. This media can be physical (e.g., paper), digital (e.g., whiteboard app), and even virtual (e.g., collaborative VR). Digital media offers an advantage since it allows easier modifications and updates during and after the design effort. This activity will continue for as long as the design needs more and better definitions, so it will be constantly addressing changes while details are being redefined from a disciplinary, systems, and DOI perspectives.
- **Facilitation.** An evolutive sketching activity is or can be collaborative in nature, but it also requires facilitation to prioritize efforts and select the best design paths, while it addresses feasible solutions for initial and subsequent eAMGs. This facilitation can be done by a chief architect or a workflow, which could be less efficient. Its objective is to enable multiple perspectives (e.g., team members, discipline models, etc.) that focus their attention into the design strategy, so they can all assess concurrently the validity of their approach from their different perspectives.
- **Workflow.** Evolutive sketching starts a collaborative and design-focused activity. Such connected activity leads to independent assessments and microstudies that are concurrently done with a disciplinary standpoint. For instance, after an initial sketch that presents the overall geometry of a device (Figure 168), the thermal team assesses the validity of such initial geometry, while manufacturing constraints can be also studied by mechanical representatives.
- **Information.** All types of information and data are valid at this initial phase, see Figure 167.Hola
- **Constant revisions.** Each sketch can present multiple modifications during the design effort.

In summary, an evolutive sketch is a powerful tool, especially for facilitated groups or workflows that allow to quickly refocus objectives while having all disciplines addressing key eAMGs concurrently and efficiently. Bringing both geometry and logical structures together (especially for hardware based ARR architectures) within a fast and highly adaptable framework, sparks the creative thinking while it helps increasing efficiency by providing a common reference.

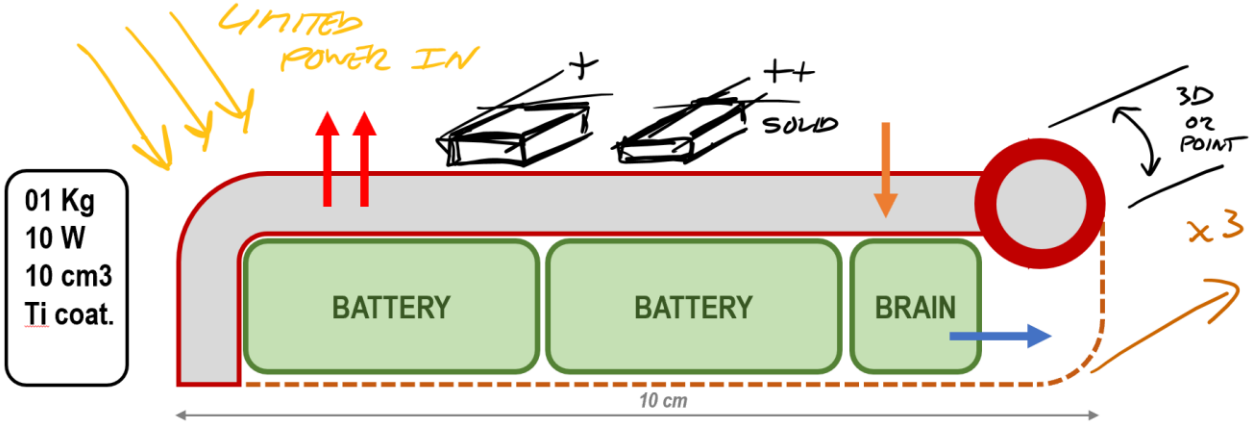



*Figure 168. Example of an evolutive sketch used in the design of a fictional and generic small electronic device.*

### 5.10.2. Seed Evolutive Architectures (eASGs)

The objective of this design methodology is to enable the maturation of the system while addressing its feasibility, thus such effort tends towards a more detailed geometrical construct or evolutive architecture seed geometry (eASG). This model incorporates evolutive sketches among other inputs. In essence, it is a more detailed design seed that addresses GBS details and DOI areas conceptually, while it provides detailed geographical definition through measurements, interfaces, computer models, etc. This is not the final graphical and logical documentation that fully describes the system because since eASGs still represent and undergoing effort and will keep changing until the full maturation of the design. This tool though is very relevant considering that system adaptability is a foundational aspect of all evolutive system architectures and subsequent development processes. This multidisciplinary trend keeps impacting multiple areas of the design engineering discipline when data-driven and multidisciplinary design techniques coexist (Cavas-Martínez et al., 2020). Some of the biggest differences with more traditional geometrical descriptions are summarized here:

- **Adaptability**. These eASGs are created to keep changing. In essence these graphics are to traditional sketches to what animated cartoons are to a painting. They are dynamic (manual or not) so the use of layers, drag-and-drop models, stickers, and other easy to modify techniques is highly recommended.
- **Multisource**. The combination of technical geometry, diagrams, figures of merit (alphanumerical data) as well as colors, images, 3D models, and symbols is at the core of this bidirectional graphic approach tackling eAMGs.

A key goal of these eASGs is to continue the development of the system architecture, while helping to manage other design efforts supporting eAMGs. Critical objectives of this toils are summarized on these points:

- **Maturity enhancement**. eASGs are used to manage, facilitate, and foster increments in the design maturity of a system architecture by addressing more connections among other disciplines and DOI areas.
- **More definition**. They systemically develop and integrate more details (GBS) on each design cycle.
- **eAMGs design strategy enhancement**. eASGs should be used to taunt current eAMGs and find new ones. It is key for facilitators or workflows to assess and try new connections and links beyond the current state at every cycle.

These evolutive tools display, manage, and use the following type of elements in a concurrent and collaborative way:

- **Detailed geometries** using multiple views and graphical schemes including hand drawing, CAD, algorithms, etc.
- **Descriptive logical information** such as flow-charts, data-visualizations, visual scripts, programming flows, etc.
- **Figures of merit** based on alphanumeric data (both qualifiable and quantifiable) that address key parameters.

Furthermore, at this level of sketch and geometry detail, eASGs workflows present the following characteristics:

- **Tool agnostic**. This approach is independent from any specific toolset, and it is based upon their capabilities.
- **Constant change** over time (e.g., mechanical dynamics, changing states, upgrades, etc.). Even if the objective of the design effort is static (e.g., a chair), the nature of a geometry seed and its process is evolutive in nature, so materials, manufacturing considerations, indirect behaviors, or future potential updates, among other aspects are part of its design thinking approach and dynamic in nature.

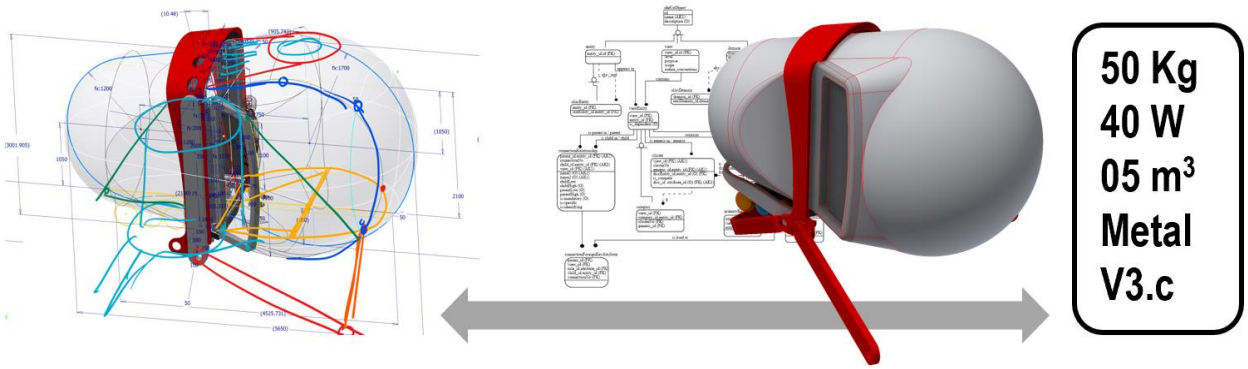



*Figure 169. Example of an evolutive architecture seed geometry (eASGs) for a personal habitat (© 2020 Raul Polit Casillas).*

## 5.11. Evolutive Seed Models (eASMs)

Previous tools have addressed geometry, behavior, and substance topics related to multiple maturity gaps in an evolutive system architecture (eAMGs) development. Evolutive geometries continue the design process mainly through a facilitated geometrical and systems perspective that considers other procedural, programmatic, and logical aspects at a very high level. These previous tools are concentrated on the design edge (design sector, Figure 144), addressing DE aspects from such enhanced perspective. Hence, these provided an initial geometry (e.g., shape and logical elements) for the system that is critical for any hardware-based effort. However, complex systems in general, and evolutive system architectures in particular also require of other logical, parametrical, and relational aspects to be also considered, managed, explored, and improved to achieve enough system maturity. This process enables feasible and more holistic solutions.

Therefore, this is the starting point for evolutive seed models or eASMs. These are a SE-oriented toolset developed as a modification of several traditional SE models to address ARR evolutive needs. These can tackle both GBS details and DOI general areas, which are especially relevant here. eASMs present the following key objectives:

- **Non-geometrical relationships**. eASMs capture and manage all types of logical relationships between systems, process, components, and other critical aspects of an evolutive system architecture. This includes multiple GBS details that are relevant to the definition and development of the system such as:
  - **Geometry enhancements** that increase or complement any level of information that is used in geometrical frameworks using CAD and BIM (e.g., interfaces, number of components, integration phases, vendor, etc.).
  - **Behavioral functions** affect the system internally, externally, or both. These include among others machine-state topics, hierarchies, data-architecture flows, and other functional descriptive models.
  - **Substance variables and feedbacks** are related to all DOI aspects. These could be simple (e.g., power, mass, etc.) or complex presenting secondary relationships (e.g., FEA derived measurements, data feedbacks, etc.)
- **Quantifiable parameters and qualifiable enhancements**. Like other SE tools, capturing parameters that can be quantified (e.g., mass, dimensions, power) is critical not only to evaluate the system but also to compare among alternatives, address changes, etc. eASMs tame the process by addressing other non-quantifiable relationships that provide nuances and key perspectives among such connections and parameters.
- **Overall DOI perspectives**. Beyond the development of a system from a life cycle standpoint, the evolutive approach also tackles design, operations, and implementations aspects simultaneously. Thus, eASMs should be built first around all identified and most critical eAMGs.

eASMs are in essence a family of tools addressing key SE needs in an evolutive development. As part of the design vector, these can and must be connected and paired with eASGs. DE and SE practices are merged within the evolutive approach, thus these evolutive system models present some key characteristics such as:

- **Feedback loop / open requirements**. These need to connect with other evolutive tools allowing multiple iterative cycles to improve designs. So, they are designed to capture a constant flow and not to just provide a final answer.
- **Figures of merit**. Capturing, highlighting, and increasing the numbers of these key parameters is critical.
- **Optimization paths**. These models are a base for further data-driven optimization activities. They should be open towards refining, creating, and analyzing new connections among them under a synergic optimization principle.
- **Linking eASGs and eAMGs**. The implementation of links and secondary parameters as doors or ports towards connecting these tools throughout multiple cycles is vital. These are not individual tools but nodes within a network.
- **Evolutive**. These models are multidisciplinary in nature, attending all GBS, DOI and ARR topics.
- **Man-machine.** All associated workflows to these models should be both human and machine friendly.

This toolset of eASMs is based on standard models such as mass equipment list (MEL, Chung et al., 2012), power equipment lists (PEL, Ochoa et al., 2009; Pasquier et al., 2019), WBS, SEMP, FFBD, among others (Table 15). There are two big families of tools within the ecosystem of eASMs: [1] GBS equipment list (eGBSEL), and [2] evolutive system diagrams (eSD). Next sections elaborate a bit more in detail these tools, however the full development of SE tools supporting optimization and implementation activities is outside the scope of this research theory and its results.

### 5.11.1. Evolutive System Diagram (eSD)

Within the eSARD approach (section 5.6) and based on the design activity in all DOI sectors, the development of 2D or 3D evolutive system diagrams (eSD) showcases multiple design subjects in the workflow, such as:

- **Systems relationships** among system, subsystems, components, inputs, processes, and connections.
- **Procedural flows** (e.g., integration, testing, etc.) addressing resource utilization and implementation steps.
- **Operational and geometrical modes** based upon the adaptability of the system.
- **Functional and information schemes** addressing the reactivity of the system.
- Any other relationship and flow among **DOI areas** and across system scales and levels.

The use of eSD diagrams for these purposes requires several rules and characteristics summarized as it follows:

- **Expandability**. Because the evolutive approach is based on the notion of continuous design, any diagram scheme needs to enable such continuity both functionally and dimensionally. Similarly, to how eADQNs are captured on an endless 3D environment, these eSDs need to be implemented with an expansion and continuation mindset.
- **Multisource**. eSDs should be able to integrate multiple sources of information as a complement to the design workflow they show graphically, logically, and systematically.
- **Multiuser**. Both framework and workflow established to create such diagrams must embrace all 3C principles, allowing multiple agents to collaborate, as well as to easily capture and restart the process at any given time.
- **Machine friendly**. Evolutive techniques are not only for human-centered workflows since they can be enhanced by data-driven techniques such as Machine learning, AI, genetic programming (section 3.3, Table 17), etc.

### 5.11.2. GBS Equipment List (eGBSEL)

Among some of the tools used within the eSARD approach, the GBS equipment list or eGBSEL summarizes from a foundational standpoint the implementation of evolutive architecture seed models (eASMs) very well. While the goal of this research is to address the gap that evolutive architectures address through the eSARD path, it is important to showcase some key tools complementing this research. These models reinforce the validity of the methodology and empathize its application. However, eGBSEL is not fully developed in this section since only its basics are presented.

From an application standpoint eGBSELs could be used across the full eSARD workflow. In essence, they are a SE complement to a design, implementation, and operations effort. However, they also represent a good foundation towards further optimization efforts. This tool presents the following basic characteristics:

- **Multisector**. They can be used within any DOI sector. However, they are ideal for the type of activity (e.g., implementation) where, tracking, capturing, and management are needed. On each one of these DOI sectors there are specific aspects related to geometry, behavior, and substance that an eGBSEL model will capture. Furthermore, they address the networked nature of an eSARD process and the complex reality behind evolutive system architectures, where multiple areas defining system characteristics, performance, and development are interconnected across scales and levels. Therefore, the domain for this models expands across sectors.
- **Data parameters**. These capture mostly quantifiable parameters, especially those figures of merit that are foundational to address eAMGs by the system solution. For instance, mass or power are traditional figures of merit in many technical fields, while the number of operations, data rates, or carbon footprint are key in others. All these parameters have ripple effects throughout the system, and they represent the best way to keep track of the evolution the system over multiple design cycles. Although if these parameters are quantifiable then multiple mathematical operations can be done with them, which paves the way to parametric studies as well as other optimization techniques. So, quantifiable rules are applied to both selection criteria and translation values.
- **Data reinforcement**. For each data parameter captured in these models there is a complementary and multisource information that can help understanding, evaluating, and even predicting patterns and conclusions in any subsequent studies. These reinforcements are not only alphanumerical in nature; thus colors, symbols, icons, images, diagrams, videos, and other types of information are elements used within this category. Any data parameter and associated reinforcement can always be multiple and linked to each other.
- **Links and connections**. Any parameter, reinforcement, or topic is not only a value but also a node within an

information network within this approach. To improve the visualization of these connections, as well as even to help categorizing them eGBSELs show all these connections both graphically and syntactically.

- **Lifecycle / development cycle**. Modeling an ever-changing evolutive system needs coordinates for such process. These include among others the following: [1] objective of the modeling (e.g., system, sub-system, component, etc.), [2] development phase and milestone (e.g., PDR, CDR, ORR, etc.) conditioning the level of detail, density of connections, etc., [3] lifecycle phase of the system (e.g., design, analysis, operations, decommission, etc.), and [4] temporal print and schedule. This is key since any eGBSEL develop within an eASM effort responds to a continuous cycle and will be eventually updated, upgraded, redone, or even finished along such design process.
- **Multidimensional**. However, these models are not bidimensional in any way since all connections among them and their evolution in time have a multidimensional nature as well. Thus, each parameter could also be understood as a vector or a matrix (link) addressing multiple states, generations, and parametric variations within the system.

eGBSELs operate around key figures of merit that complete, qualify, extend, quantify, and make feasible most relevant eAMGs of a system architecture. So, if such gaps are being addressed by seed geometries along design paths, these models capture and manage key parameters (figures of merit) and most relevant relationships behind them. For instance, in developing the same generic electro-mechanical system from previous examples, the packing factor was identified as an eAMG (Table 29). Associated to that gap is the volume of any subsystem that is integrated within such device, which becomes a figure of merit and therefore a parameter in the eGBSEL. Then, such parameter can be located within the eSARD design sector and the geometrical details of the GBS trifecta. Under such approach, this also relates to material properties (e.g., mass) and the selection of manufacturing technique (e.g., availability feasibility, cost). Furthermore, there are also links to other DOI areas such as operations (e.g., how the system is deployed), which also conditions how the system is integrated (implementation). In essence, this exemplifies the complexity behind an eSAR system and its design process. Although, these parameters are not only about measuring cubic meters and other analytical values, but also about integration procedures that are better explained through videos, images, etc. There are many other connections, some of them are previously known and some others are identified along the process, thus these eGBSELs need to capture a network of relationships. Figure 170 shows a simplified eGBSEL where several elements and operations are represented:

- **eGBSEL information** (top). This area captures topics such as system architecture, task, study, development phase, lifecycle step, timeline stamp, etc. These provide coordinates for all modeling tools, as well as they become a reference for other subsequent development efforts. Any variations or new generations can also use them.
- **In / out parameters** (left & right columns). These are connection ports for other eGBSELs to create links among them, as well as for the tool to provide outputs based on many inputs provided within the concept network.
- **Topic** such as system, subsystem, or subject (A#, 2nd left column). This area relates to any point that needs to be studied and tracked. They present a hierarchal and relational reference to list and group them at multiple levels.
- **DOI areas** (top second row) address all three key sectors in the eSARD methodology including the following:
  - **GBS areas** (columns below DOI area). Geometry, behavior, and substance overall detailed topics.
  - **Parameters** (PG#, PB#, PS#, within each GBS column). These are defined by variable identification, unit, description, etc. These parameters can have a quantifiable, quantifiable, or hybrid nature.
  - **Reinforcements** (RG#, RB#, RS#, next to parameter boxes) include images, videos, drawings, etc.
  - **Overall links** (LA#, next to system column) are ports connecting and managing any input within the eGBSEL.

eGBSELs serve as a complement or foundation for other more complex data-driven frameworks capable of performing more elaborated analysis. Nevertheless, within this layout, multiple operations can be done such as:

- **Listing** of parameters, connections, and associated parameters that will grow when more details are addressed.
- **Grouping** of those parameters enabling other analysis techniques.
- **Quantification** of parameters, mathematical analysis (e.g., total sums, etc.), etc.
- **Qualification** of those parameters thought reinforcement, links, color maps, and other methods.
- **Comparison** among entries and links, as well as identification of gaps.
- **Ranking** based upon multi-criteria and with regards to quantification, qualification, or hybrid principles.
- **Linking** between parameters, groups, areas, subtopics, etc.

- **Comparative analysis** based on all the previous points, as well as any other induced criteria.
- **Causality analysis** among entries, and especially links within the model.
- **Feasibility analysis** based on different criteria to identify gaps, contradictions, oppositions, etc.

Therefore, eGBSELs present a simple but powerful tool to address not only parameters associated with eAMGs, but also connections and links among development areas, system architecture subsystems, and even unknown relationships.

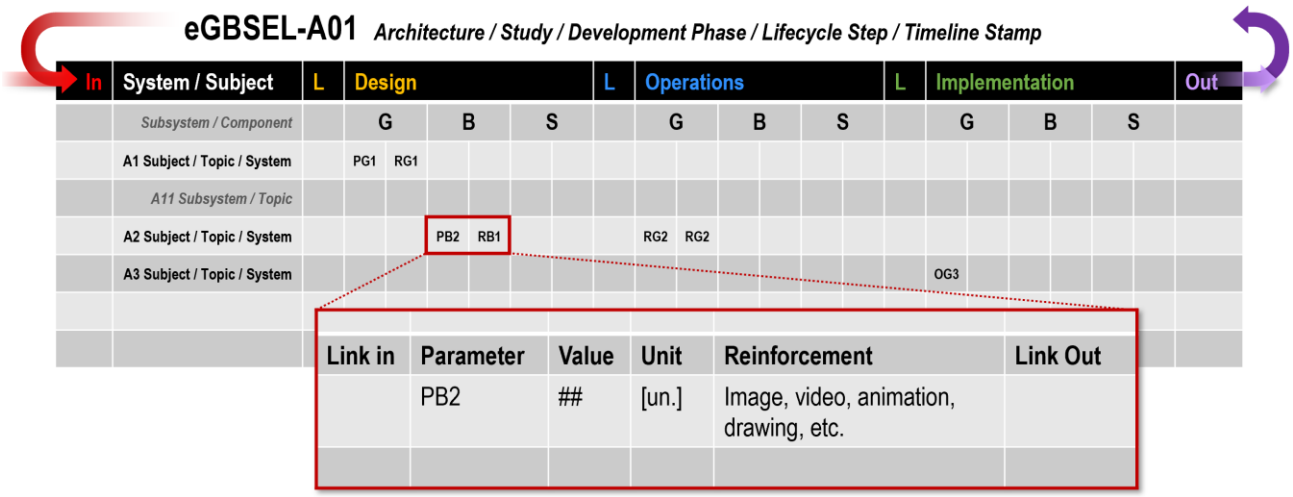



*Figure 170. Simplified version of a generic evolutive geometry, behavior, and substance equipmment list (eGBSEL).*

## 5.12. Architecture Maturity Levels (eAML)

While processes and tools are critical to mature any system architecture design, it is also important to assess what is the level of system development and maturation required or achieved with the following actions:

- Evaluate the solution from a maturity and feasibility standpoints.
- Assess remaining efforts to enable the completion of the system concept.
- Manage current and future design efforts and resources.
- Compare a specific solution against design alternatives, system variations, future updates, upgrades, etc.
- Support system performance evaluations and studies.
- Determine the importance of eAMGs for a given system architecture (qualification and qualification)
- Provide a quick reference for designers and workflows to assess the maturity of the system and manage future efforts.

Historically there have been several scales created to assess these levels in the evolution of a system, as well as some of its components as presented in section 3.2. Among others, here is a summary of the most relevant:

- **Technology readiness levels** (TRL). These were originally created by NASA in the 1970s to assess how mature a specific technology was, so multiple technology options could be evaluated towards a final system development (Mankins, 1995). Nowadays, TRLs are used across industries and countries worldwide (Tomaschek et al., 2016). There are nine levels addressing all different stages of a technology development lifecycle (Figure 171, NASA, 2014), such as: [1] basic principles and technology research, [2] technology concept formulation and feasibility proof, [3] demonstration of proof of concept, analytically and/or experimentally, [4] component and/or breadboard validation in a laboratory environment, [5] component and/or breadboard validation in a relevant environment, [6] system and subsystem model or prototype demonstration in a relevant environment, [7] demonstration of a system prototype, [8] system validated in a real environment, [9] actual system proven in real conditions (heritage). While this approach can guide the technology development through its lifecycle it does not address areas such as system design, system efficiency, and other system implementation topics.

- **Concept maturity levels** (CML). These have also been used in design and formulation work across multiple industries such as MedTech or aerospace. They are used to assess the maturity of a concept, as well as the capability of its development to meet basic requirements through different reviews and gates (Wessen et al., 2013). Nevertheless, this approach does not tackle in depth other implementation aspects since it only addresses if the concept is mature enough for a PDR level.
- **Integration readiness levels** (IRL) address the integration of system components and data among interfaces and across all different hierarchical levels (Jesus and Jr., 2018; Long, 2011). There are seven levels going from the interface between technologies (level 1) to the verification and validation of integrated technologies (level 7) (Eisner, 2011).
- **System maturity levels** (SML). These are based on both TRL and IRLs to assess several human subjective aspects in the process of maturing a system from a probabilistic standpoint (Tan et al., 2011).
- **Modeling maturity levels** (MML). These address modeling phases within a software development effort ((Kleppe et al., 2003) with the goal to improve processes while evaluating teams and workflows, etc.
- **Capability maturity models** (CMM) address the degree of formality, development, and optimization within a software project including initial, repeatable, defined, capable, and efficient levels (Paulk et al., 1993).
- **Implementation maturity models** (IMM) are an evolution of CMMs. It is applied to software implementation processes that consider human resources, information access, available means or resources, and control techniques, among others (Persse, 2001).
- **Testing maturity models** (TML) are based on CMM to assess testing capabilities. There are also five levels going from: [1] no testing, [2] definition, [3] integration, [4] management, and [5] optimization (Burnstein et al., 1998).
- **Manufacturing readiness levels** (MRL) assess the maturity of multiple manufacturing processes for industry assessments, supplier analysis, supply change studies, etc. There are ten MRL levels or threads (Blokdyk, 2019).

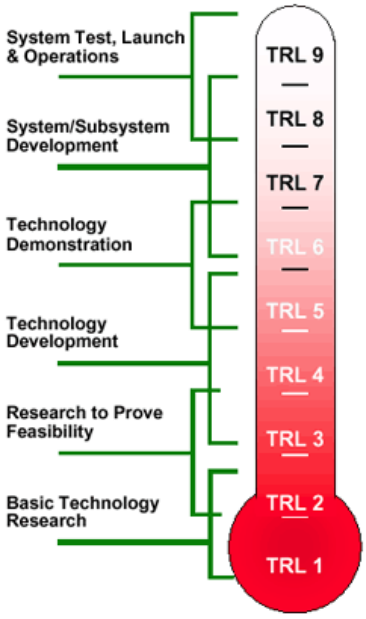



*Figure 171. TRL levels after NASA (2014).*

While these approaches assess the system, its components, and the system completion phase from a very specialized perspective, such perspective lacks synergy (Gove and Uzdzinski, 2013). For instance, these approaches are about [1] assessing maturation without implementation, [2] technology feasibility without addressing concept designs, and [3] system design perspective without assessing implementation schemes or solution alternatives. Contrarily, an evolutive architecture based on ARR principles is in constant change, and its subsequent evolutive design process (eSARD) considers all DOI aspects. Thus, to evaluate the maturity of an evolutive system architecture (eSAR), as well as to help managing multiple design efforts a more holistic index is needed based on some aforementioned principles behind these other scales.

**Architecture Maturity Levels** (AML) are such holistic reference scale that is used to address multiple maturity aspects within the eSARs development process. These levels address all three DOI areas by tackling these areas:

- **Definition.** This addresses the full system design, its operative scheme, and the implementation path. There are multiple levels and gates (e.g., PDR, CDR, etc.) within the design process. AMLs measure how close the system is to reach the level of definition required at each gate. The higher the AML is for a system design, the more definition it possesses in any DOI area (design, operations, and implementation).
- **Completeness**. AMGs present gaps in the completeness of a system independently from how defined such system design or its process could be. AMLs are also a tool to measure such completeness. The higher the number is within an AMG scale, the more relevant gap it becomes and the lower an associated AML level becomes (see Table 30).
- **Feasibility**. Finally and complementing such system completeness, this scale also considers how feasible a system design actually is. This assessment is critical and considers all three DOI areas too. This is especially key with new systems, since it measures how feasible the next generation of a system becomes. This also measures indirectly system performance and how much the new generation differs from direct or indirect heritage solutions (DOI). The

more feasible the system design, its operative mode, and all implementation paths are, the higher the AML is.

Therefore, the AML scale provides a number from one to nine for each DOI area. The lower the number, the less mature the system is in that area. Each DOI area of maturity assessment considers multiple geometry, behavior, and substance topics (GBS), as well the importance and severity of their associated AMGs. For instance, an evolutive architecture with an AML 354 has a low level three regarding design maturity (geometry), a higher level five of operation definition (behavior), and a level four with regards to implementation, feasibility, and description. Those three digits could be averaged to provide a single AML level addressing feasibility. In this case the average simplified AML will be four. Table 30 shows a more detailed description of these compound scale factors, and Figure 172 present a three-dimensional graphical representation of AML evaluation levels. AMLs are also a way to assess the importance and number of critical AMGs.

AMLs are used to plan, manage, and organize design efforts, as well as to create comparison metrics among: [1] alternative solutions, [2] system variations within the same design path (continuous design), and [3] alternative solutions regarding infusion options and integration. These levels can be applied to system, subsystem, components, and processes.

| Levels | Design | Operation | Implementation | Gates | Gaps |
|---|---|---|---|---|---|
| 1 | Research | Research | Research | Requirement | Concept / Heritage |
| 2 | Concept | Approach | Resources | SFR | Architectural level |
| 3 | Initial design | Scheme | Strategy | PDR | System level |
| 4 | Simulated design | Simulation | Simulation | CDR | Subsystem level |
| 5 | Prototyped design | Laboratory | Prototype | TRR | Component level |
| 6 | Tested design | Relevant Env. | Testing | SVR | Test level |
| 7 | Refined design | Analog Env. | One-off | ORR | Implementation level |
| 8 | Operated design | Real Environment | Short Series | OPRR | Operation level |
| 9 | Proven design | Feedback | Production | Real system | |

*Table 30. Evolutive architecture maturity levels (eAMLs) considering system design, operations, and implementation aspects.*

Thus, AML can be used for other purposes during the development of a new system architecture such as:

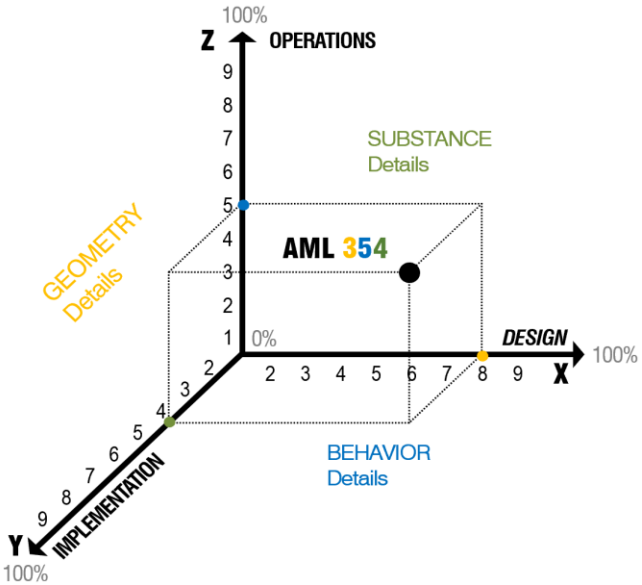



*Figure 172. AMG levels for system architecture and subsystems.*

- Design assessment and management.
- Evaluation and comparison among solutions.
- Assessment between design phases.
- Management and schedule of resources (e.g., workforce, computational power, etc.)
- Results assessment.
- Heritage validation.
- Technology infusion, assessment, and roadmaps.

AMLs also represent a simplified way to address the complexity of the process towards developing evolutive system architectures based upon ARR principles (adaptability, reactivity, and regeneration). These levels are not meant to become a rigid scale, but rather a flexible approach that can be tuned to the needs of the field, challenge, and culture, while providing a common ground across disciplines and systems.

## 5.13. Refining Design, Fast Synchronous Design Cycles

Once design flows and working frameworks have been outlined, maturity metrics established, and design principles and objectives defined, it is key to understand how this development approach advances towards addressing ARR design goals. The eSARD methodology tackles [1] overall architecture (macro scale) aspects, as well as [2] detailed discipline topics regarding subsystems (micro scale) simultaneously, to achieve the maturity of a system architecture fast and efficiently.

The evolutive approach combines systems engineering and design engineering methodologies so any system architecture solution is considered an instantiation within such continuous process. Thus, the eSARD approach requires multiple design cycles tackling subsequent system maturation phases, alternative design paths, and comparative studies among design species. This synchronous process is conducted within the 3C framework tackling simultaneously all DOI areas (design, operations, and implementation) and system levels, from multidisciplinary standpoints and across the system lifecycle. This synchronicity in the method is achieved by developing and improving the starting point of this process through AMGs, since they tackle key synergetic topics while facilitating efforts across teams, machines, and individuals. The eSARD process has a universal standpoint, but it is optimized towards complex hardware-based system architecture designs. As a workflow this process has a series of six networked phases among all DOI sectors as Table 31 summarizes below.

| Evolutive Fast Synchronous Design Cycles | | | | |
|---|---|---|---|---|
| | **Objectives** | **Inputs** | **Tools** | **Outputs** |
| Exploration | • Explore goals<br>• Explore trade space<br>• Identify AMG<br>• Assess drivers / ARR | • Requirements<br>• Constraints<br>• Goals | • eADQNs | • eADQNs<br>• eAMGs |
| Design | • Architecture space<br>• Trade space path<br>• eAMG strategy | • eADQNs<br>• eAMGs<br>• Client inputs | • eASGs<br>• Generative CAD<br>• Parametric CAD<br>• 3C sessions | • eASGs<br>• eASMs |
| Validation | • Basic analysis<br>• Prototyping<br>• Parametric studies<br>• ASG validation | • eADQNs<br>• eAMGs<br>• eASGs<br>• eASMs | • Multidisciplinary analysis<br>• Prototyping<br>• 3C Sessions | • eAML<br>• eASMs<br>• V&V path<br>• Testing path |
| Implementation | • Implementation path (manufacturing, coding fabrication, etc.)<br>• V&V strategy<br>• Substance path | • eADQNs<br>• eAMGs<br>• eASGs | • Impl-eASMs<br>• 3C sessions | • Impl-eASMs<br>• Prototype<br>• Production<br>• Deployment |
| Operations | • Ops-Con strategy<br>• Operative path<br>• Operative architecture | • eADQN<br>• eAMG<br>• eASG<br>• eASM | • Ops-eASM<br>• 3C sessions | • Ops-eASMs<br>• Ops-Con Arch. |
| Optimization | • Optimization path<br>• New eASGs<br>• Better efficiency<br>• Better performance<br>• Reuse / eepurpose<br>• Feedback | • eADQN<br>• eAMG<br>• eASG<br>• Analysis AMBS | • Opt-eASM<br>• 3C sessions | • Opt-eASMs<br>• New eASGs<br>• New eASMs |

Table 31. *Evolutive fast synchronous design cycles objectives, tools, and processes within the eSARD methodology.*

## 5.14. Metrics and Comparison

The eSARD process developed within this chapter addresses in general the design, development, implementation, and use of complex systems across their lifecycle, and specifically the needs of evolutive system architectures (ARR) as complex hardware-based systems. However, as important as those principles, approaches, tools, and techniques described are the need to assess and measure the process success is also critical. This is needed to evaluate any development effort at hand, as well as the management and integration of such methodology within teams, organizational cultures, and personal practices. Next sections present some key parameters and overall approaches with these purposes:

- Evaluating the results of the design effort as well as the utilization of resources.
- Comparing methods with other DSE methods (traditional or otherwise).
- Evaluating results and the relative cost of the full process for a given solution.

Hence, there are five initial areas that provide such initial reference framework for measuring success such as: [1] schedule and time, [2] complexity management, [3] system performance, [4] concept robustness, and finally [5] relative cost.

The eSARD process has been designed to implement several key characteristics that increase the level of success in addressing these key areas. Thus, the approach is built upon [1] systems engineering and design engineering state-of-the-art techniques, [2] key needs and characteristics of evolutive system architectures that are described by ARR principles, and finally [3] key gaps in those development techniques that have been addressed using method inspired by natural evolution. Among such key characteristics that were described in this section the most relevant are the following:

- **3C networked approach** provides a concurrent, collaborative, and communicative framework enabling faster and better communication among agents, and thus a more efficient design environment as previously described.
- **Continuous design** has an influence in the system as a product as well as in the methodology, leading towards reusing and repurposing. The principle of continuity allows to reduce future work while improving designs at hand.
- **Based on synergetic eAMGs**, the eSARD approach starts the design process from the most critical and limiting topics (gaps), ensuring better efficiency in developing system feasibility for each design.
- **Heritage** becomes a building block to validate and compare solutions, as well as a potential foundation towards new concepts. Because each design becomes heritage (parent generation), its process can be inherited as well.
- **SE+DE** are combined and interlinked within this DSE approach reducing lead times and improving effort efficiency.
- **ARR** objectives not only describe evolutive architecture needs, but they also increase by definition the potential efficiency of a system architecture and the subsequent design process (section 4).
- **Full cycle DOI principles** address, all three pillars of any system. Since the networked approach includes operations and implementation aspects in addition to any design effort, it allows to achieve more efficiency in the process. This is done by tackling early in the process potential issues that eventually can appear at the end of the lifecycle.
- **GBS details**. This approach is not only about high-level architecture design topics but also about ensuring system and sub-systems details are addressed especially if they can condition the closeness of the system (eAMGs).

Next sections address how these topics contribute to the efficiency and success of the eSARD approach. Table 32 presents a comparative summary highlighting metrics, main advantages, and differences with other approaches.

### 5.14.1. Schedule

In the design of complex systems, time is often a key resource due to schedule constraints and the subsequent cost of specialized workforce. The faster a design approach provides a feasible solution, the more efficient and therefore successful it would become. Thus, these characteristics of the eSARD process contribute to this rational by (see Table 32):

- Increasing the number and depth of topics being address simultaneously in the design process.
- Using synergetic points to advance the maturation process faster.
- Providing a holistic approach that considers both details and high-level architecture trades at the same time.
- Enabling a modular design process so results, models, and processes can be reused and repurposed.
- Using less workforce and design resources for the same level of results.

These differences make this approach especially different when compared with waterfall, spiral, or even linear SE

and DE methods (chapter 3). Linear workflows are less efficient and adaptable than networked methods (Figure 16) since they need multiple cycles to achieve system completion. Networked methods like eSARD are capable instead of distributing resources (including time) upon areas requiring more development. Metrics to assess time efficiency include among others:

- Number of design topics per unit of time.
- Number of disciplines simultaneously being address per unit of time.
- Number of people/seats/agents on a design topic per unit of time.

### 5.14.2. Complexity Management

Complex systems in general, and those requiring a combination of software and hardware in particular need a design approach capable of tackling different scales, perspectives, lifecycles, and disciples from both sides. The eSARD approach must address all DOI areas from a hardware and software perspectives. Furthermore, complexity management entails a combination of: [1] design scales including components (details), subsystem, and system levels, [2] collaboration of multiple disciplines, [3] multiple agents, and [4] multiple versions and generations, etc. The more complexity a process can handle with less resources, the more capable and efficient such process becomes. Thus, eSARD characteristics allow:

- To handle synergy and multidisciplinary knowledge more efficiently.
- To tackle multiple-level design efforts.
- To handle continuous design processes at any stage of the system lifecycle.

As previous sections and chapters already have developed, the overall approach of the eSARD methodology is to address ARR system architectures characteristics. So, multiple metrics could be associated with these principles, such as:

- System scales and system levels addressed by the process.
- Lifecycles phases tackled within the effort.
- Number of parameters addressed and managed per design effort.

### 5.14.3. System Performance

A key goal behind the eSARD approach, and in general of any design effort is to improve or achieve better levels of system performance, independently of the field or application. The faster, easier, and with less resources this is done, the more efficient and successful a design process will become. Not all design efforts are driven by a certain system performance threshold that must be obtained, so the capability of such process to improve designs is still a key characteristic of most system design processes. In general, from a metrics perspective eSARD enables an approach oriented towards:

- The use of heritage solutions as both building blocks and validation inputs across the lifecycle.
- Tackling a system design by considering its most critical gaps towards system feasibility (eAMGs).
- A multi-level look at the design process from a system characteristics (ARR) standpoint developing principles (DOI) and details (GSB) as much synergistically, simultaneously, and concurrently as possible.
- Parametric and algorithmic models used in data-driven methods to enhance system performance by finding a balance across opposing design forces. For instance, an eSARD approach would tackle the balance between mass, structural behavior, thermal performance, and aesthetics through open (secondary) and close (primary) design variables.

This area is indeed complicated to measure due to the large universe of options. However, there are still some initial metrics that could be identified. Like in the previous section, these parameters are just one rather small group of starting points that are used to understand the nature of the process. These are some of the most relevant metrics among others:

- Comparative levels of system performance variables across the full system architecture.
- Number of performance variables or figures of merit per design or optimization effort.
- Duration of the design effort to achieve a certain performance level.
- Number of resources per unit of time to achieve a certain performance level.

### 5.14.4. Robustness

Beyond performance and complexity the eSARD approach tackles another key metric, the robustness of the system design across its concept, model, implementation, and operations. These levels can be addressed together or independently,

but in all cases they refer to how the system handles: [1] changes in the environment, [2] GBS challenges, and [3] variations of requirements. The more external changes the system can handle the more robust it will become, reducing the potential need for new design efforts, while increasing performance and reducing cost. The eSARD methodology responds to this principle using the following strategies:

- The use of heritage within a continuous approach enables to achieve improvements easier across the process.
- The eSARD approach starts from the weakest and most critical points (eAMG) in the system maturation workflow.
- All BGS details are addressed synergistically considering design (geometry), systems (functions), and implementation (substance) aspects simultaneously across all levels and components.
- The DOI approach also tackles simultaneously key details in all previous points.

These are some metrics associated to these principles among others:

- Number of variations and alternatives per system design.
- Number of gaps (eAMGs) addressed within a single design effort.
- Topics, parameters, and figures of merit per system design as well as per design effort.

### 5.14.5. Relative Cost and Use of Resources

All previous points are directly connected and linked among them when it comes to cost and the use of resources. Resources include materials, energy, code, and other aspects under the system substance concept. They also include the need of workforce, computational power, and other procedural resources associated with the design workflow itself. In general, the more any system architecture embraces key ARR principles of adaptability, reactivity, and regeneration (chapter 4), the more cost effective will be. Thus, eSARD characteristics allow the following strategies:

- Reduce the need of resources (e.g., agents) across the whole design process, by enabling more with less.
- Increase the adaptability of the solution and the design process by doing better with less.
- Address multiple design strategies and paths simultaneously, which improves the return of the design effort.

Similarly, from the overarching perspective of the cost factor there are multiple parameters to be studied such as:

- Number of resources per system design or per design effort.
- Gaps and figures of merit per resource parameter.
- Reutilization ratio of previous design efforts or heritage solutions.
- Cost of design resources per BGS detail or DOI milestone.

### 5.14.6. Metrics Summary

In conclusion and based on previous sections there are several initial parameters serving as a foundation for a success metrics and a comparison framework. These parameters can be studied as relative percentages, making the comparison among them easier, while allowing to create an efficiency footprint regarding its success as a process. Figure 173 shows a summary and a relative comparison with other techniques and methods. The higher the percentage of each parameter is, the better it will be. These parameters include among others the following:

- **Number and nature of design topics being tackled simultaneously**. This includes the capability to handle both qualitative and quantitative variables. This parameter addresses the bandwidth of the process and its capability to provide more efficient systems and workflows in terms of schedule, resources, and quality combined.
- **Number of disciplines or disciplinary efforts addressed** is a key metric in any design process. The more and the deeper that multiple disciplines can be tackled, the more efficient the process will be. In essence this addresses the efficacy of the approach since it is not only about disciplinary results, but also the management of such efforts what counts as a metric of the workflow. The more disciplines being tackled, the higher the success percentage is.
- **Number of people and agents** involved in the process is also critical since it has direct consequences towards cost, management, feasibility, liability, etc. This metric is about measuring and applying the evolutive principle of doing better with less, while indirectly addressing management and scalability. Design agents in the process include people, machines, workflows, and combinations. The smaller number of agents is, the higher the success percentage is.
- **Number of design alternatives considered**. The eSARD approach is developed with the need of tackling multiple

design paths in mind. An evolutive system is not only one instance, but a family of solutions. This principle has direct consequences in such metrics, since it allows to [1] reduce future efforts when upgrades or redesigns are needed, as well to [2] reduce impact, time, and cost towards other design efforts that can reuse system or workflow parts.

- **The amount and depth of systems and subsystems** measures the capability of the process to address both details and overarching architecture topics as part of the design process. This has consequences across the methodology as well as the use of resources. The more diverse in scale the workflow can be, the higher the success percentage will be too since it would allow to tackle the eSARD methods from deeper and broader perspectives.
- **Relative cost and resources**. This metric is related to all the above parameters, and it summarizes the essence of the system efficiency in terms of resources as well as the result as a system architecture. As such, this metric needs to be tackled from the perspective of all the resources used to run the design process for a given system design and a specific system performance. The higher performance the system obtains with less design resources used in the effort, the more affordable and efficient will become.

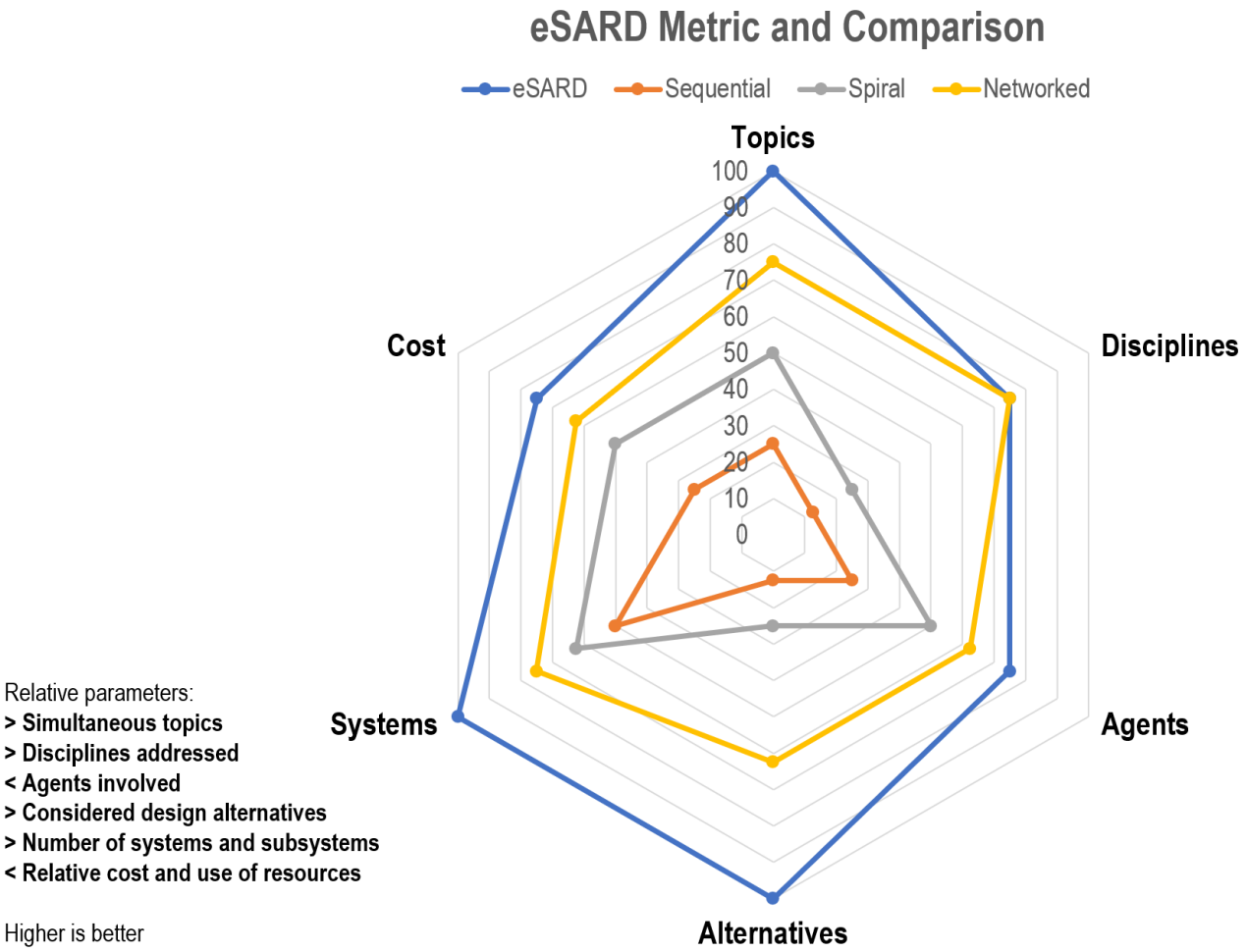



*Figure 173. eSARD metrics parameters and relative comparison with other approaches. The higher the number, the better.*

Graphically, overarching metrics and general comparisons among eSARD and other methods such as sequential (linear or waterfall), spiral, and networked (chapter 3) is rendered in Figure 173. Furthermore Table 32 presents a summary of all previous points within this section.

| eSARD Characteristics, Success and Metrics | | | | | | |
|---|---|---|---|---|---|---|
| Charact. / Areas | Time | Complexity | Performance | Robustness | Cost | Parameters |
| 3C | • More efficient<br>• Faster efforts | • Easier complexity<br>• Less | • Leaner processes<br>• Better performance | • Better solutions<br>• Broader approach | • Less resources<br>• Less agents | Agents |
| Metrics | Designs / time | Topics / time | Disciplines / design | Alternatives / design | Design / agent | |
| Continuous | • Repurpose<br>• Easier upgrades | • More generations<br>• Heritage use | • Better solutions<br>• Heritage enabler | • Broader views<br>• More alternatives | • Faster cycles<br>• More efficiency | Alternatives |
| Metrics | Alternatives / effort | Alternatives / design | Alternative / F. merit | Designs / alternative | Resources / agent | |
| eAMGs | • Faster efforts<br>• Easier changes | • More complexity<br>• More resilience | • Better performance<br>• Better optimization | • More robust sys.<br>• Less risk | • Less cost<br>• Better optimization | Disciplines Topics |
| Metrics | Gaps / effort | Gaps / design | Gaps / F. merit | Number of gaps | Gaps / agent | |
| SE+DE | • Faster efforts<br>• Easier mgmt. | • More standpoints<br>• Full optimization | • Full parametrics<br>• More efficiency | • More options<br>• More adaptability | • More completeness<br>• Less resources | Systems |
| Metrics | F. Merit / effort | F. Merit / design | Figures of merit | F. Merit / variations | F. Merit / agent | |
| ARR | • Evolutive<br>• Faster progress | • More adaptability<br>• Better with less | • Better performance<br>• Beyond sustainable | • More alternatives<br>• Better interactions | • Less resources<br>• Evolutive response | Alternatives |
| Metrics | Topics / effort | Topics / design | Topics / F. merit | Topics / variations | Topics / resources | |
| DOI | • Easier develop.<br>• Better progress | • Real full lifecycle<br>• More disciples | • Full optimization<br>• Less resources | • Lifecycle synergy<br>• More connections | • Less waste<br>• Lifecycle opt. | Systems |
| Metrics | Lifecycle phases | L. phases / design | L. Phases / F. merit | L. P. / variations | L. P. / agent | |
| BGS | • Better implement.<br>• Easier progress | • Full lifecycle details<br>• Better feasibility | • Greater synergy<br>• Less resources | • More details dev.<br>• More compatibility | • More compatibility<br>• More recycling | Cost |
| Metrics | Parameters / effort | Parameters / design | Parameters / gaps | Parameters / var. | Param. / resource | |
| Main eSARD Advantage | More efficient and faster workflow | Broader and deeper design activities | More capable and synergetic systems | Better trade and design exploration | Less resources, people, and time | |

*Table 32. Summary matrix of eSARD success and comparison metrics.*

## 5.15. Conclusion

Current design methodologies are mainly based on a 'divide-and-conquer' approach, implementing an 'in-line' or sequential system design approach. While many of methods consider multiple disciplines, generally those are not developed from an integrated standpoint. In response to the special characteristics of evolutive system architectures presented in chapter 4 and summarized under the principles of adaptability, reactivity, and regeneration (ARR), the eSARD approach is developed to address both the overall process as well as key all principles, objectives, and tools behind its evolutive methodology. This method is inspired by evolutionary mechanisms (section 5.1), and some key gaps within stat-of-the-art DE and SE techniques (chapter 4) to create a more efficient, faster, and more capable approach that addresses the special needs of eSAR system architectures as section 5.2 previously introduced. Furthermore, this methodology integrates tools and methods coming from other DES techniques, while creates and modifies existing tools to provide a dynamic and highly adaptable workflow tackling the system design process at any level.

ARR basic principles provide a reference framework for the eSARD method that is based upon the evolutive design tetrahedron. Such construct allows to address the design workflow at different levels, including: [1] high level system architecture characteristics (ARR) describing the special needs and capabilities of eSAR systems, [2] geometry, behavior, and substance (GBS) system details where a hardware-based design workflow defines, conceptualizes, and implements any system, and finally [3] the detailed design workflow associated to all these scales considering design, implementation, and operations (DOI) system topics required to fully implement any system architecture design. Thus, this method addresses a system that is developed across scales and lifecycle phases as section 5.2 in general, and section 5.3 in particular presented. The eSARD process introduces a holistic approach that addresses the full design lifecycle process by tackling

all system scales concurrently, synergistically, and efficiently. Such approach is also based upon current methods, and new toolsets specifically design for evolutive system architectures (Table 25).

Section 5.4 presented eSARD most critical design objectives within the evolutive reference framework and around the areas of adaptable design, reactive system performance, and finally regenerative use of resources. Such objectives lead to an overall approach that minimizes cost, addresses cultural influence, enables multiple concurrent design paths simultaneously to finally provide enough details to make eSARD a competitive methodology (section 5.4.4).

Design objectives set the foundation for key design principles (section 5.5) used across the eSARD methodology to guide all design efforts and activities. These principles include: [1] 'doing better with less', by defeating a problem by design (adaptability, section 5.5.1), [2] designing 'smarter with less', through continuous solutions and operations (reactivity, section 5.5.2), and [3] doing 'more with less', by addressing the optimization of all resource utilization across the full design and system lifecycle (regeneration, section 5.5.3).

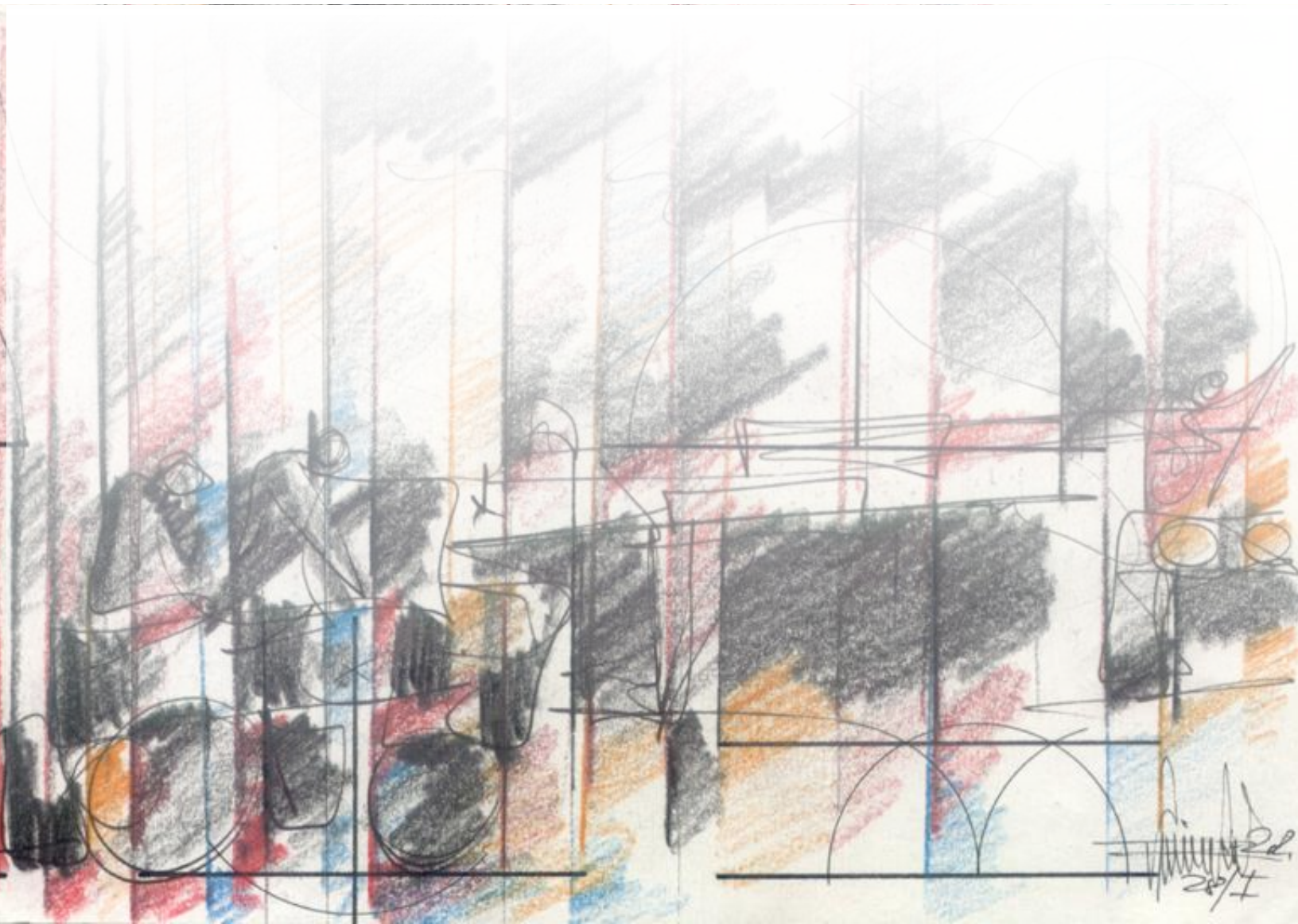
Once all foundational aspects of the eSARD approach have been laid out, this chapter presented the model and subsequent diagram used to describe, organize, manage, and implement all multiple steps, activities, milestones, products, and tools concurrently used within an eSARD workflow (section 5.6). This is the eSARD helix model or eSARD_he (Figure 144). Such model provides a simplified two-dimensional representation of a highly networked, concurrent, and in some ways three-dimensional design activity and workflow (section 5.6.1). This model is organized around three sectors based upon all DOI areas (design, implementation, and operations) that are always happening simultaneously. Each sector is in essence a Vee model, with all three sectors integrated around a spiral that describes the full cycle of the system. Such diagram presents a series of milestones or design gates in every vertex, which keep increasing the maturity and definition of the system across lifecycle phases and development areas (GBS). The edges of those sectors tackle critical and interconnected design areas, such as: design, systems engineering, implementation, verification and testing, operations, and full system optimization. Within every edge there are a series of tools and activities used to create products needed for critical review and milestones, but most importantly to address, study and develop the system architecture design at hand. Section 5.6.1 presents all key elements of this diagram, while section 5.6.2 describes in detail the first design sector within the model and the operations taking place within it. Finally, sections 5.6.3 and 5.6.4 introduce other topics regarding the implementation and operative sector, respectively. Furthermore, this level of interconnection requires tools to evaluate the state of development among sectors, which is done through verification loops that were described in section 5.6.5. Finally, an approach like this not only requires a model to describe operations, but a workspace framework that enables physically (teams), virtually (collaborative), and digitally (data) an efficient use and infusion of the method. This is the 3C evolutive framework (concurrent, collaborative, and communicative described in section 5.6.6.

In the development of the eSARD methodology, this chapter also introduced a description of its subsequent design workflow (section 5.7) and presented in detail all key evolutive design tools used in the first sector. This approach is based on identifying the most relevant and synergetic gaps within a system design through a series of curated and highly multidisciplinary questions or eADQNs (section 5.8). These questions tackle those aspects conditioning feasibility, efficiency, and implementation of an evolutive system concept. The most critical among them becomes a system maturation gap or eAMG (section 5.9) which sets the starting point of the eSARD design process. From here, first solutions start to form through a series of subsequent design models and tools called seed geometries or eASGs (section 5.10). Such designs then lead to the creation of system models or eASMs (section 0) using a series of modified SE tools that bring the solution to a certain level of maturity and definition which is measured by evolutive maturity levels or eAMLs (section 5.12). Similarly to TRLS, and CMLS, these levels allow to organize, compare, and evaluate design solutions that are developed under a series of fast and synchronous design cycles (section 0). Finally, this chapter provides a series of metrics and comparison principles to assess design progress and compare efforts among techniques, frameworks, and system solution (section 5.14).

With a multidisciplinary approach, this research activity and associated practice complements current state-of-the-art design and system engineering methodology trends while opening a new and complementary path, especially towards complex hardware-based system architectures. Nevertheless, this approach is independent from the technical area of application and considers links among hardware and software-based perspectives.

# STUDY CASE
## Evolutive Micro-habitat Architecture
## CHAPTER 6

*"Simplicity is complexity resolved".*
***Constantin Brancusi***



# 6. Study Case: Evolutive Micro-Habitat Architecture

This chapter applies many of the methods and tools explained in previous sections to a generic study case. The subject of this study is a hardware-based system, which requires the consideration of several evolutive topics within the areas of adaptability, reactivity, and regeneration. The goal of this chapter is not to present a fully detailed solution for the design challenge at hand, but rather to showcase key aspects of the eSARD workflow and methodology. Furthermore, the chapter emphasizes main differences with other methods, as well as some potential metrics to be used at different phases of the process. The eSARD process used here is not a rigid process, thus this research only presents a foundation that can be later modified, customized, enhanced, reduced, or complemented by other methodologies depending on the application.

## 6.1. The Architecture Field of Micro-Habitats

Habitat architecture design in general is a complex and potentially highly evolutive field for complex hardware-based systems. Among many applications and design practices are buildings, houses, technical habitats, shelters, etc. Among them there is a specific subset of system architectures that is very interesting as an example, small scales habitats or micro-habitats (Horden, 2010). These small buildings include off-grid shelters, research outpost, mountain retreats, viewpoints, emergency shelter, treehouses, and playhouses, among others. However, this field is complex, multidisciplinary, and quite specialized (Figure 174). It also presents a long heritage over centuries with regards to functionality, implementation, and design approaches. Among their many generic characteristics these are quite common across this field:

- **Compact size**. These habitats tend to be very small with a modular design and structure.
- **Transportability**. Often these architecture systems are portable, transportable, and prefabricated.
- **Advance materials**. Due to their especial application, size, and experimental nature these showcase new materials and manufacturing techniques that are not often used or considered in other larger size constructions.
- **Lightweight mass**. Due to all previous points these tend to be very lightweight since multiple uncommon and high-tech techniques, such as inflatables, lightweight panels, tensegrities, new structural schemes, etc. are used.

Hence, all these micro-habitats in general, and specifically those technical shelters requiring off-grid capabilities and highly portability related to basic evolutive characteristics (ARR) as the following points summarize:

- **Adaptability**. These habitats need to respond to multiple changing weather conditions, different user needs, and usage schemes over the system lifetime. Furthermore, such system architectures need to respond to issues with construction material availability, workforce knowledge and skills, and transport feasibility, among many more issues.
- **Reactivity**. Dwelling systems such as these, even if they are temporary, need to react at the most basic level to all environmental conditions. For instance, they need to provide and retain heat in cold weather, provide cooling mechanisms in hot climates, manage open and close spaces based on usage conditions, etc.
- **Regeneration**. Off-grid sustainable habitats need to create power and manage waste across multiple conditions too. While system performance and capability might change, the consequences of this will affect the whole system.

Furthermore, the field of micro-habitats is part of a broader architectural field in line with all design stressors described on chapter 2, such as: climate change, energy needs, complexity, need for better performance, the pressure of heritage, and resource scarcity due to population growth, among others. Thus, this field of micro-habitats presents a perfect area of study due to its complexity and the context of operations. While these small-size habitats (Richardson, 2009) are multidisciplinary in nature and present a long history of heritage (centuries in some cases), however they have not evolved much during the last decades. Many disciplines involved in their development are quantifiable such energy use, use of material, etc. However, other topics such as aesthetics and user experience are difficult to quantify but very relevant towards the qualification and acceptance of any system solution. So, these highly portable and off-grid habitats require quantum loops in terms of system performance and user experience if modern and future standard of comfort need to be achieved. Furthermore, energy, structure, and operations performance are also key during the lifetime of the system with a subsequent great impact in our everyday life as users worldwide. Within such vast area of design, this field of microarchitecture tackling small habitats or as lately described as 'tiny homes' (Couto, 2016) presents the ideal candidate for this methodology due to all technical hurdles involved in the design process and the limitations in terms of resources, scale, transport, etc.

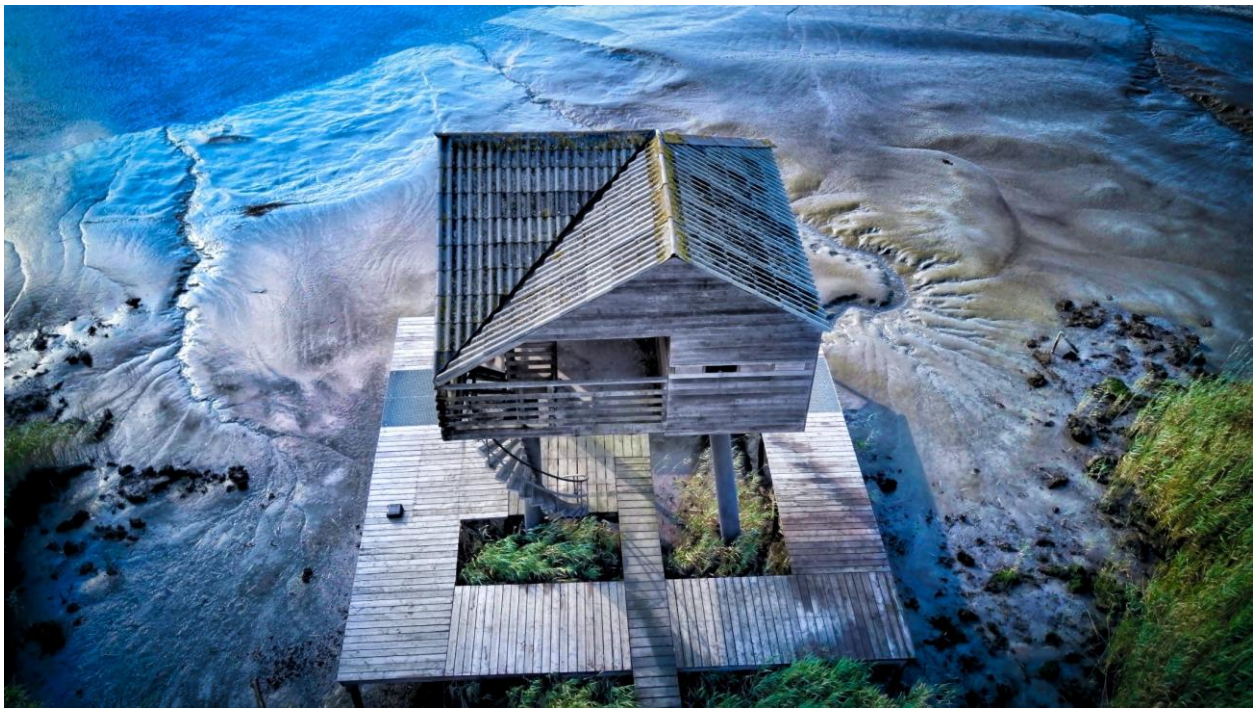


*Figure 174. Example of micro habitat or microarchitecture in the Netherlands (Reiderwolder Polderdijk, 9688 Drieborg).*

## 6.2. Heritage

Within an evolutive design process the study of heritage is very important since not only it provides validation schemes, but also enables to assess and create new design paths if used properly. When it comes to micro habitats, there have been multiple instantiations and application of the concept of a small habitat over time. Since this is not the objective of this research Table 33 only presents a short summary of examples and relevant heritage solutions for small habitats.

| Name | Description | Period | Adaptability | Reactivity | Regeneration | Reference |
|---|---|---|---|---|---|---|
| | | | ***Design*** | ***Operations*** | ***Implementation*** | |
| ***Classical*** | | | | | | |
| **Tempietto** | Bramante's small temple at San Pietro in Montorio (Rome). *Religious and representative use.* | S. XVI | New design<br>New approach | Experiencing the outside of a small building as the key feature. | New construction techniques are tested and infused. | (Roth, 1994)<br>(Benevolo, 1977)<br>(Freiberg, 2014)<br>(Markus, 2008) |
| ***Technical*** | | | | | | |
| **Shelters** | Small havens which are portable or transportable. *Technical dwelling and survival use.* | S.XIX | Low maintenance<br>Low design effort | Construction and operations in multiple remote areas | In situ resources utilization approaches | (Beard, 2020) |
| **Tents** | Highly portable temporary shelters. *Survival, technical, and recreational use.* | S. I | Lightweight structures<br>Tensile structures<br>Multiple conditions | Fast manual setup | Repurposing | (Horning, 2009) |
| ***High-Tech*** | | | | | | |
| **Micro-architecture** | Small temporary, portable, and permanent buildings. *Recreational, research, and dwelling use.* | S. XX | Multiple locations<br>Extreme conditions | Multiple uses | Off-grid capabilities<br>Repurposing | (Horden, 2008)<br>(Richardson, 2009)<br>(Richardson, 2007) |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Inflatable habitats** | Small temporary buildings. *Recreational, survival, research, and dwelling use.* | S. XX | Multiple locations<br>Extreme conditions<br>Highly compactable | High transportability<br>Extreme conditions | ISRU<br>Air as construction material | (Francis, 2019) |
| ***Conceptual*** | | | | | | |
| **Mobile** | Highly portable habitats. *Recreational, research, mobile, and dwelling use.* | S. XX | Multiple locations<br>Highly portable<br>New designs | Multiple uses<br>Limited uses | Off-grid capabilities<br>Repurposing | (Willemin, 2004)<br>(Roke, 2017)<br>(Siegal, 2002) |
| **Trailers** | Highly portable habitats. *Recreational, mobile, and dwelling use.* | S. XX | Multiple locations<br>Highly portable<br>Vehicle-driven | Multiple uses<br>Limited uses | Off-grid capabilities | (Keister, 2008)<br>(Wood, 2002) |
| ***Tiny Homes*** | | | | | | |
| **Small Houses** | Small and temporary buildings. *Recreational, research, emergency, and dwelling use.* | S. XXI | Multiple locations<br>New designs | Multiple uses | New materials<br>Repurposing | (Couto, 2016)<br>(Roke, 2016)<br>(Zeiger, 2009)<br>(Kahn, 2012) |
| **Cabins** | Small and temporary secondary dwellings. *Recreational, research, and dwelling use.* | S. XX | Multiple locations<br>New designs | Multiple uses<br>Multiple climates | Off-grid capabilities | (Jodidio, 2018) |
| **Treehouses** | *Recreational, research, play, and dwelling use.* | S. XVIII | Multiple locations<br>New designs | Limited uses | New materials<br>Recycling | (Jodidio, 2018)<br>(Nelson, 2020) |

*Table 33. Summary of micro habitats across history, uses, and DOI critical aspects.*

*Figure 175. [Left, top] El tempietto in San Pietro in Montorio by Bramante (Markus, 2008) [Top, center] Small portable aluminum trailer (Meyers, D.) [Right, top] Micro-architecture habitat for warm weather conditions (Samoh, A.) [Left, bottom] Small portable tent for hot climates (Hendry, P.) [Right, bottom] Technical outpost in the mountains (Nir, A.)*

## 6.3. Study Case Approach

The general objective of this study case is to address the first steps of an eSARD workflow to design a highly portable habitat with many evolutive characteristics and needs. Furthermore, the goal behind this chapter is to apply the evolutive design workflow while providing more detailed insights, comparisons, and metrics regarding this research. Since design is the highlight of this initial research, this chapter does not provide a complete full eSARD process addressing implementation, and operations (DOI). However, such areas will be addressed superficially and links with them highlighted. The example selected for this chapter is an evolutive portable habitat (EPH) developed by the author for multiple natural catastrophe and dwelling scenarios. EPH features design, manufacturing, and energy challenges across its lifecycle. The author keeps the copyright of this design for any use and purpose worldwide.

### 6.3.1. Design Objectives and Requirements

The main objective is to address the design of a portable habitat capable of quickly deploying an extremely compactable living volume (including floor, walls, and ceiling), while serving as a baseline to customize multiple environmental protection variations. There are several key areas of ARR requirements for this exercise:

- **Adaptability** (geometry). Such habitat should be able to provide a wide geometrical adaptability to address portability, deployment, changes in use, as well as other environmental changes. Furthermore, the geometrical solution should enable an easy adaptability to multiple designs and customizations later on with a minimum extra cost. The habitat must serve both temporary and permanent dwelling situations without compromising transportability.
- **Reactivity** (operations). Deployment operations for this habitat should be simple, deterministic, and potentially automated. The number of components and parts should be minimized to a maximum. This system architecture should be able to function under different weather and environmental conditions. The habitat must be transportable within available shipping options and reduce the number of people required for integration, deployment, and transport. Any deployed structure must consider aerodynamics to reduce wind loads, as well as water and snow accumulations.
- **Regeneration** (energy and material). Any selected design must enable a sufficient energy production capability to be fully sustainable under multiple environmental condition. This could be done using solar cells or other energy production systems, which means the habitat must deploy enough surface and/or volume for such systems. A low energy approach is emphasized for manufacturing, deployment, and operations across the lifecycle. From a materials standpoint, the habitat must reduce mass as much as possible, while using recyclable materials with easy repairability and manufacturing. Mechanical and material constraints are driven by market availability restraining solutions that require exotic materials and very expensive mechanisms that are not commonly available.

### 6.3.2. Heritage and State-of-the-art

There is not much heritage regarding portable habitats capable of providing enough energy for a full off-grid use. Furthermore, highly portable solutions such as tents and other technical shelters (Table 33) tend to be temporary in nature, while presenting complicated deployment systems. However, such solutions have a long history of use and present an excellent validation path as building blocks(heritage) for a new generation of ARR dwelling systems. Table 33 references present state-of-the-art solutions for similar challenges, which are not shown due to succinctness reasons. In summary though, this solution would represent a quantum leap in terms of system performance if compared with heritage solutions.

### 6.3.3. Figures of merit

Figure 176 summarizes objectives and initial figures of merit for this concept design including these ones:

- **Mass**. The more lightweight the solution is, the more compatible with ARR principles it will become. Less mass also means potentially less cost and less energy during manufacturing, transport, and operations.
- **Deployed Surface**. The more surface is deployed, the more volume and energy production can be achieved.
- **Number of parts.** This refers indirectly to system complexity and design efficiency considering actuators, manufacturing parts, and integration steps, among others. The lower the number, the higher the efficiency of the solution will be in the long term.


*Figure 176. Summary of objectives for the eSARD study case and initial figures of merit. © 2020 Raul Polit Casillas (patent pending).*

## 6.4. Approach and Set-Up

The eSARD approach dives into all design, implementation, and operation aspects as it was described in chapter 5 by using a networked approach. This means the process expands and contracts sequentially, addressing from higher-level constraints (e.g., architecture design decisions), to lower-level details and vice versa. As developed in previous sections and shown in Figure 177, multiple milestones and tools are used both sequentially and within a networked model to gradually improve the maturity of the system, while addressing its feasibility and efficiency. Next sections will develop this in detail.

### 6.4.1. 3C Working Environment

The first step once a design effort is programmed is to set up a working framework. This applies regardless of being a one-person effort, a team activity, or a combination of automated processes (machines) and workforce teams. The following table summarizes considerations, tools, and schemes on all different cases regarding an evolutive working context.

| | Agents | Uses / Applications | Tools | Connections | eSARD Examples |
|---|---|---|---|---|---|
| **CONCURRENT** | *Multiple design cycles \| Networked tools and activities \| All DOI sectors \| Facilitated efforts \| Scalable \| Highly dynamic* | | | | |
| **Analytical Framework** | Individual | 1. Capturing, analysis, exploration, and study of quantifiable parameters, and/or scaled qualifiable parameters.<br>2. Process storytelling<br>3. Parametric and generative studies | Notebooks<br>Mathematical models<br>Spreadsheets | Concepts<br>Parameters<br>Data<br>Models<br>Analysis<br>Tools<br>Designs | Design parameters<br>Figures of merit<br>eASMs system models |
| | Team | | + Databases<br>+ Shared models<br>+ WIKI-like tools<br>+ Algorithms | | + Interconnected Eng. models (CAD, BIM, MBSE)<br>+ Data-driven models |
| | Machine | | | | + Generative workflows<br>+ AI-driven models |
| **Geometrical Framework** | Individual | 1. Capturing, analysis, exploration, and study of system geometries.<br>2. Process storytelling regarding all DOI aspects<br>3. Facilitated efforts | Sketchbooks<br>3D computer models<br>4D models (movies)<br>Mock-ups | + Geometries<br>+ Models<br>+ Heritage<br>+ Concepts<br>+ Parameters<br>+ Data<br>+ Analysis<br>+ Designs | Evolutive sketches<br>eASGs system gaps<br>Interconnected Eng. models (CAD, BIM, MBSE) |
| | Team | | + Whiteboard efforts<br>+ Computer assemblies<br>+ Prototypes | | + Assemblies<br>+ Research |
| | Machine | | + Visual databases<br>+ Inferred designs<br>+ Cloud models | | + Generative workflows<br>+ AI-driven models |

| | | | | | |
|---|---|---|---|---|---|
| **COLLABORATIVE** | *Co-authored efforts \| Jam design sessions \| Constant revisions \| Scalable \| Highly dynamic* | | | | |
| **Discipline Dynamics** | Individual | 1. Coordinate work by multiple seats / disciplines<br>2. Identify synergies within disciplinary detailed work and models | Disciplines / seats<br>Tools and models | DOI sectors<br>BGS details<br>Parameters (qual / quant)<br>Models | eADQNs<br>eAMGs system gaps |
| | Team | | + Specialized discipline workflows (human) | | |
| | Machine | | + Automated processes<br>+ Artificial intelligences | | |
| **Agent Dynamics** | Individual | 1. Coordinate agents, seats, and disciplines<br>2. Compensate for biases<br>3. Manage moods, lack of resources, failures, etc. | Disciplines / Seats<br>Tools and models | + Agents<br>+ Interactions | eADQNs<br>eAMGs system gaps<br>eAMLs maturity levels |
| | Team | | + Human agents | | |
| | Machine | | + Automated processes<br>+ Artificial intelligences | | |
| **COMMUNICATIVE** | *Curated and facilitated effort \| All hands and all resources on deck approach \| Highly dynamic \| Organizational culture* | | | | |
| **Design Exchange** | Individual | 1. Foster disruption<br>2. Assess heritage<br>3. Coordinate design | Analytical framework<br>Geometrical framework<br>Requirements | Concepts<br>Models | eADQNs<br>eAMGs system gaps<br>eAMLs maturity level<br>eASGs architectures<br>eASMs system models |
| | Team | | + Specialized agents | | |
| | Machine | | + Automated processes | | |
| **Discipline Exchange** | Individual | 1. Foster synergy<br>2. Improve info exchange | Disciplines / seats<br>Tools and models | + Details models<br>+ Efforts<br>+ Protocols | eADQNs<br>eASGs system gaps<br>eASMs system models |
| | Team | | + Human agents need | | |
| | Machine | | + Automated processes | | |
| **Agent Exchange** | Individual | 1. Foster synergy<br>2. Promote data dialog<br>2. Enhanced exchange | Disciplines / seats<br>Customer needs | + Agents<br>+ Models<br>+ Efforts<br>+ Moods<br>+ Protocols<br>+ Cultural customs<br>+ Soft skills | eAMGs system gaps<br>eAMLs system levels |
| | Team | | + Human agents<br>+ Cultural constraints | | |
| | Machine | | + Artificial intelligences | | |

*Table 34. eSARD evolutive 3C working environment set-up and characteristics.*

Regarding the workflow framework there are several relevant metrics to be considered, such as:

- **Time** (design efficiency). This addresses the efficacy of the design process. For instance, how long does it take for different workflow frameworks (evolutive or not) to achieve the same system maturity?
- **Number of agents**. This approach potentially reduces the number of agents involved by increasing overall design efficiency. However, the number of agents required for the same results and schedule is a metric to be considered.
- **Results**. This is a complex and yet critical metric that can have multiple perspectives. For instance, how does the quality of the design efforts compare among different framework and methods? Does it meet all requirements?

### 6.4.2. eSARD Workflow

Once multiple tools have been selected and the workflow framework has been set-up, the next step is to establish a design strategy following the eSARD method. Once more, this process is not rigid, and presents a networked nature so there is not a fixed starting point. The goal is to assess, study, fill, connect, and reassess all parts within a given eSARD diagram that will keep improving details and broadness with every design cycle. The subsequent design network is made by all those models, parameters, sketches, eAMGs, eASGs, and links used among them with a specific concept structure.

Figure 177 presents the general eSARD helix diagram considering all three DOI sectors (system design, implementation, and operations). The full process must consider all sectors and include optimization efforts as part of the evolutive methodology. However, in this study case only the first part of the system design sector will be presented. This includes key steps towards milestones 1, 2, and 3. Table 35 presents and elaborates key steps, milestones, and connections within the system design effort that was developed within this study case for an evolutive portable habitat architecture and all related subsystems.

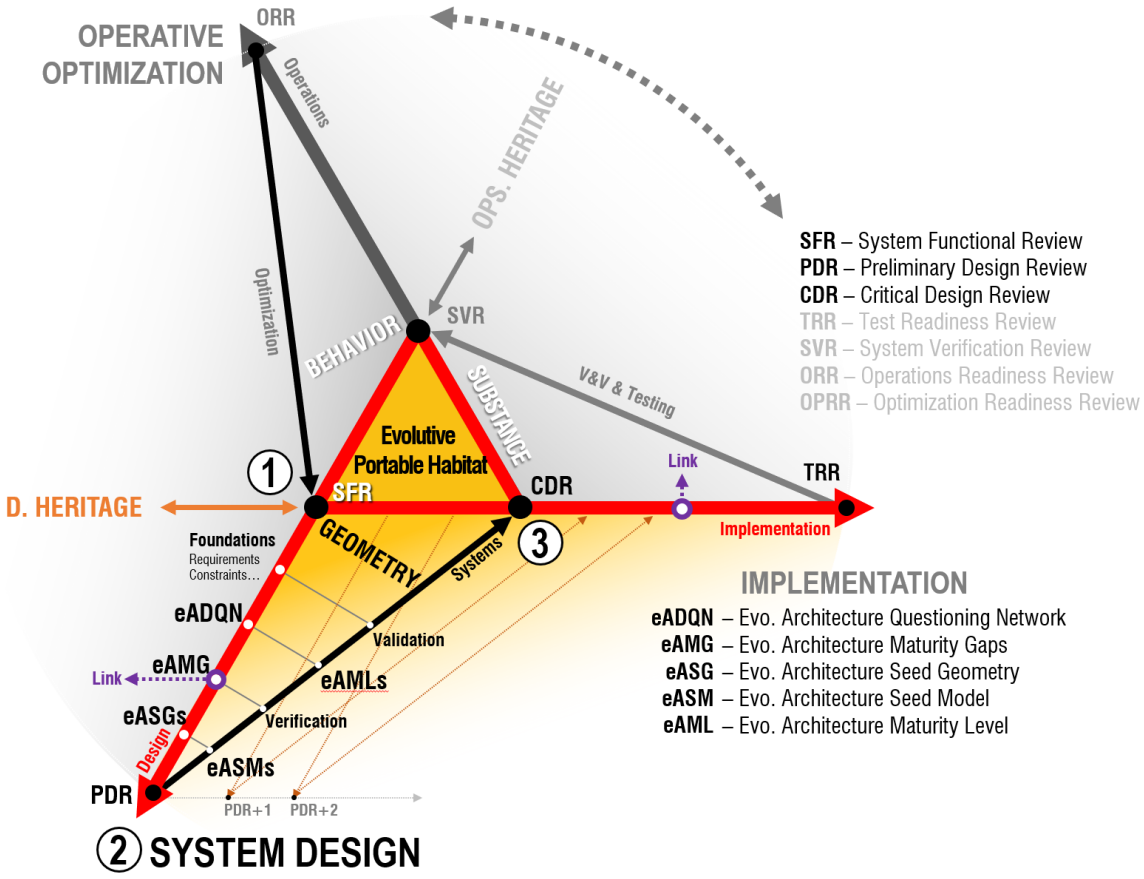



*Figure 177. General eSARD diagram showing an emphasis in the system design sector that was developed for this study case.*

| Milestone | Description / Objectives | Key Tools | Actions / Results | | | Links |
|---|---|---|---|---|---|---|
| | | | *Design* | *Operations* | *Implementation* | |
| **Effort** | 1. Assignment assessment<br>2. Evaluation of primary requirements<br>3. Heritage concepts, systems, and technologies. | Client interviews<br>Effort assessment | | | | SFR<br>DOI |
| **1. SFR** | 1. eSARD starting point<br>2. Secondary requirements<br>3. Constraints<br>4. Resources<br>4. eADQN | Data collection<br>Client interviews<br>Research<br>eADQNs<br>eAMGs | 1. eAMGs (Synergetic GBS gaps)<br>2. Level of GBS<br>3. DOI details | *Overall study* | *Overall study* | PDR<br>DOI |
| **2. PDR** | Full concept design | eADQNs<br>eAMGs<br>eASGs<br>Basic analysis | 1. eASGs<br>2. DOI gaps.<br>3. Basic geometry<br>4. Basic system def. | *1. Operational constraints*<br>*2. System behavior basics* | *1. Implementation Constraints*<br>*2. System substance basics* | CDR<br>DOI |
| **3. CDR** | Full detailed design | eAMSs<br>eAMLs<br>Detailed analysis<br>V&V | 1. GBS details<br>2. Figures of merit<br>3. Detailed geometry<br>4. Detail Sys. Def.<br>3. Parametrics | *1. System behavior details and design influences*<br>*2. Operative path* | *1. System substance details and design influences*<br>*2. Operative path* | TRR<br>DOI |
| *TRR* | *Full testing approach* | | | | | *SVR* |

*Table 35. Phases, tools, and connections within the eSARD design sector for the EPH study case.*

### 6.4.3. Requirements, DOI, GBS, and General Design Principles

In this study case there are several initial aspects to be captured, defined, and assessed as final steps to set up an eSARD process. These include at least the following topics presented in Table 36.

| | | ID / Description | Goals (general) | Objectives (details) | *Links* |
|---|---|---|---|---|---|
| **Design Objectives** | | Effort | Feasible EPH architecture | Identify key subsystems<br>Subsystems design path | |
| **Primary Requirements** | | R1 Dwelling volume | R1.1 High expansion<br>R1.2 Permanent<br>R1.3 Temporary | 1- 40 $m^2$ | |
| | | R2 Transportability | R2.1 Manual deployment<br>R2.2 Automated deploy.<br>R2.3 Easy relocation | Compatible with standard shipping system (2.2 x 2.8 x 6 m) | |
| | | R3 Mass reduction | R3.1 Low structural mass<br>R3.2 Low envelope mass | Low density solution (6000 kg max.) | |
| | | R4 Thermal insulation | R4.1 Low-high temp. range | Sustainable materials | |
| | | R5 Structural integrity | R5.1 Static loads<br>R5.2 Dynamic loads | Minimum # parts | |
| | | R6 Energy | R6.1 Minimize surface | | |
| | | R7 Style | R7.1 High customization | Color, texture, size, details | |
| **2nd Requirements** | | TBD based on eADQNs | | | |
| **Heritage** | | Table 33 (references)<br>+ Mobile habitats<br>+ Trailers<br>+ Advanced tents | Assess previous solution | Compare baseline design | |
| **Evolutive Areas** | | | | *Figures of Merit* | *Links* |
| **ARR** | **Adaptability** | A1 All-weather cond.<br>A2 Multiple terrain cond. | | *TBD* | R1, R2, R3, R5, R6, R7 |
| | **Reactivity** | R1 Environment react.<br>R2 User reaction | | *TBD* | R5 |
| | **Regeneration** | Re1 Solar energy | Re1.1 Maximize surface | *Solar surface* | R6, R4 |
| **DOI** | **Design** | D1 Multiple application<br>D2 Easy customization | D1.1 Modularity<br>D2.1 3D printing | *Combinations* | |
| | **Operations** | O1 Easy manual operation<br>O2 Environmental reaction | O1.1 Minimize actuators<br>O2.1 Minimize sensor | *Easiness* | |
| | **Implementation** | I1 Recyclable materials<br>I2 Customize | I1.1 Adv. manufacturing | *Mass* | |
| **GBS** | **Geometry** | *TBD* | TBD | *Compactability* | |
| | **Behavior** | *TBD* | Heritage actuators (if any) | *Manual ops* | |
| | **Substance** | *TBD* | TBD | *Env. Upgrade* | |
| **Evo Design Principles** | | Better with less | *Multifunctionality* | | |
| | | More with less | *Minimum number of elements* | | |
| | | Smarter with less | *Seamless operations with less elements* | | |

*Table 36. Key requirements and parameters defining the Design DOI sector activity.*

## 6.5. Evolutive Architecture Dynamic Question Network (1D, eADQNs)

Once design objectives, initial requirement evaluations, and heritage assessments (if any) have been studied the next step in the eSARD design process is to question the design concept itself to find synergetic gaps that will lead all subsequent design efforts. Chapter 4 and 5 already presented details about eSAR and eSARD methods. This first one-dimensional step is very critical and begins with series of questions (Table 36). Next paragraph (Table 37) and Figure 179 summarize and weight in multiple questions simultaneously. Also Figure 179 graphically summarizes the eADQN process.

The process starts by studying the summary of design requirements and ARR needs (Table 37) from a DOI standpoint. This is led by the chief architect to explore the architecture space. DOI questions lead to address GBS details, which are more specific and often present multiple connections and similarities among them. Then, this leads to start evaluating figures of merits behind each branch. These synergies are the foundation as well for the maturation space in which the eSARD approach operates. Then DOI, GBS, and key figures of merit are evaluated considering its relevance toward the feasibility of the system, synergies across disciplines, and all connections being tackled (Table 37). This networked process keeps increasing the number of questions, connections, and evaluations until the most basic ones are identified. This could be done regardless the number of people on the team or the type of team. Among many questions based on requirements for this study there some of the most relevant:

**DOI – Design (Geometry)**

- How can the habitat be easily deployed in any condition and terrain?
- How does the design minimize the number of components while increasing its overall adaptability?
- How its design enables an easy customization?
- What is the most compactable architecture configuration?

**DOI – Implementation (Substance)**

- What is the easiest customization scheme?
- How does the concept ensure system recyclability for this habitat design?
- How can the concept reduce manufacturing steps?
- How is the need of construction materials being reduced?

**DOI – Operations (Behavior)**

- How does the system achieve adaptable environmental protection?
- How have the number of actuators, signal, and sensors been minimized?
- What is the strategy to ensure the integration of both manual and automated operations?

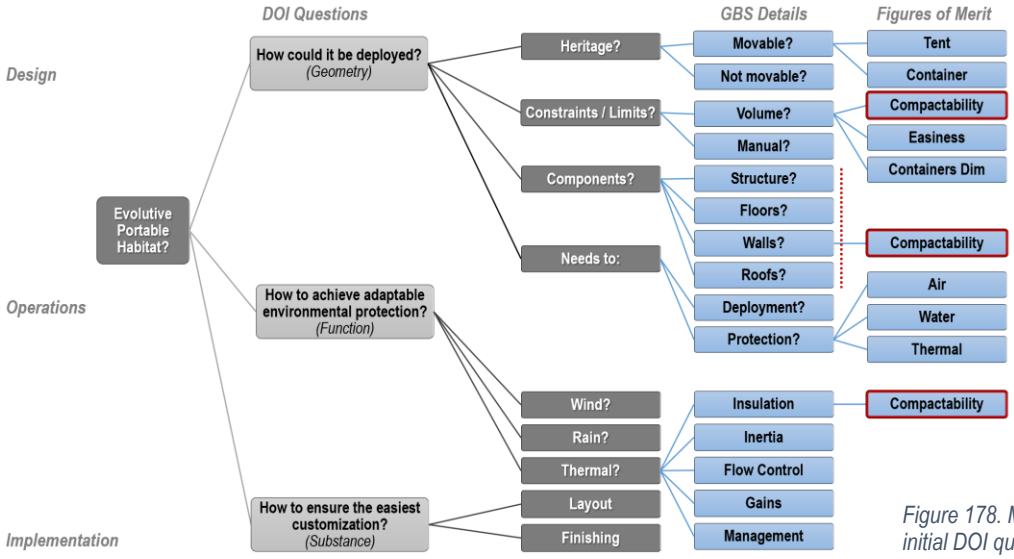
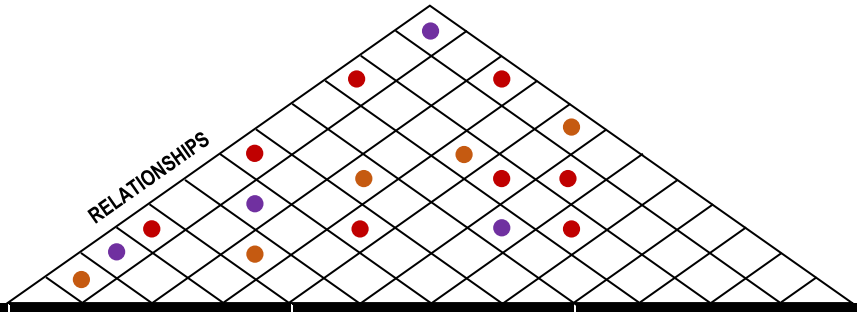



*Figure 178. Mind map scheme of some initial DOI questions, and GBS details.*




| | | | GEOMETRY | | | | BEHAVIOR | | | | SUBSTANCE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *High Med Low* | **DOI GAPS** | **System Importance** | **Deployment easiness** | Component reduction | Custom design | **Compactable Arch.** | Env. Adaptability | Minimize # actuators | **Manual operations** | Sensor-driven ops. | **Customization** | Recyclability | Mfg. Steps reduction | Material reduction |
| **Selection** | **Synergistic** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | **Complementary** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | **Dependencies** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | **Direct causality** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | **Multiple areas** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | **Many levels** | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| ***Refined*** | ***Relative Cost*** | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | ***Availability*** | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | ***Applicability*** | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | ***Feasibility*** | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | *Partial Total* | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Raking** | **BSG Details** | | 8 | 7 | 3 | 10 | 3 | 1 | 8 | 5 | 5 | 8 | 6 | 1 |
| *1 Low \| 10 High* | **Requirements** | | 2 | 3 | 5 | 10 | 1 | 4 | 8 | 2 | 9 | 4 | 3 | 1 |
| | **Constraints** | | 1 | 4 | 3 | 7 | 5 | 6 | 10 | 3 | 7 | 3 | 5 | 1 |
| | **Disciplines** | | 3 | 8 | 2 | 8 | 3 | 1 | 5 | 1 | 9 | 1 | 4 | 3 |
| | **Heritage** | | 2 | 2 | 1 | 2 | 1 | 4 | 2 | 6 | 7 | 3 | 6 | 2 |
| | **Sys. Closeness** | | 5 | 2 | 2 | 10 | 4 | 1 | 8 | 3 | 3 | 4 | 3 | 5 |
| | **Programmatic** | | 4 | 2 | 3 | 5 | 5 | 1 | 5 | 5 | 10 | 1 | 2 | 7 |
| | **Unknowns** | | 1 | 4 | 1 | 7 | 1 | 1 | 6 | 1 | 2 | 3 | 5 | 3 |
| | *Weighted Importance* | | 26 | 32 | 20 | **59** | 23 | 19 | **52** | 26 | **52** | 27 | 34 | 23 |
| | *Relative Importance* | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |

*Table 37. eADQNs and eAMGs identifcation for the EPH study case.*

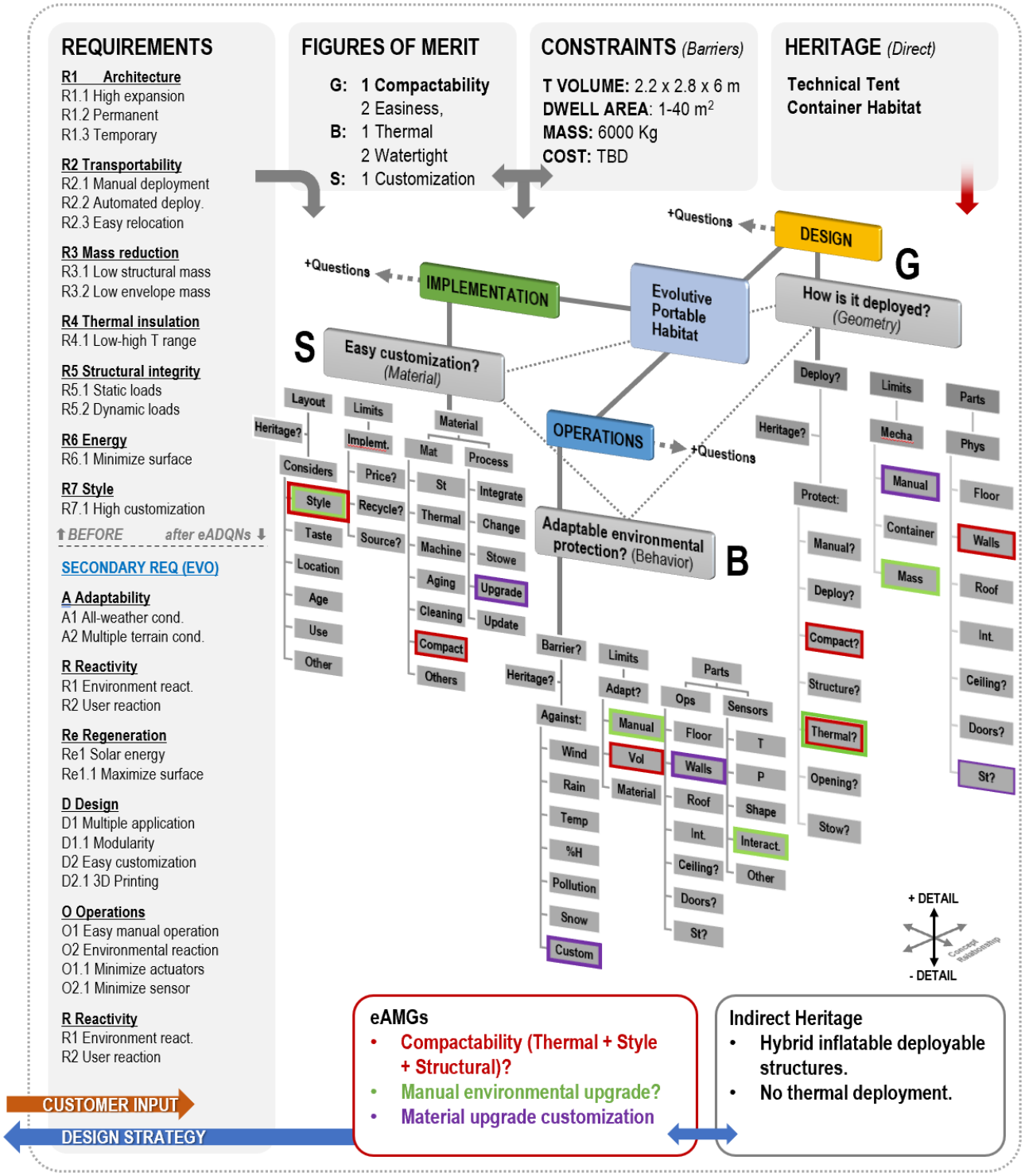



*Figure 179. eSARD summary of eADQNs for the EPH development study case.*

### 6.5.1. Conclusion

After developing the eADQN process, several conclusions arose influencing the rest of design process:

- **Secondary requirements**. The study of initial requirements, and the exploration of both architecture space (potential design path) and maturation space (key gaps affecting system feasibility) enabled to discover certain new gaps that are critical to fulfill key design objectives as these points summarize:
  - Following ARR characteristics, and DOI areas several secondary requirements were identified (Figure 179).
  - The compactability of the system is critical and related to all DOI areas, and GBS details such as:
    - Deployment capability is conditioned by the system compactability.
    - Thermal and environmental protection also need to be compactable and compatible.
    - There is no relevant heritage regarding thermal and environmental protection compactability.
  - The possibility of manual upgrades for environmental protection elements is also critical.
  - Material customization also needs material upgrades and updates based on previous points.
- **Evaluations of eADQNs**. Table 37 presents some questions elaborated during eSARD sessions and a study of their relevance. Four questions were identified as the most relevant, with the first one being the most critical:
  - How could we achieve a fully compactable architecture (addressing multiple GBS details and disciplines)?
  - How does the design achieve system deployment easiness?
  - How are manual operations enabled?
  - How multiple levels of customization are achieved for new systems?
- From these key questions three potential eAMGs are identified and elaborated on section 6.6.
- This facilitated process can be done multiple times to improve and refine these findings. The deeper and more synergetic those gaps are, the easier and more efficient the design process will be.

### 6.5.2. Metrics

There are also multiple metrics by which to assess the eSARD process at this point, such as:

- **Agility**. By diving into these synergetic connections and gaps the design effort tackles not only key critical areas but also those capable of affecting and conditioning multiple disciplinary studies that are required to mature systems. For instance, by addressing the compactability of the design from a thermal, architectural, customization, and structural perspective the design process tackles simultaneously multiple gaps and requirements affecting those disciplines. The number of topics and disciplines per unit of time, and per design architecture is higher this way.
- **Adaptability**. This approach enables to assess what key topics are allowing more system variations as well as the influence of heritage solutions upon them. This way more variations per time and design are achieved.
- **Depth**. This approach allows to pre-consider GBS details enabling the process to identify showstoppers issues faster.

## 6.6. Evolutive Architecture Maturity Gaps (eAMGs)

Among all design questions without answer, one becomes the most critical as Figure 179 and Table 37 show: how the system can achieve a fully compactable architecture that allows both manual handling and easy customization? This is the initial eAMG affecting all DOI areas and the initial challenge in which the design activity will concentrate upon. This means that all the following points are being tackled simultaneously:

- **Multiple disciplines** are considered simultaneously, such as architecture, mechanical, structural, thermal, etc.
- **Geometry.** The use, deployment, and environmental protection of the design depends on key geometrical aspects.
- **Behavior** also conditions changes, upgrades, and operations manually or otherwise.
- **Substance.** This includes system integration, material selection, material availability, relative cost, and workforce.
- **Heritage.** Inflatable hybrid deployable habitats showcase a potential heritage technology to be used in this design.

This process will require multiple iterations (SFR to CDR), questions, and connections that must be reassessed after each design cycle. Initial eAMGs are the most efficient and multidimensional (2D) starting points for the design process.

## 6.7. Evolutive Architecture Seed Geometries (eASGs)

The next step in the eSARD process is to facilitate a design activity or tool that could be done between one or more people. The goal is to use eAMGs as an objective, addressing all multiple GBS aspects while keeping an eye on other DOI sectors that could be developed simultaneously.

In this example, the study of a deployment approach for the structure and material solution providing the environmental protection (including thermal and other weather elements) is key for the feasibility of the full system. Thus, both storage and deployment states need to be addressed considering operation easiness and availability of all materials being used. The exploration of multiple design paths within the architecture and maturation space is summarized very briefly in Figure 180. As explained in previous chapters colors, views, and details relate to multiple design constraints. Due to the scope and limitations of this research only some sketches that were made for the process are shown. It is worth mentioning, that considering overall DOI perspectives and BGS details support the agility of process while eAMGs have a ripple effect across disciplines. In this study there are two major design paths being explored (Figure 180).

Once a seed sketch is created, its geometry (DOI design sector) allows the mechanical discipline to assess the challenges of mechanism design, the thermal discipline to evaluate surfaces and cross sections properties, while an overall architecture approach thinks about customization, user experience, and operation easiness. In essence these initial geometries for the system provide a common framework across design perspectives, as well as a reference towards operations and implementation standpoints. That synergetic thinking is already a seed in the eAMG and must be maintained and expanded during the use of this tool. There are three major related activities within this step:

- **Evolutive sketch** (by hand). Following section 5.10 and as presented in Figure 180, these sketches represent geometric, operational, and implementation aspects to both explore and create a design paths or strategies. Thus, it is not a traditional sketch since starting with ARR characteristics it addresses many DOI aspects, and it is used to organize the subsequent development activity. For instance, it allows the mechanical team to start studying actuators constraints, while it provides a reference for the thermal effort to balance opposed design forces.
- **Seed architecture** (CAD). Hand or computer sketches created by a chief architect (in a group or alone) will keep increasing details as in any other design process. Hence, these will keep addressing more and more GBS details that drive design specifics within the system being developed. Beyond hand sketches, this approach evolves into using fast computer geometrical models that are either created my humans (CAD) or by machines (generative). These models not only provide initial dimensions, but also initial geometries for all disciplines to start basic analysis as well. This is shown in Figure 180 as CAD plus sketches at the end of each design path.

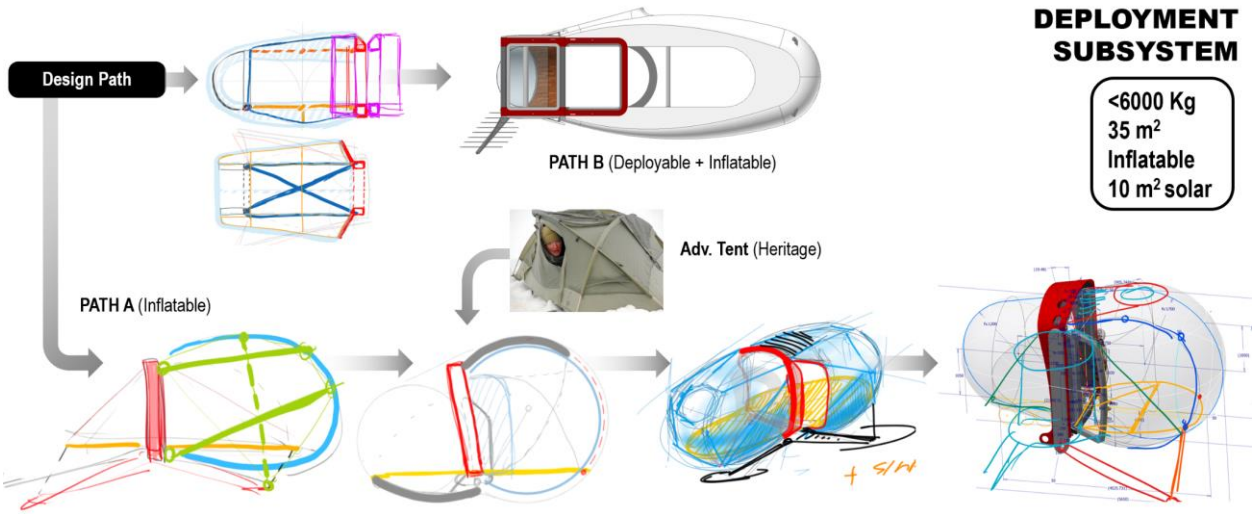



*Figure 180.Design paths based on eAMGs for a EPH deployment subsystem. © 2020 Raul Polit Casillas (patent pending).*

- **System definition** (spreadsheet). These initial but increasingly detailed perspectives not only allow architects and engineers to assess other aspects of other solutions early in the process (e.g., implementation and operations), but they also enable the initial capture of key figures of merit within an eGBSEL (section 5.11.2). Figure 181 shows a capture of an initial eGBSEL in this example. These are a foundation for system models (eASM) being used to fully capture the system and enable an analysis of other design alternatives towards a CDR level. This enables enough detail and a full exploration of the maturation space. All parameters used here are linked with geometrical models.

### 6.7.1. Heritage

At each step or level in the evolutive approach, the assessment of heritage allows new options, new technologies, and validated elements to be infused into the system design. In this case, heritage is used in two ways:

- **Initial assessment**. The evolution of heritage solutions such as tents, container habitats, and trailers, provides a reference framework by which to compare against requirements and new solutions. However, these are just inputs, and by any means they could restrict new evolutive system solutions. Also, this is used to find inspiration in other fields such as protective gear, inflatables, origami structures, etc.
- **Deployment subsystem**. Regarding system architecture components the use of heritage brings the capability to compare solutions from a performance standpoint, as well as to evaluate others details such as cost, integration, manufacturing, etc. Always from a non-restrictive approach, this also allows to bring new technologies or solutions on board (as heritage or validation) and to provide a reference framework to assess other DOI details.

### 6.7.2. Metrics

This approach allows to achieve more reliable designs in less time. While this chapter only presents a summary of such process, this method reduces the overall time to CAD in almost a half when compared with other more traditional and sequential efforts. It also allowed to have in less than two months a full system (not shown) addressing many DOI areas towards a PDR level review using only one person, as opposed to a regular team of four. The end result is a more detailed eASG after two cycles (Figure 180). This method makes the most of every minute and every bit of information by keeping the tension between a truly broad perspective and multiple system details. In terms of metrics the following points offer a summary:

- More design topics are tackled simultaneously.
- An increased number of disciplines are addressed simultaneously.
- Less people and agents are needed in the process to achieve a similar level of maturation.
- Greater number of design alternatives are considered, and more innovative solutions are obtained.
- Better system and subsystems definition occur faster due to the eSARD synergetic approach.
- Less time and computer power are required since the effort is potentially faster and uses less agents.

## 6.8. Evolutive Architecture System Models (eASMs)

Following the identification of eAMGs and the creation of eASGs, the modeling of the system is the next step. Always with a networked perspective that interconnects all models, the creation of eASMs (Figure 181) serves many purposes, such as parametric studies, evaluation of solutions, technical budgets, tracking of progress, etc. eASMs present a series of key

| IN | System / Subject | # | DESIGN | | | | | | | | | | | | OPERATIONS | | | | | | | | | | | | OUT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Geometry | | | | Behavior | | | | Substance | | | | Geometry | | | | Behavior | | | | Substance | | | | |
| | *Subsystem / Component* | | *Param.* | *Value* | *Unit* | *Reinf.* | *P* | *V* | *U* | *R* | *P* | *V* | *U* | *R* | *P* | *V* | *U* | *R* | *P* | *V* | *U* | *R* | *P* | *V* | *U* | *R* | |
| | **A1 Fabric Deployment** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | *A1-1 Ext Fabric* | 1 | A1-1G | 10 | m2 | | A1-1C | 10 | Colors | web | A1-1S | 300 | Kg | www | | | | | | | | | | | | | |
| | *A1-2 Plastic Membrane* | 2 | | | | | | | | | A1-1S | 35 | Kg | www | | | | | | | | | | | | | |
| A1 | **A2 Structure** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A1-1 | A2-1 Deplo. Floor | 2 | A2-1G | 1 | m3 | | | | | | A2-1S | 300 | Kg | www | A2-1-OG | 2 | Act. | | A2-1-OB | 6 | Mov. | www | A2-1-OS | 100 | W | Cell | |
| | A2-2 Deplo. Beams | 4 | A2-2G | 0.2 | m3 | | | | | | A2-1S | 35 | Kg | www | A2-2-OG | 3 | Act. | | A2-2-OB | 9 | Mov. | www | A2-2-OS | 25 | W | www | |

*Figure 181. Portion of the initial eGBSEL for the EPH design (first design cycle).*

characteristic that makes them different to others SE tools. Among some pf the most relevant are the following:

- **GBS details / DOI areas**. On an eGBSEL each parameter is addressed from the double perspective of DOI processes and GBS system details. This enables much deeper system definitions, studies, evaluations, and parameters.
- **Foundational**. The previous point enables to create multiple studies with less measurements and parameters, such as mass lists, power assessments, etc. The eGBSEL becomes a hub to study, capture, and assess the system architecture design holistically.
- **Links**. The eSARD approach is networked, which allows to look at the eGBSEL from a hyperlink-driven approach. Each parameter is both an analytical value and a conceptual link among other parameters.

## 6.9. Next Steps and Phases

To achieve a full CDR level, these steps need to be iterated several times until all figures of merit validate the system design or the initial requirements are met with sufficient margin. The identification of secondary requirements as part of the process could be used as an imperative to change design path or simply as an advice, however they enable the final system feasibility. Therefore, further iterative processes must happen to increase definition, while at the same time other steps (sectors) need to be considered. These activities within a networked approach include the following among others:

- **Analysis**. On this phase basic analysis are required to study and assess solutions. In this case, geometrical modeling of inflatables, structural analysis, and thermal models are created to name a few.
- **Validation**. In parallel to this activity, the second edge of the DOI design sector addresses how to validate such analysis and models. Thus, critical models must be created to study, scale, and tweak subsequent designs.
- **Testing**. Similarly, during the DOI design phase a testing concept plan needs to be created and started.
- **Operations** (DOI). Independently from the DOI operative sector, it is important to consider and introduce key operational aspects within each phase. Here, this includes manual operation details, user experience, and actuators.
- **Implementation** (DOI). This happens as well with implementation aspects such as integration, material selection, and manufacturing. These are included in all design considerations early in the process, and while they will also be fully developed in detail within the implementation sector. Both basics topics and links are integrated in this phase.
- **Optimization**. Each parameter can be used as part of an optimization strategy within a hyperconnected approach such as this. However and most importantly, any design strategy or path must consider where such optimization presents enough design margins and how to use them afterwards for new system variations or even new efforts.

## 6.10. Conclusion

The example selected for this study showed the need for many of the key characteristics that define an evolutive system architecture (eSAR). This was done after a first look through the perspective of an eSARD approach. From the beginning key ARR system areas (adaptability, reactivity, and regeneration) were initially addressed since they required design and analysis for a successful system solution. However, such approach also highlighted the need to assess other direct and indirect heritage solutions, as well as the benefits of exploring all initial requirements more in depth.

Thus, the first step in this example was to set up a 3C evolutive environment, addressing collaborative, communicative, and concurrent subjects that included tools, agents, and preliminary connections, among other models. This not only set up the tone for the design process workflow, but it also speeded up all subsequent work activities. So, both tools and the environment in which they operate required some design beyond the system solution itself, and this made a big difference. This is a relevant aspect of the eSARD approach that differs from other methods. The analysis of heritage solutions from a systems, component, and technology level perspective, created a very useful pool of information to be used later on during any design activity across all multiple phases and sectors of the process.

The eSARD approach (section 6.4.2) addressed the design activity in this example from a combined perspective involving all DOI areas (design, operations, and implementation). While this example only tackled the first one, Table 35 presented the full workflow of the activity. Once the environment and heritage studies were studied, the eSARD process presented several phases providing key conclusions that are summarized in the following points:

- **Requirements, DOI and GBS principles** (section 6.4.3). The study of these topics under the eSARD approach brings a very holistic and interconnected perspective to assess a design challenge (Table 36). This led to the creation of secondary requirements evolving from this study, which were not initially provided by the customer but became critical to find a feasible solution. The solution been considered here is not just for one single EPH, but rather a family of EPHs considering possible future upgrades and changes. Applying such design stressor made possible to simplify solutions and discover design paths that will reduce manufacturing step, costs, and mass later.
- **Dynamic questioning techniques using eADQNs** (section 6.5) allow to find the most critical DOI question conditioning the system feasibility and the design workflow in general (Figure 178). In this case system compactability, manual operations, and possible environmental upgrades turned out to be the most critical aspects for this endeavor. These tools allowed to assess and reinforce how critical was the adaptability of the system for its success.
- **Maturity gaps (eAMG)**. The use of eAMG techniques (section 6.6) allowed to identify the most critical gap for this design, which is also the most relevant topic integrating key disciplines, such as mechanical engineering (mechanisms), thermal management, and user experience topics conditioning style and interior design. Thus, the compactability of the habitat geometry was a key maturation gap (Table 37) and more important than mass reduction or cost. Without answering this gap, the solution is not feasible, but at the same time solving this problem from such multidisciplinary standpoint automatically allows to have three major disciplines tackled. So, one synergetic topic integrates multiple views, so focusing on it allows to find a better and more well-rounded solutions much faster.
- **Evolutive seed geometries (eASG)**. Such gap in the design effort (Section 0) enabled to form a concept geometry (Figure 180) addressing multiple design paths and disciplinary questions simultaneously. These geometries considered mechanical topics, thermal performance, manufacturing techniques, SE topics, and style, among others.
- **Evolutive system models (eASM)**. Finally, the last step within the DOI design sector is the creation of an initial system model that captures quantifiable and qualifiable parameters, while creating links to eASG models (section 6.8). In this case, the tool selected to capture the system was an evolutive eGBSEL. This spreadsheet-based model is reinforced with hyperlinks and covers all ARR and GBS system development areas, allowing to create a much broader and detailed view of the eSAR at hand. This will be the foundation for subsequent parametric studies.

In essence, the use of an eSARD methodology allowed to find a solution very efficiently in terms of schedule (time), tools, and agents (section 6.7.2). However, the key is that this approach allowed to better tackle the design activity from a conceptual, logical, practical, and programmatic standpoint. While next steps and phases (section 6.9) will certainly bring the full power of this approach to other DOI areas such as implementation and operations, all the enhancements and efficiencies provided by this technique are showcased by both product and process.

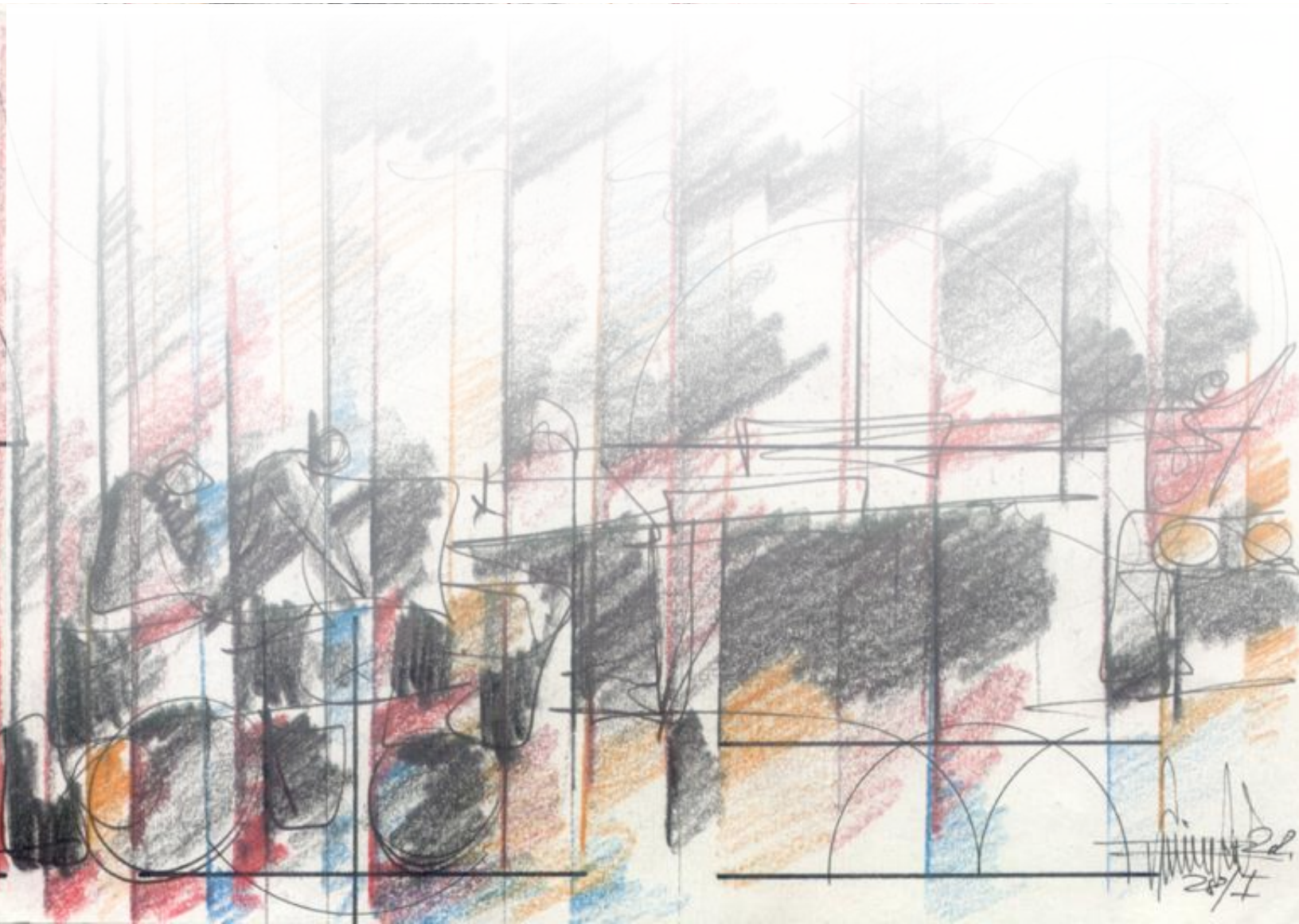
# EVOLUTIVE PATHS

## Discussion and Final Conclusion

## CHAPTER 7

*"A good question is never answered. It is not a bolt to be tightened into place but a seed to be planted and to bear more seed toward the hope of greening the landscape of idea."*

***John Ciardi***



# 7. Conclusion: Evolutive Architecture System Design Paths

To conclude this thesis and summarize the research activity that was conducted there are several key points elaborated across the following sections. They finalize this dissertation creating a starting point towards new developments.

## 7.1. Discussion

The starting point for this research were complex hardware-based systems (CHS) in general, and a subset of these defined as evolutive system architectures (eSAR). Furthermore, design and system engineering methodologies enabling such systems were part of the research goals. Previous chapters elaborated the path used along this dissertation to address all three initial research questions. Answers and discussions to these questions are summarized in the following points.

- **What new characteristics and complementary design needs do these ultra-complex systems present within resource-scarce environments?**
  - A holistic analysis of world trends in chapter 2 showed that several design stressors are influencing the practice of SE and DE to develop complex systems (CHS). These affect the basic balance between systems needs and available resources (Figure 6) and keep forcing the need for adaptability in any robust approach that tackles them. Thus, new situations require new approaches. Some of the most relevant stressors are the following.
    - **Resource scarcity** such as energy, materials, and even workforce are driven by climate, economy, and competitiveness among other reasons, forcing systems to become more efficient and use less resources.
    - **System complexity** is increasing, especially when hardware systems (CHS) keep being enhanced with software, data, and other interactive capabilities (e.g., robotics applied to consumer products).
    - Better **system performance** needs to be achieved at faster speeds due to market constraints.
    - Increasing **multidisciplinarity** of system designs and their management must be considered nowadays.
    - **Process agility** of the system and the capability of the work effort to be reused are critical towards cost.
    - Complex systems (CHS) are increasingly more **networked** presenting more links among subsystems and their own contextual environment regardless of being physical, digital, virtual, or any combination of these.
    - **Technical design heritage** gradually influences the balance between new solutions and the risk posture.
    - **Innovation** in new systems keeps evolving towards disruption rather than partial or incremental changes.
    - **Cultural disruption,** new tools, and workflows affect design activities at both product and process levels**.**
  - Within such changing context, **evolutive system architectures** are identified as a subset of hardware-based systems (CHS) in response to current and future design stressors (chapter 4). These highly complex and field-independent systems are driven by previous stressors and new design needs. They have three major keystone characteristics or ARR, which also become the foundation towards a subsequent design methodology.
    - **Adaptability**. Any system architecture is conceived as a continuous evolutionary process where both geometrical and analytical system definitions (quantifiable or otherwise) keep changing and evolving. Any previous solution (heritage) could become a validated building block. Overall, the system geometry is always completed by functional system capabilities and implementation schemes addressing the use of resources.
    - **Reactivity**. An evolutive system is not passive and interacts with its environmental context and its components. Thus, besides the system geometry its functional and logical descriptions are critical for the behavioral completeness of the system (reactions), as it is towards its adaptability and associated processes.
    - **Regeneration**. The use, recycling, repurpose, and regeneration of all resources used by the system and its design process are also part of its development process pushing the limits of its sustainability.
  - New and changing situations define new needs for upcoming systems and methods. Complex systems require more and more the combination of both design engineering and systems engineering practices, which presents an answer to this question under the light of ARR evolutive principles. This is also a beginning for others research questions, especially when considering hardware and software interactions, and their virtual combinations.

- **What principles could enhance traditionally sequential design and systems engineering workflows to achieve faster, better, and more efficiently such multidisciplinary complex systems?**
  - A thorough literature review presented key relevant and linked gaps within **DE and SE methods** towards developing CHS when studied from a geometrical and analytical standpoint (chapter 3), such as:
    - **Synergy**. DSE methods tend to tackle multiple disciplines sequentially or in parallel.
    - **Continuity**. DSE methods are driven by point-design or discreet solutions rather than families of solutions (species).
    - **Qualification** of parameters is widely distributed among them but qualifiable aspects are challenging or impossible.
    - **Geometry**. SE methodologies do not tackle geometrical information well, and SE processes struggle with complex system definitions that could be customized or enhanced.
    - Truly **full system lifecycle** is often missing across methods.
    - **Flexibility** in DSE methods if often very complex or missing. Tools tend to re-work solutions often due to cultural reason.
    - **Disruption** is often not fostered by the method itself.
    - **Fast-paced** workflows are often challenging across efforts.
    - System **connectivity** links tend to be missing in DSE efforts.

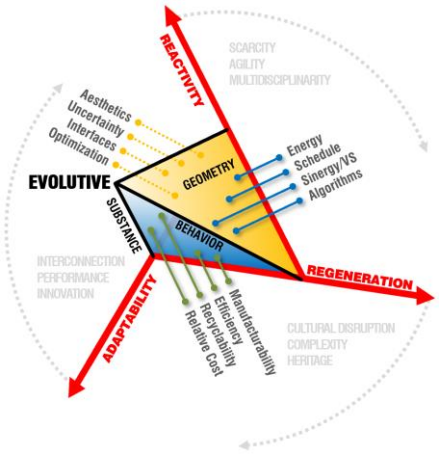



*Figure 182. Evolutive design tetrahedron.*

  - Some of these gaps have been addressed by **evolutionary techniques** (section 3.3) since the early 1950s. These were inspired by nature and mainly applied to computer science instead of CHS and other hardware-driven applications. Among key principles and contributions (section 4.1.1) several can be highlighted since evolutionary solutions are [1] continuous (species), [2] multidimensional and multidisciplinary. Their design approach is [3] agile, [4] evolvable, [5] networked, [6] heritage-driven (genetics), [7] environment-driven (co-evolution), and [8] they address the full development process (eco-evo-devo) of the system (organism).
  - Thus, the **evolutive approach unites adaptive and evolutionary principles** (chapter 4). This fill and creates synergies among DE and SE identified gaps through axioms and principles that address ARR characteristics.
  - The characteristics of **evolutive system architectures (eSARs)** were addressed in chapter 4. The associated design process developed or **eSARD** (Chapter 5) directly tackled this question. This was summarized with the **evolutive design tetrahedron** (Figure 182) that us based upon three basic system levels (GBS) such as:
    - **Geometry** (adaptability) includes geometrical aspects, aesthetics, uncertainty, interfaces, and optimization.
    - **Behavior** (reactivity) considers functional aspects such as energy, schedule, synergy, and algorithms.
    - **Substance** (regeneration) considers implementation topics of multiple natures (physical, digital, virtual, or a combinations), as well as manufacturability, efficiency, recyclability, and relative cost topics.
  - Upon these points, the **eSARD method** enables a complete developing cycle for evolutive system architectures that is based on three network keystones or system principles such as **design, operations, and implementation (DOI)**. The goal of this approach is to create a design method that achieves more efficiently evolutive, reactive, and regenerative system architectures, handling and addressing global stressors. This is mainly based on a synergetic perspective that tackles each lifecycle phase and every design challenge from within synergies among disciplines and components, rather than the classical 'divide-and-conquer' approach. These principles handle relationships among complex design drivers faster and more efficiently (Figure 124).
  - Gaps among state-of-the-art DE and SE methodologies are intertwined and complemented by eSARD principles as an **answer to this research question**. These principles are derived from both global environmental stressors and new system needs. They expand more traditional perspectives to address all key ARR needs of an eSAR, while **opening further research opportunities** towards the application of the evolutive approach to other technical, design, and development fields.

- **How could a design method that considers previous questions be used to develop more efficiently complex systems within such environment, when there is no direct heritage and ultra-system performance is a must?**

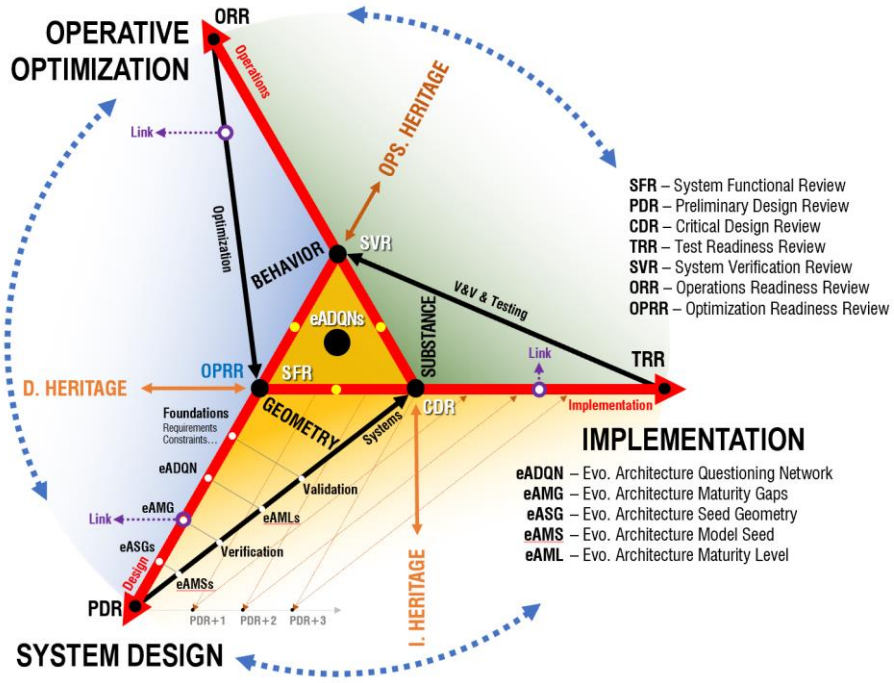



*Figure 183. Evolutive system design networked process (eSARD) addressing design, operations, and implementation sectors.*

- Developing evolutive architectures efficiently requires of a full-cycle, synergetic, and networked design method. This means a highly adaptable workflow that handles continuous designs efforts, addresses multiple links across tools and milestones, and infuses heritage inputs along the process. The **eASRD** method (Figure 183) is developed upon three universal and interconnected development areas within this thesis:
  - **Design** (geometry) addresses organizational, design, systems, esthetics, and validation aspects.
  - **Implementation** (substance) tackles from manufacturing or coding to validation, verification, and testing.
  - **Operative optimization** (behavior) includes system operations and overall system optimization.
- eSARD methods pursues **disruption** (adaptability), **smartness** (reactivity) and **efficacy** (resource regeneration) upon the system (section 5.4) and within a measurable reference framework (Figure 135). This also responds to the need of designing better, smarter, and broader, while achieving more with less (resources, parts, cost).
- Consequently, an **eSARD workflow** should be designed to increase ARR capability in the system (Figure 139) from an individual, team, machine, or hybrid perspective while making the most of current SOA methods, evolutive principles, and ARR system characteristics. Thus, such process is organized around a helix presenting:
  - Three continuous, replicable, and networked DOI sectors that tackle all development areas and tools.
  - Milestones and verification steps across the full life cycle: SFR, PDR, CDR, TRR, SVR, ORR, and OPRR.
  - Tools allowing to dynamically identify synergetic gaps (eADQNs) for a better system capability (eAMGs). These also enable a broader look at the design process (eASGs) and the system definition (eASMs). The result is faster design times, better relative heritage use, and better system performance with same metrics.
- By challenging the inquiring nature of the design process and the links across all traditional phases, this research question finds a new path that is enabled by new tools using current capabilities. This also paves the way to a more fluid but manageable approach in the design of hardware-based complex system architectures (eSAR).

## 7.2. Applications and Limitations

The scope and objective of this research was to tackle in general the development of hardware-based system architectures that could be enhanced by software, virtual, and robotic perspectives. More specifically, the design portion of the workflow was developed in detail while connections to other areas of the evolutive process have been only highlighted.

The goal of this dissertation was to address key principles regarding how evolutive system architectures should be tackled, as well as to evaluate and develop drivers, methods, and tools to develop them. This not only applies to evolutive systems but presents a foundation towards developing any complex system participating of ARR principles. From a hardware-based standpoint the eSARD approach tackles three different levels of development, as follows.

- **Architecture evolutive characteristics (ARR)**. These are high-level evolutive oriented characteristics regarding CHS design and implementation under global stressors such as the scarcity of resources.
- **System and process development phases (DOI)**. These applied to generic steps so the system can be developed, operated, and improved. They complement and connect SOA techniques and well-known milestones within current SE and DE practices across the world and among multiple industrial and technical sectors.
- **System details (GBS)**. Finally, these are universal details that need to be addressed towards implementing any complex system, but especially those with an evolutive nature, including shape, interfaces, functions, behaviors, data-structures, materials, coding needs, recyclable schemes, or decommission strategies, among many more. These are not driven by the process itself such other referenced works, but rather the implementation of the system.

This effort has not addressed in detail the implication of this approach towards other non-hardware efforts tackling portions of the evolutive approach, however it presents a feasible foundation for these due its synergetic and flexible nature. These other applications include digital and virtual systems (e.g., software), as well as hybrids (e.g., robotics).

## 7.3. Conclusion

The world is always in constant change, but perhaps nowadays more than ever due to greater climate, economical, and social reasons happening simultaneously at a global scale. The practice of complex systems design is affected by such growing new context stressors as chapter 2 introduced. Today, systems and services need an increase in their capabilities due to common stressor such as competitiveness, speed-to-market, and competitiveness, among others. However, there are also an increasing number of other global stressors with great influence over these design activities. These include among others: [1] the growing scarcity of resources, [2] the widespread need for better system performances across technical fields, [3] the influence of cultural heritage over workforce and workflows, and [4] the infusion of data-driven techniques, among many other technology disruptions with a direct influence over hardware-based complex systems (CHS).

Three research questions (section 1.2 ) bounded the initial hypothesis for this dissertation, how design mechanisms inspired by natural evolution are applied to the design and physical implementation of complex systems to enable more efficient development processes and substantiate more adaptable complex system architectures (CHS). Nevertheless, this thesis is based upon years of research, extensive literature reviews, practical experience among many technical fields, and a deep knowledge of the stat-of-the-art design engineering (DE) and systems engineering (SE) techniques.

Therefore, these design stressors emphasize upcoming changes in the ratio between systems needs and available resources (section 2.3). Furthermore, this also means that future complex systems and their associated design methodologies will have to adapt. This is especially meaningful towards the development of complex hardware-based systems due to the increasing need of multidisciplinary requirements and the influence of disruptive technologies changing how we could design, implement, and even operate such systems.

Under this light, this doctoral thesis and associated research is based on years of practice started by studying such new needs, as well as key gaps identified within current state-of-the-art design engineering and systems engineering techniques (chapter 3). The need to correlate complex geometrical definitions with more advanced and networked system descriptions highlighted that such gaps among them tend to be complementary and lack adaptability towards a new reality based on change, uncertainty, and fast-paced developments.

This thorough study of needs, gaps, and capabilities led to the development of the evolutive perspective as a hybrid standpoint between the application of adaptable principles to system products and evolutionary methods to their development processes. Evolutionary principles extracted from nature were studied (section 3.3), as well their initial applications mainly to computer science since the 50s through genetic algorithms and evolutionary programming techniques. The result is a subset of CHS with some especial characteristics and a subsequent system development method.

**Evolutive systems architectures (eSARs**) are a type of CHS in response to previous general stressors (chapter 4) and due to the application of evolutive principles to the system design. Specifically, these present three key characteristics.

- **Adaptability**, which is related to the capability of the system to change and adapt both geometry and behavior. Under this approach these systems are dynamic, and their development is continuous in nature. This system characteristic relates to both geometrical aspects of the system and the evaluation of system functions.
- **Reactivity**. These systems are smart and interact with the environment, thus their operational nature and scheme needs to be addressed from the very beginning. This principle relates to system interactions primarily.
- **Regeneration**. Finally, it is not just about the system but its implementation (physical, digital, hybrid). Hence, system substance, resources utilization, recyclability, and sustainability are considered at all levels.

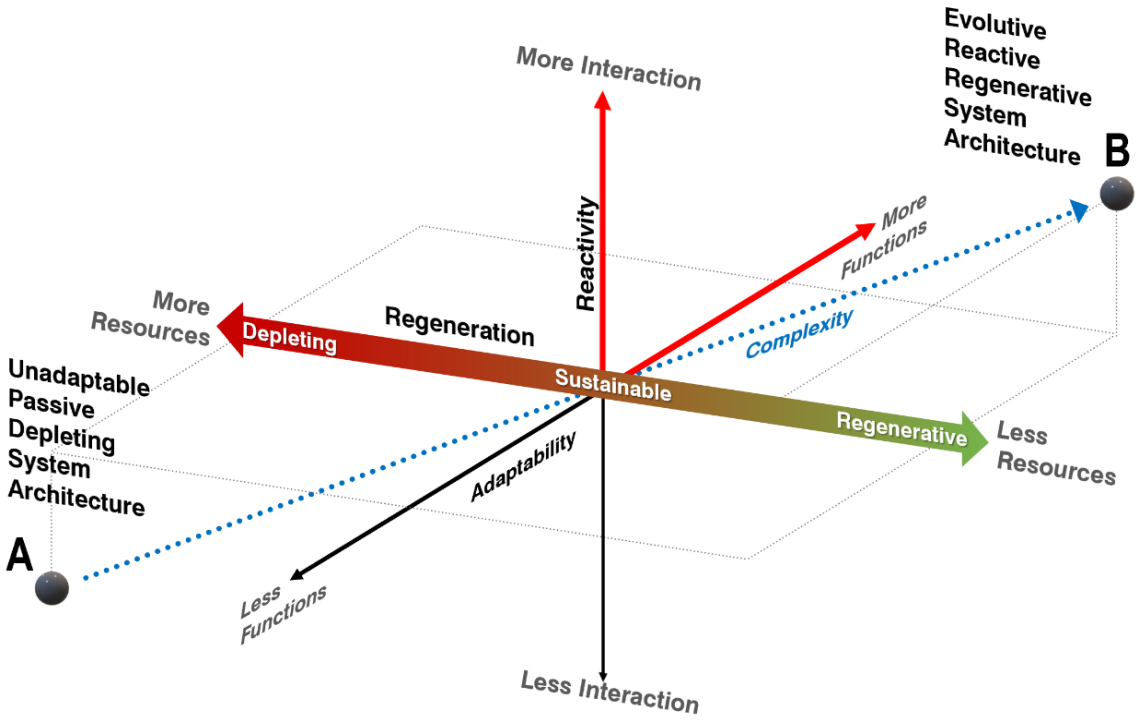



*Figure 184. Evolutive reference framework for complex evolutive system architecures (eSAR) based upon ARR principles.*

The evolutive approach towards complex systems is founded on a series of key design drivers associated to these characteristics. There are integrated within a three-dimensional design reference framework or **evolutive tetrahedron** (Figure 182) providing coordinates to define such systems. Within this space, two levels of development such as **design principles (DOI)**, and **systems details (GBS)** are interconnected from the networked standpoint of this approach.

The **eSARD method** is created in response to such unique system characteristics requiring a development approach that embraces those foundational principles. This approach is based upon [1] finding synergetic gaps and links among requirements, technologies, subsystems, and heritage solutions, among others, [2] the study of the system from multiple ARR standpoints under a networked perspective, and [3] specific design objectives and principles to develop better complex eSAR systems faster, easier, cheaper, and more efficiently. The eSARD method is multilevel, multidisciplinary, and highly adaptable in nature.

Furthermore, its design approach (chapter 5) allows to tackle many more relevant topics towards the feasibility and integrity of the system, while addressing simultaneously more disciplines. This method embraces new technologies and techniques, complementing other traditional tools and approaches that can be used by individuals, teams, machines, and hybrid workflows. The core of this approach is about finding proper questions, by intensifying both depth and range of any initial questioning. This allows to find from a truly multidisciplinary standpoint all real showstoppers for any system design. Finally, the eSARD methods brings enough flexibility to the process to quickly change, evolve, and vary the scale and level of the effort covering from high-level areas to system details towards feasible and high-performance system designs.

Hence, this process (section 5.7) enhances traditional methods with a series of evolution-inspired tools and techniques that enable a more efficient and measurable methodology. Among the most relevant tools within this method, **evolutive questioning networks (eADQNs**) described in section 5.8 allow to find critical **maturity gaps or eAMGs** (section 5.9) which become the multidisciplinary and multilevel starting point of the design process. These gaps are critical because they allow to find most synergetic design topics that condition the feasibility of the system, as well as the efficiency of the development process since their multidisciplinary nature helps tackling multiple perspectives all at once.

An eSARD workflow continuously assesses all different facets of **designing, implementing, and operating (DOI)** both systems and system families (species). This enables a very holistic yet practical approach towards any DSE activity. The beginning of the process is tackled by interconnected **evolutive seed geometries or eASGs** (section 5.10) that lead to evolutive **system models** (eASMs) that describe and define the system. Tools within these models include **evolutive system sketches** (eSD, section 0) and **GBS equipment lists** (eGBSELs, section 5.11.2) which are evolutive enhancements of classic tools that substantiate a more open, adaptable, and reliable method. Such approach is supported by connections and links among all development sectors as described in the eSARD helix diagram (section 5.6 ) and the subsequent assessment and infusion of heritage inputs.

This evolutive approach also presents a way to measure and evaluate the development of a system with the use of **architecture maturity levels (eAMLs)** as described in section 5.12. Hence, fast pace, facilitated, and curated design cycles can be managed and effectively conducted within this design framework. The eSARD approach infuses a **collaborative, concurrent, and collaborative (3C) framework** to its practice ensuring its adaptability towards new tools, capabilities, and design agents in the future. In essence, this approach presents a universal method for CHSs which is customized for eSARs.

## 7.4. Research Contributions

In summary, key research contribution in previous sections could be summarized as follows.

- **A holistic analysis of global design stressors** influencing the balance between system needs, method capabilities, and available resources across the process (chapter 2), as well as their influence in all general design efforts.
- **A thorough literature review and joint analysis of gaps and links among design engineering (DE), systems engineering (SE), and evolutionary driven techniques** addressing the efficient design of complex hardware-based systems (CHS) and eSAR. Chapter 3 presents several tables addressing those areas from these perspectives:
  - Technique, lifecycle design phases, drivers, and barriers.
  - Capability of addressing geometrical, analytical, qualitative, and quantitative information.
  - Field of study, scope, domain, type of products, inherent adaptability, and perspective.
  - Associated tools, frameworks, workflows, and platforms.
- **A novel definition and classification of evolutive system architectures (eSARs)** as a subset of generic complex hardware-based systems that are driven by adaptability, regeneration, and reactivity (chapter 4). This includes:
  - General approach and evolutive principles in relationship with global stressors.
  - Evolutive system keystones and general characteristics.
  - Evolutive drivers, interrelations, and evolutive design reference framework.
- **A new design approach and associated development method (eSARD)** in chapter 5 tackling all aspect of hardware-based system development from a multidisciplinary, full-cycle, and detail-inclusive perspective. This also

integrates design engineering and systems engineering practices with evolutionary-inspired approaches including:

- Links among state-of-the-art system design methodologies and evolutive system needs.
- **Evolutive design tetrahedron**, ARR areas, evolutive design objectives, drivers, and principles.
- **eSARD_he** helix model for system design engineering based on **DOI areas and GBS details**.
- **Evolutive design workflow and evolutive design tools** including among others the following:
    - Dynamic system and evolutive design questioning (eADQNs)
    - Synergetic design gaps (eAMGs) and derived workflows.
    - Evolutive seed geometries (eASGs) and evolutive sketches.
    - Evolutive System model (eASMs) and GBS equipment lists (eGBSEL).
- **Architecture maturity levels (eAML)**, metrics, and comparisons.

## 7.5. Future Work

The eSARD approach tackles the full cycle of hardware-based complex system development. This dissertation has addressed key characteristics of such systems, and some operational aspects. However, since the emphasis has been placed mainly on design tools, milestones, and workflows there is a series of topics and potential research activities for future research and development work such as:

- eSARD workflows, tools, milestones, and perspectives regarding system implementation.
- eSARD workflows, tools, milestones, and perspectives regarding system operations and control.
- eSARD optimization perspectives, techniques, and workflows with other data-driven methods.

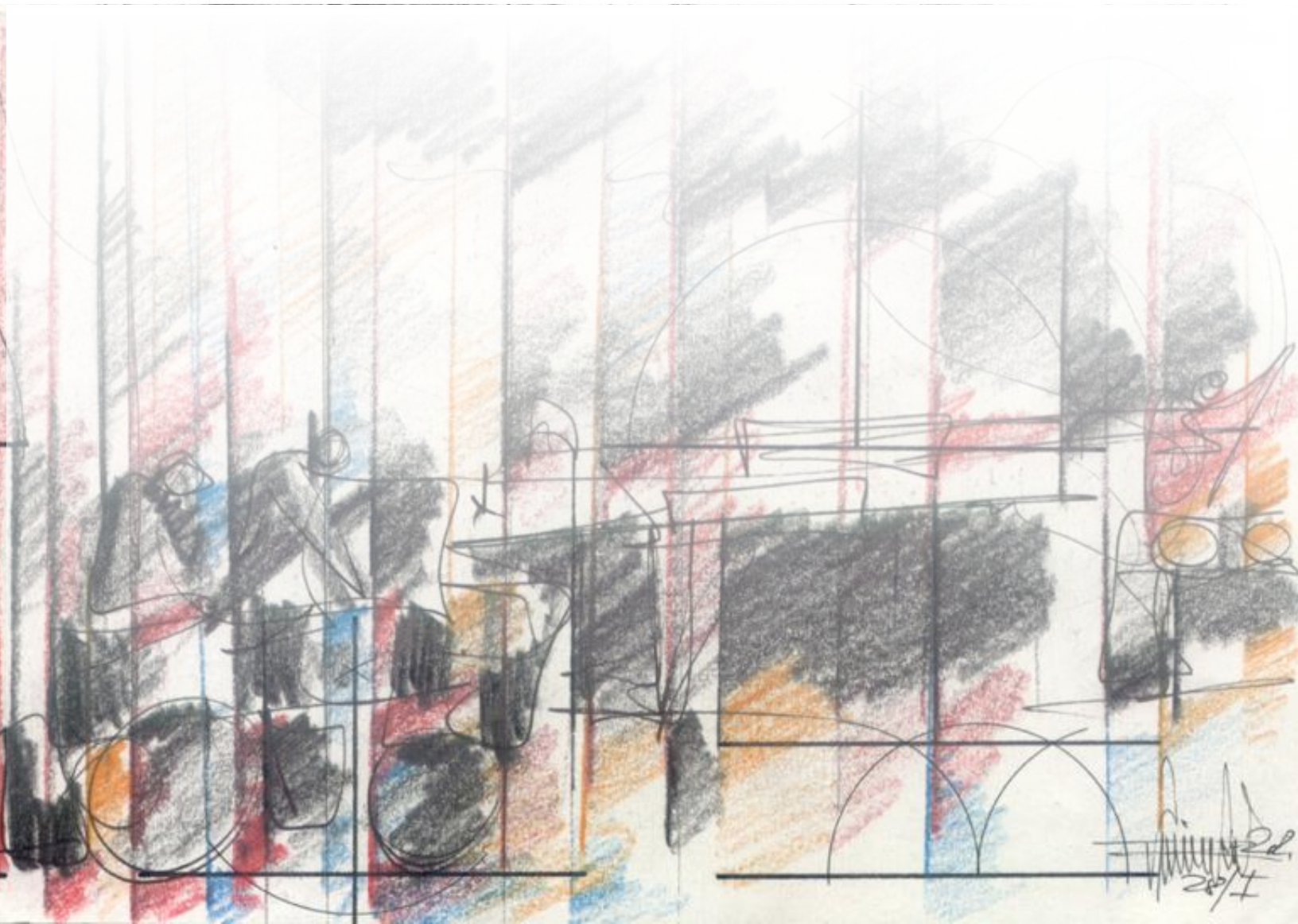
Furthermore, the implications towards using evolutive principles and eSARD methods on virtual and digital systems such as data-based software projects, AI-power techniques, and VR/AR-based designs is another research path to be developed upon this dissertation. The evolutive architecture design approach is a synergetic design platform built upon the shoulder of many giants. This approach brings and applies the power of dynamic questioning, evolutive connections, and continuous adaptable design across subsystems and lifecycles. Furthermore, it also creates the foundation for an adaptable platform ensuring a better pairing between upcoming system needs and new toolset capabilities using the most effective approach to handle change and entropy, evolution.

# EVOLUTIVE ARCHITECURES

## Résumé en français

## CHAPTER 8

*"Il n'est rien de réel que le rêve et l'amour."*
***Anna de Noailles***



# 8. Résumé en français

L'activité humaine sur Terre a été motivée par le besoin d'innover pour progresser. Les besoins de survie, l'avantage concurrentiel et la curiosité intellectuelle, entre autres, nous ont incités à aller au-delà, et souvent contre l'éternelle difficulté de trouver des ressources ou du soutien. Au cours de la deuxième décennie du siècle, la complexité, le patrimoine et la rareté des ressources, entre autres, influencent de plus en plus les architectures complexes en termes de conception, d'optimisation et de mise en œuvre. Basée sur des synergies critiques entre des domaines tels que l'ingénierie des systèmes, l'architecture et la conception technique, cette thèse présente des résultats, des approches et des contributions à une nouvelle méthodologie d'ingénierie de systèmes.

De nos jours, les systèmes complexes tels que les voitures, les ordinateurs, les systèmes robotiques, les plates-formes virtuelles, les bâtiments intelligents et autres dispositifs électromécaniques montrent un besoin croissant de faire des bonds en avant en termes de performance des systèmes, qui sont souvent au-delà des limites de toute connaissance existante. Par exemple, les produits de consommation sont de plus en plus sophistiqués et nécessitent une meilleure intégration du matériel et des logiciels, ainsi que la prise en compte d'autres exigences sociales et culturelles pour être compétitifs. Il y a quelques décennies, des systèmes étaient purement mécaniques, alors qu'aujourd'hui des systèmes comme une automobile, comprennent des centaines de milliers de lignes de code et font appel à d'autres techniques de fabrication disruptives (par exemple, la fabrication additive) pour offrir une meilleure qualité, et une personnalisation plus facile à un prix plus bas. Au-delà de la compétitivité d'un produit sur les marchés mondiaux, la demande de meilleures performances des systèmes (par exemple, des maisons durables consommant moins d'énergie), et l'adaptabilité des systèmes (par exemple, des systèmes modulaires) est une tendance croissante. En effet, de nos jours, les entreprises de télécommunications ne se contentent plus de transmettre des données sur de grandes distances, mais doivent également tenir compte des tendances sociales, de la connectivité des sous-systèmes et des expériences des utilisateurs. En substance, cette nouvelle complexité inhérente est considérée dans cette recherche comme une réalité multidisciplinaire en réseau plutôt que comme un défi unidimensionnel. Notre monde devient rapidement plus complexe, nos méthodes de conception doivent donc évoluer en parallèle.

Dans le même temps, nous entrons dans une toute nouvelle phase en termes de disponibilité des ressources en raison de l'incertitude climatique et de la croissance démographique, ainsi que d'autres facteurs socio-économiques. Par conséquent, l'équilibre entre le besoin de complexité et la disponibilité des ressources (par exemple, l'énergie, la main-d'œuvre, les matériaux de construction, etc.) entre dans un nouveau paradigme, qui est le point de départ de cette recherche. Quel que soit le domaine d'activité (architecture, construction automobile, finance, conception de produits, médecine, aérospatiale, etc.), la nécessité d'aller au-delà en termes de performance, de nouveauté, d'efficacité, d'unicité et d'adaptabilité du système devient une force majeure dans la conception de tout système technique complexe. Les marchés, les clients ne cesseront d'exiger davantage de l'architecture des systèmes, ce qui aura une incidence sur leur nature en tant que système (artefact) et sur la manière dont ils sont développés (méthode) au cours des multiples phases de développement, telles que la conception, l'optimisation, le prototypage, la mise en œuvre, la gestion et la durabilité. Ainsi, compte tenu des contraintes patrimoniales et de la rareté des ressources, comment pourrions-nous atteindre de bien meilleurs niveaux de performance et de capacité des systèmes lors du développement de nouveaux systèmes complexes ?

Cette thèse présente les bases théoriques, les analyses documentaires, les lacunes de la pratique, les méthodologies et les cas d'étude d'une approche technique nouvelle, rapide et synergique de l'ingénierie des systèmes de conception des architectures de systèmes complexes. Inspirée par des principes évolutifs, des principes adaptatifs et des techniques de pointe éprouvées, cette approche de l'architecture évolutive s'attaque à la conception, à l'optimisation et à la mise en œuvre de systèmes complexes sous des contraintes strictes, tout en se concentrant sur des connexions multiples entre les disciplines. L'objectif primordial de cette approche est de surmonter rapidement les obstacles à la conception qui sont motivés par les connaissances existantes, la performance et l'unicité, de la même manière que la nature le fait, en tant que processus continu et efficace reposant sur des synergies plutôt que sur des divisions entre disciplines et sous-systèmes.

Cette thèse est structurée et centrée sur la définition d'une méthodologie tout en soulignant d'autres phases telles que la mise en œuvre et les opérations. Pour illustrer cette méthodologie, un système d'habitat portable intelligent est utilisé comme cas d'étude.

Après un chapitre introductif, une deuxième partie présente le contexte et la raison d'être de cette méthode synergique en abordant les facteurs de stress, les obstacles, les facteurs favorables et les lacunes. La nécessité de concevoir mieux et plus efficacement implique également de le faire plus rapidement et à moindre coût. Cette nécessité est fondée sur les incertitudes climatiques, socio-économiques et techniques à venir, qui entraînent de nouveaux équilibres entre les besoins, les ressources et le patrimoine du système. Les besoins techniques, scientifiques et commerciaux sont en concurrence pour obtenir des performances meilleures et plus rapides, ce qui conduit à des systèmes de plus en plus complexes. À long terme, cela met non seulement à l'épreuve les capacités de conception actuelles, mais rend également plus difficile l'adoption de nouvelles solutions, en particulier pour les secteurs et organisations riches en patrimoine et peu enclins à prendre des risques. En outre, une telle méthode basée sur l'évolution devrait offrir une adaptabilité, une évolutivité et une efficacité face à toute amélioration spectaculaire permise par les solutions actuelles de pointe. Le troisième chapitre présente donc une analyse bibliographique approfondie des méthodes de conception, des théories et des approches d'ingénierie des systèmes. Les tendances de conception frugale, sociale et low-tech, parmi beaucoup d'autres, proposent de multiples options pour faire "plus avec moins", mais cette approche évolutive aborde la question de faire "mieux avec moins" dans le contexte des conceptions "high-tech" inspirées de la nature et des domaines de l'ingénierie des systèmes. Les méthodes évolutives sont axées sur le changement systématique, radical et perturbateur plutôt que sur l'innovation incrémentale.

Dans ce contexte général, le quatrième chapitre met l'accent sur une série de caractéristiques clés des architectures de systèmes basés sur le matériel, qui deviennent de plus en plus critiques en raison des multiples sources de pénurie de ressources, ainsi que de la nécessité de gérer une complexité de système beaucoup plus grande, à la fois en tant que produit et dans le cadre du processus de développement. Les architectures évolutives se définissent par une approche régénératrice de l'utilisation des ressources, une adaptabilité de haut niveau du système et un mode opérationnel axé sur la réactivité.

Dans la perspective évolutive, tout système complexe peut être décrit par sa géométrie (principes descriptifs), son comportement (principes fonctionnels) et sa substance (nature des composants). Le chapitre suivant expose donc la méthodologie de conception de systèmes évolutifs dans ce contexte. Le chapitre 6 présente un cas d'étude qui illustre les principes, les étapes, les outils et les critères évolutifs utilisés pour obtenir rapidement et efficacement des solutions de conception réalisables et ultra-performantes, tout en étant agnostique vis-à-vis des outils. Cet exemple fournit la base de référence pour répondre à toutes les questions de recherche ainsi que pour obtenir des conclusions pour les multiples contributions développées et présentées dans cette thèse de doctorat.

Le fondement de cette recherche repose sur près de 20 ans d'expérience professionnelle dans la conception de systèmes complexes. Cette thèse est complétée par d'autres exemples de recherche fondamentale du domaine public qui ont également été publiés par l'auteur au cours de son activité au *Jet Propulsion Laboratory* de la NASA-Caltech pendant près d'une décennie. Ainsi, les lignes directrices et les résultats présentés dans cette thèse développent une base théorique, applicable à la conception, l'optimisation et la mise en œuvre de toute conception d'architecture de système complexe (évolutive ou non) dans de multiples domaines techniques. Faire "mieux avec moins" est essentiel pour faire face à la rareté des ressources, répondre au besoin d'agilité de la conception, et augmenter l'adaptabilité aux exigences de systèmes plus complexes au-delà de toute solution patrimoniale. En outre, il est également essentiel d'aborder cet objectif dans une perspective holistique, comme le fait la nature, tout en tirant le meilleur parti des techniques de conception et de fabrication actuelles. Ainsi, cette approche crée une base pour l'infusion des méthodes d'automatisation à venir, et d'autres techniques perturbatrices concernant à la fois la conception et la mise en œuvre. Dans un monde confronté à des facteurs de stress croissants et changeants tels que le changement climatique, la croissance démographique, la complexité des systèmes et les pressions constantes du marché, nous proposons un moyen plus efficace d'obtenir des solutions meilleures et plus globales, afin de pouvoir continuer à relever de nouveaux défis.

# 8.1. Introduction (chapitre 1)

## 8.1.1. Motivation, contexte et énoncé du problème

De nos jours, la pratique de la conception de systèmes multidisciplinaires dans les domaines techniques est de plus en plus complexe en raison d'un nombre croissant de facteurs de stress mondiaux tels que la rareté des ressources, la disponibilité de la main-d'œuvre et l'influence du patrimoine culturel et technique, entre autres. Cette accélération de la situation est particulièrement pertinente pour les architectures de systèmes complexes basées sur le matériel, car non seulement elles deviennent un mélange de matériel, de logiciel, de données et d'interaction avec l'utilisateur, mais elles exigent également de nouvelles capacités dans un monde où la révolution basée sur les données atteint notre réalité physique. Souvent, ces nouveaux systèmes n'ont pas beaucoup de patrimoine (au sens de connaissances capitalisées) pertinent, pourtant ils visent des défis exigeant des niveaux de performance beaucoup plus élevés.

L'objectif principal de cette thèse est de fournir une méthodologie de conception fondamentale pour permettre ces systèmes complexes à base de matériel (CHS) qui évoluent rapidement. Plus précisément, l'objectif est de structurer comment évoluer efficacement d'une solution système [A] non adaptable, passive et épuisant les ressources, vers une architecture système [B] hautement adaptable, réactive et régénérative (Figure 1). Cet énoncé du problème implique également l'évaluation de nouveaux contextes, de nouvelles pratiques et de nouvelles caractéristiques du système en tant qu'éléments clés pour l'élaboration d'une nouvelle approche de conception de système adaptable et résiliente. Basée sur des techniques de pointe dans de multiples domaines et inspirée par la nature, cette recherche sur la conception du système est également associée aux flux de travail opérationnels, de mise en œuvre et d'optimisation requis dans un processus de développement de système comme celui-ci.

## 8.1.2. Questions de recherche

Dans un contexte aussi complexe, cette thèse s'articule autour de plusieurs questions de recherche entrelacées :

1. Quelles sont les nouvelles caractéristiques et les besoins de conception complémentaires que présentent ces systèmes ultra-complexes dans des environnements où les ressources sont limitées ?

2. Quels principes pourraient améliorer les processus plus traditionnels de conception et d'ingénierie des systèmes afin de réaliser plus rapidement, mieux et plus efficacement de tels systèmes complexes multidisciplinaires ?

3. Comment une méthode de conception qui tient compte des questions précédentes pourrait-elle être utilisée pour développer plus efficacement des systèmes complexes dans un tel environnement, lorsqu'il n'y a pas d'héritage direct et que la performance du système est une nécessité ?

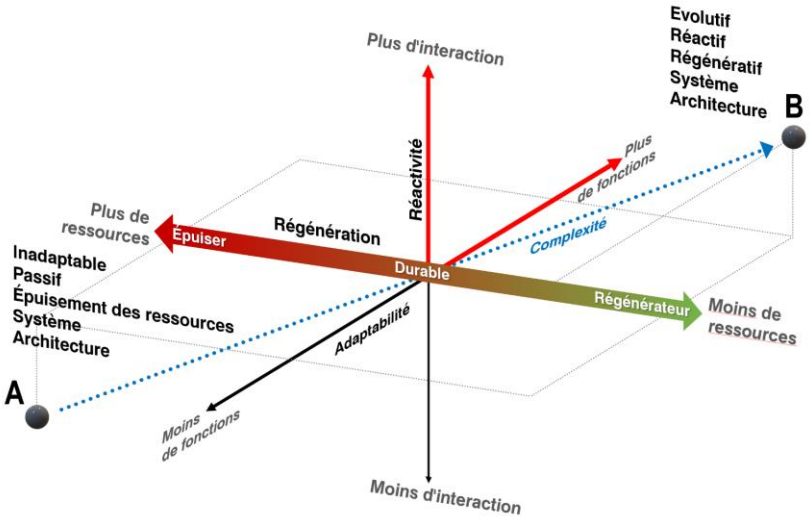



*Figure 185. Représentation tridimensionnelle des coordonnées du design évolutif (adaptabilité, régénération et réactivité).*

## 8.2. Contexte de pénurie : Besoins et ressources (chapitre 2)

En tant qu'espèce, nous avons été guidés par le besoin de faire plus. L'envie de trouver, de découvrir ou d'atteindre de nouveaux progrès a stimulé l'innovation au cours des millénaires, s'attaquant souvent à ce que nous faisons et à la façon dont nous le faisons. Nous avons même classé des périodes de notre histoire (par exemple, l'âge du bronze) en fonction des capacités de conception et de fabrication que nous avons développées (Harari, 2018). Les besoins de survie, l'avantage concurrentiel et la curiosité intellectuelle, entre autres, nous ont incités à repousser toutes les limites, souvent à l'encontre des idées préconçues culturelles, des peurs personnelles et de l'éternel tracas de trouver suffisamment de ressources pour lancer de telles nouvelles entreprises.


*Figure 186. Évolution du type d'informations transmises par les dispositifs de communication du XIXe siècle aux années 2020.*

Au cours de la deuxième décennie de ce siècle, la nécessité de faire face à des niveaux de complexité croissants en termes de conception, de mise en œuvre et de gestion continue de croître (Kravtsov et Kadtke, 1996), car nous exigeons toujours plus de nos architectures, indépendamment du domaine d'application. L'évolution des capacités, et donc de la complexité, des systèmes de communication au cours du siècle dernier en est un bon exemple (voir Figure 186). Dans le contexte de cette recherche, comme présenté en section 1.8.1, le terme architecture (système) fait référence à la définition au plus haut niveau d'un artefact complexe basé sur un sous-système (matériel, logiciel, ou les deux). Par exemple, un bâtiment, une voiture et un produit de consommation électromécanique sont de bons exemples d'architectures génériques parmi beaucoup d'autres. En outre, les données montrent que nous sommes confrontés à une phase à venir en termes de disponibilité des ressources en raison de l'incertitude climatique (OMM, 2020) et de la croissance démographique (Nations unies et al., 2019), tandis qu'un nombre croissant de technologies nouvelles et perturbatrices (Buchholz et al., 2020) pourraient devenir essentielles pour relever ces défis. Par conséquent, l'équilibre entre "ce dont nous avons besoin" (exigences) et "ce que nous pouvons faire" (ressources / capacités), devient un champ d'exploration ouvert de nos jours. Cela s'inscrit très bien dans un nouveau paradigme pour la conception et la mise en œuvre de systèmes complexes. Ainsi, aborder la relation entre les besoins (exigences) et les ressources (contraintes), à travers la perspective de la théorie de l'offre-demande (Sloman et al., 2018) nous permet de comprendre la complexité du développement de l'architecture d'un système (Figure 6) comme un équilibre entre ces forces. Plus les besoins ou exigences sont couverts avec moins de ressources, plus un système devient efficace. Ainsi, la pente de cette courbe pourrait être comprise comme la complexité d'une telle architecture. Suivre une courbe de complexité efficace (ligne bleue) est souvent compliqué, car les contraintes économiques, humaines, culturelles et techniques ont tendance à aplatir cette courbe. Par conséquent, des besoins similaires pourraient être couverts avec une utilisation plus efficace des ressources. Cependant, un saut dans cette efficacité n'est souvent possible que par le biais de nouvelles voies systématiques ou de technologies perturbatrices. Permettre et structurer un tel saut est l'objectif d'un processus de conception évolutif développé dans cette thèse.

Quel que soit le domaine de travail (par exemple, l'architecture, la construction automobile, la finance, la conception de produits, la médecine, etc.), le besoin d'aller au-delà en termes de performance, de nouveauté, d'efficacité, d'unicité et d'adaptabilité devient une force majeure dans tout effort de conception technique complexe. Dans le contexte actuel de l'ingénierie de conception et des pratiques d'ingénierie des systèmes, de multiples facteurs influencent cet équilibre des forces, et ils seront étudiés dans les sections suivantes. Ces facteurs sont à la base de nouvelles approches des systèmes d'architecture complexes et des méthodes de conception qui leur sont associées.

Les conséquences de la rareté des ressources et de la volonté humaine d'aller plus loin poussent les nouveaux efforts et méthodes de conception à faire plus avec moins (Radjou et Prabhu, 2014), ce qui pousse essentiellement ces efforts vers la partie supérieure gauche de la courbe besoins-ressources présentée à la **Error! Reference source not found.**. Les marchés, les clients et les exigences exigent davantage de toute architecture de système, ce qui a une incidence sur ce qu'elle est en tant que système et sur la manière dont elle est développée en tant que méthode. Cette pression se répercute sur toutes les multiples phases de développement du cycle de vie d'une conception, telles que la conception, l'optimisation, le prototypage, la mise en œuvre, la gestion et la durabilité (Pahl et al., 2007).

Au fil du temps, des tendances et des pratiques, il existe de multiples façons d'aborder l'équilibre entre les tendances représenté dans la figure 6. Alors que la section 3 présentera une revue de la littérature élaborée et détaillée dans les domaines de l'ingénierie de conception, de l'ingénierie des systèmes et des principes d'évolution, il est utile de mentionner certaines approches générales pour clarifier le contexte de cette recherche. L'équilibre entre les exigences et l'utilisation des ressources peut également être compris comme l'équilibre entre la technologie, l'utilisateur, la complexité et le coût. De ce point de vue, nous pouvons identifier plusieurs tendances telles que :

- La low-tech qui aborde les problèmes complexes avec un design et une technologie simple (Hirsch-Kreinsen et Jacobson, 2008), répondant au manque de ressources ainsi qu'aux besoins humains associés (Philippe, 2020). Cette tendance a eu une grande influence dans les années 70 avec l'approche "do-it-yourself" (DIY) (Wolf et Mcquitty, 2011). Sous cette tendance, nous pourrions également inclure les termes small-tech, no-tech, slow-tech et design passif, entre autres variations de principes connexes.
- Le design social aborde la complexité en plaçant les besoins sociaux et humains au centre de ce processus (Margolin et Margolin, 2002). L'utilisation et le niveau de la technologie ne sont pas aussi importants que la responsabilité qui la sous-tend. Cette approche a des applications dans tous les domaines, touchant des domaines complexes tels que l'urbanisme et l'architecture (Michael et Lin, 2018).
- La conception frugale telle que décrite dans cette intro consiste à faire plus avec moins (Radjou et Prabhu, 2014) tout en apportant la notion de contrôle et d'équité dans cet équilibre grâce à l'innovation (Micaëlli et al., 2016). La clé est l'efficacité de l'approche, et elle est applicable à différents niveaux de technologie ainsi qu'à différents types d'équilibres.
- La haute technologie, la deep-tech ou la frontier-tech, au contraire, répondent à ces batailles critiques en s'appuyant fortement sur des solutions techniques de pointe qui ne reflètent pas nécessairement d'autres aspects sociaux et d'innovation (Steenhuis et Bruijn, 2006). L'influence de cette perspective peut être observée dans les solutions d'architecture (Macdonald, 2019), les projets de technologie de l'information (Cortright et Mayer, 2001), les nouveaux développements de l'IA (Malach-Pines et Özbilgin, 2010), etc.

Néanmoins, cette recherche embrasse tout ce spectre avec une perspective large en étant agnostique quant au niveau de technologie utilisé dans cet équilibre, ainsi que d'autres aspects tels que l'innovation ou les approches sociales. L'objectif est toutefois de s'intéresser d'abord à la capacité du système, puis à la méthode de conception. Tout autre domaine de cette activité affectant la complexité de ce défi pourrait et devrait être abordé indépendamment de la position adoptée. En d'autres termes, d'un point de vue vraiment général, il s'agit d'obtenir le meilleur et le plus optimisé des résultats du système.

Ainsi, l'objectif principal de cette recherche est de savoir comment nous pourrions atteindre plus efficacement de meilleures performances et capacités de système lors du développement de nouveaux systèmes complexes qui n'ont pas d'héritage antérieur. En outre, la rareté inhérente des ressources souligne également la nécessité d'une approche de conception et d'ingénierie système suffisamment adaptable aux changements à court et à long terme concernant les exigences, les contraintes, les méthodes, etc. En d'autres termes, l'objectif de ce cadre théorique (validé par des années d'expérience pratique dans différents secteurs) est de déterminer comment concevoir plus rapidement et plus intelligemment en faisant non seulement plus mais mieux avec moins. Il est toujours nécessaire de s'attaquer aux principaux facteurs de stress qui affectent l'équilibre dans le développement de toute architecture complexe, mais cela est particulièrement pertinent dans les conditions de pénurie prévisibles qui affecteront la pratique technique de l'architecture et de l'ingénierie dans les prochaines décennies. Ainsi, les sections suivantes abordent les facteurs de stress liés à la conception de contextes multiples.

La conception de systèmes complexes basés sur du matériel doit aujourd'hui faire face à une série de facteurs de stress dus à des changements croissants dans le contexte de nombreuses pratiques techniques. Ces facteurs de stress, que les sections précédentes ont résumé, influencent actuellement toute pratique de conception de systèmes, mais ils constituent également une tendance croissante. L'absence d'héritage réel pour un nouveau système, l'augmentation de la rareté des ressources et l'influence culturelle d'une façon établie de faire des affaires soulèvent une question qui affecte à la fois le résultat du produit et le processus de conception qui le sous-tend : Comment rendre un processus de conception multidisciplinaire suffisamment efficace (Rowan, 2019) lorsqu'un défi est relevé pour la première fois alors que les résultats potentiels exigent une approche radicalement nouvelle et que de nouvelles méthodologies sont très probablement nécessaires ?

Compte tenu de la complexité inhérente derrière de telles architectures de système dans tous les domaines, l'objectif de la réponse à cette question pourrait être davantage de créer une base solide, universelle et adaptable qui permet une telle nouvelle conception, plutôt qu'une approche statique qui pourrait facilement devenir trop adaptée à un domaine spécifique ou à un contexte déterministe donné. La nature même de ces facteurs de stress rend nécessaire l'adaptabilité de toute approche qui les aborde de manière robuste. Une méthode permettant de trouver des architectures de système plus rapides et plus matures offre une plateforme puissante pour réduire l'utilisation des ressources (par exemple, la main-d'œuvre, les calculs, etc.) tout au long du cycle de vie du système, de l'exploration et de l'idéation à la mise en œuvre et aux opérations.

Le monde d'aujourd'hui évolue de façon spectaculaire, sous de multiples angles à la fois. Tout processus visant à concevoir, développer et mettre en œuvre des systèmes actuels et futurs doit tenir compte de ces conditions changeantes. En outre, ces systèmes eux-mêmes doivent être capables de faire face à des conditions qui évoluent rapidement, de réduire l'utilisation des ressources et d'adopter toutes les capacités perturbatrices que les nouvelles méthodologies de conception et de mise en œuvre peuvent permettre.

## 8.3. Conception, systèmes et évolution : Revue de la littérature (chapitre 3)

Ce chapitre traite de l'héritage, de l'état de l'art et des lacunes liées aux théories, outils et méthodologies appliquées en conception, et en architecture de systèmes complexes. Ces approches et pratiques proviennent des domaines de l'ingénierie, de la biologie et de l'informatique, et elles représentent la base du développement d'une approche nouvelle et complémentaire de conception de systèmes évolutifs. Ainsi, ce chapitre est structuré en quatre parties.

Les trois premières parties traitent des méthodologies dans ces domaines :

- **Ingénierie de la conception** (3.1). Cette section passe en revue les méthodologies et les théories de conception à travers l'histoire de l'humanité d'un point de vue multidisciplinaire vers les architectures de systèmes et les systèmes complexes.
- **L'ingénierie des systèmes** (3.2).
- **Théories et conception évolutives** (3.3). Cette section comprend des principes, des méthodes et des applications dans différents domaines et disciplines. Cette section est tout à fait fondamentale puisqu'elle aborde à la fois une vue d'ensemble des principes de l'évolution naturelle, ainsi que leur application aux techniques actuelles de calcul évolutionnaire et autres techniques d'ingénierie.

Cet état de l'art des méthodes est effectué dans la perspective de conception d'architectures de systèmes physiques. L'objectif principal est d'aborder les méthodologies de conception en tenant compte de contraintes telles que la complexité, l'héritage, la rareté et l'agilité, entre autres. Alors que chaque section présente les conclusions et les lacunes à la lumière de cette recherche, la dernière section 3.4 introduit une conclusion générale comme clé de voûte de cette thèse.

De multiples raisons sont à l'origine de cette revue de littérature à travers les domaines. Elles sont résumées dans les points suivants :

- **Concevoir pour la complexité**. Aborder le processus de conception et la gestion du processus de conception d'une architecture de système complexe a été une pratique en évolution depuis les débuts de la civilisation. Au cours des dernières décennies, la notion de conception a été appliquée non seulement au système physique, mais aussi aux

logiciels et autres services. Il est donc essentiel de comprendre et d'identifier les principales lacunes dans l'ensemble du spectre de cette activité, en tenant compte des éléments suivants [1] le temps, [2] le cycle de vie, [3] le domaine, [4] les capacités logicielles et matérielles, [5] l'efficacité et [6] la vitesse, entre autres. Dans le domaine des systèmes physiques, ce point aborde la définition géométrique et la gestion des pièces, composants, assemblages et autres visualisations techniques.

- L'autre aspect de ce processus consiste à gérer les aspects **non géométriques d'une architecture de système complexe**, y compris la documentation, le développement, la définition, les bases d'optimisation, etc. C'est le domaine de l'ingénierie des systèmes (SE) et, par conséquent, cette littérature passe en revue les tendances théoriques et pratiques dans ce domaine. Néanmoins, dans le vaste domaine de l'ingénierie des systèmes, cette recherche s'intéresse particulièrement aux méthodologies orientées vers une manière plus efficace d'aborder les grands systèmes complexes, indépendamment de leur nature logicielle ou matérielle.
- Au cours des dernières décennies, la complexité et l'efficacité ont souvent été abordées dans des domaines techniques avec des **méthodologies inspirées de la nature**. La biologie en général, et la sélection naturelle en particulier, sont en effet devenues deux domaines critiques dans cette approche. Par exemple, les techniques de calcul évolutives telles que les algorithmes génétiques ont donné naissance, dans les années 90, à une nouvelle approche de la programmation et de l'optimisation des systèmes. Il est donc essentiel de passer en revue toute la littérature disponible sur les points suivants : [1] les principes naturels fondamentaux utilisés par ces techniques, [2] les applications pratiques du point de vue de la conception des systèmes, des calculs, des logiciels et du matériel.

Bien que ces domaines puissent sembler sans lien entre eux, la réalité est qu'ils sont étroitement liés à la conception et à l'optimisation de la conception de tout système complexe. D'un autre côté, une perspective aussi large pourrait également présenter de multiples lacunes potentielles entre ces domaines et, surtout, entre les connexions et les synergies clés entre eux. Une méthode pour aborder un problème multidisciplinaire nécessite une base multidisciplinaire. Les sections suivantes présentent les résultats et les examens de ces techniques de pointe, ainsi que les lacunes critiques en matière de conception.

La section 1.8 a déjà présenté les multiples définitions utilisées dans le cadre de cette recherche. Néanmoins, les domaines de la conception technique et de l'ingénierie des systèmes pourraient être imbriqués dans certaines de ces méthodologies et techniques. Par exemple, les concepts de la pensée conceptuelle et de la pensée systémique en ingénierie se chevauchent (Greene et al., 2017). Par conséquent, dans ces cas, il convient de noter ces liens, et leur approche ne sera étudiée que dans l'une des sections.

L'étude des techniques et des pratiques de pointe dans les domaines de l'ingénierie de conception (DE) et de l'ingénierie des systèmes (SE) conduit à plusieurs conclusions générales qui sont résumées dans cette section. De plus, cette revue de la littérature sur les principes généraux d'évolution et leurs applications fournit également plusieurs points clés concluants et fondateurs pour cette recherche visant directement le domaine de la conception d'architectures de systèmes basés sur le matériel.

Les **théories et méthodologies d'ingénierie de conception étudiées** dans la section 3.1 présentent une approche générale de "diviser pour mieux régner". Cela implique que les sous-systèmes, et en particulier les disciplines, ont tendance à être abordés indépendamment les uns des autres en suivant un processus d'agrégation séquentiel tout au long du cycle de vie de la conception. Ainsi, les disciplines sont essentiellement abordées l'une après l'autre et parfois partiellement en parallèle. D'autre part, des méthodologies de conception très solides, telles que la conception prescriptive, englobent à la fois l'analyse et la synthèse d'un point de vue scientifique, présentant en général deux tendances. Elles peuvent être plus [1] créatives (par exemple, la conception innovante) avec d'énormes capacités pour résoudre des problèmes complexes, mais moins puissantes que les techniques de conception détaillée, ou [2] plus rigides, comme les techniques axées sur les méthodes (par exemple, axiomatiques), mais avec une base puissante pour les flux de travail informatisés, ce qui les rend moins capables de trouver des solutions plus innovantes sans héritage.

De même, les outils et les flux de travail décrits à la section 3.1 présentent à la fois une base et une approche pratique autour de ces deux tendances opposées. Les développements informatiques et les techniques axées sur les données ont permis d'intégrer simultanément l'analyse et la conception dans le processus de conception et de développement, mais cela

se fait encore principalement d'un point de vue paramétrique. En substance, l'ingénierie de la conception au 21e siècle présente un écart entre la conciliation de la synergie multidisciplinaire caractéristique de techniques plus simples (et plus anciennes) telles que celles créées à l'aube de la pratique de l'architecture, et les nouvelles capacités analytiques et axées sur les processus qui sont aujourd'hui renforcées par l'intelligence artificielle et les techniques d'apprentissage automatique. Ainsi, l'adaptabilité est le chaînon manquant entre ces voies apparemment opposées.

Néanmoins, depuis les années 1950, les grands systèmes complexes nécessitent de plus en plus d'aborder des aspects non géométriques. **L'ingénierie des systèmes complexes** est devenue une troisième branche très solide, en plus des approches de conception décrites précédemment. Une étude approfondie de la revue de la littérature et des pratiques de pointe réalisée dans la section 3.2 a montré que le développement très rapide de ce domaine au cours du dernier demi-siècle a donné lieu à de multiples approches, théories, flux de travail et techniques. En général, depuis les débuts basés sur les documents jusqu'aux techniques de pointe actuelles basées sur les modèles, l'IS a considéré le système comme une construction abstraite (modèle), mais il y a eu des défis concernant la façon d'intégrer la géométrie dans le processus d'IS.

En général, la plupart de ces techniques examinées présentent une méthodologie rigide (partiellement associée à des méthodes DE), à l'exception des méthodologies itératives (IID), du langage OPM et du squelette qui exercent une approche de conception de système plus continue. En outre, les méthodes d'ES n'ont pas tendance à reconnaître pleinement le cycle de vie complet des systèmes complexes, ignorant souvent des phases telles que le recyclage, la réaffectation et le déclassement, pour n'en citer que quelques-unes. Cependant, il n'y a pas d'intégration claire entre les théories et leur pratique. Parallèlement, le flux de travail associé à ces pratiques tend à être rigide et ne présente pas beaucoup de synergie entre les disciplines qui sont abordées par les activités de SE ou avec d'autres processus de conception, comme exposé dans la section 3.2.

La flexibilité des flux de travail et des méthodes lorsqu'il s'agit de systèmes complexes et hautement adaptables est le dénominateur commun entre la conception et l'ingénierie des systèmes dans le cadre d'un effort conjoint. Les techniques inspirées de la nature qui traitent de la conception complexe tout en augmentant l'efficacité et la qualité des systèmes ont été développées principalement dans le domaine de l'informatique (par exemple, les algorithmes génétiques), mais aussi dans celui de l'ingénierie des systèmes.

**Les méthodes de calcul évolutif (EC)** sont des techniques très performantes et souvent rapides pour l'optimisation des systèmes dans les processus axés sur les données. Cependant, elles présentent des lacunes importantes par rapport à une méthodologie complète capable de créer des systèmes complexes basés sur le matériel. Certaines de ces techniques, telles que les techniques de conception évolutive et les applications robotiques, constituent des approches efficaces pour gérer et intégrer les informations géométriques. L'examen de la littérature révèle qu'il s'agit de méthodes évolutionnistes quantifiables, mais qu'il existe des lacunes en ce qui concerne les méthodes capables de produire des flux de travail évolutionnistes qualifiables en utilisant un point de vue multidisciplinaire à des fins de mise en œuvre et de développement.

Enfin, l'étude des principes naturels de base qui sous-tendent l'EC basée sur l'évolution naturelle inspire des mécanismes capables d'embrasser la continuité, la flexibilité et l'héritage non seulement du point de vue des données ou de l'information, mais aussi du point de vue de la conception du matériel. La section 3.3 présente le potentiel et l'importance de la prise en compte des processus de développement dans la création d'un système complexe et efficace (par exemple, l'évodévo).

## 8.4. Architectures de systèmes évolutifs (chapitre 4)

Le contexte présenté dans la section 2 définit une série de caractéristiques clés affectant la pratique actuelle et future des disciplines de l'ingénierie de conception et de l'ingénierie des systèmes. Au-delà de ces pratiques, les architectures de systèmes en tant que solutions sont également influencées par les effets de la rareté, de l'agilité, de la complexité et de l'héritage. Ceci est particulièrement critique pour la conception de systèmes matériels à haute performance en raison de la complexité inhérente des systèmes (CHS), du niveau de performance qui leur est souvent associé, ainsi que de leur nature multidisciplinaire. Ainsi, cette recherche se concentre sur deux domaines :

- **Les architectures de systèmes évolutifs (eSAR)** comme une classe de systèmes dans de tels contextes.

- **La conception d'architectures de systèmes évolutifs (eSARD)** comme une méthodologie de conception pratique pour modéliser de telles architectures.

Cette approche de conception évolutive peut certainement être appliquée à tout développement d'architecture de système, indépendamment du domaine d'application, et elle est présentée comme un cadre théorique. L'objectif global est d'augmenter les performances et l'efficacité des systèmes au-delà de tout héritage existant, tout en utilisant une perspective agile et au niveau du système.

La section suivante aborde plusieurs sujets définitifs tels que : [1] l'approche évolutive globale (section 4.1), [2] les caractéristiques clés des systèmes évolutifs (section 4.2), [3] les contraintes et les facteurs de conception (section 4.3), [4] la définition et la méthodologie eSAR (section 4.4), [5] la complexité en tant qu'intégration évolutive (section 0), et [6] la conclusion générale (section 4.6).

## 8.4.1. Les clés de voûte du système évolutif

### *8.4.1.1. Adaptabilité*

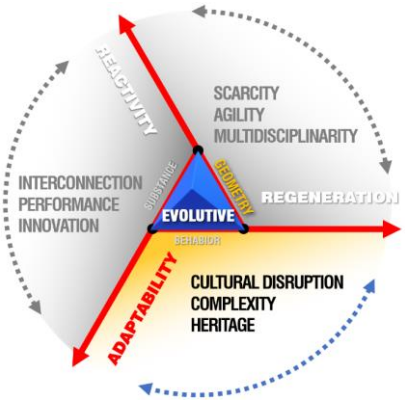



*Figure 187. Adaptabilité dans le tétraèdre évolutif de la conception de l'architecture du système.*

L'adaptabilité est une clé de voûte du tétraèdre évolutif qui met en évidence les aspects géométriques et l'activité de conception (Figure 187**Error! Reference source not found.**). Comme cela a été présenté précédemment, cette clé de voûte de la conception évolutive est liée à tous les facteurs de stress du contexte, mais elle est particulièrement liée à la complexité, au patrimoine et à la perturbation culturelle.

En outre, cette clé de voûte peut également être comprise comme étant directement liée au concept de patrimoine continu. Toute conception dans le cadre de l'approche évolutive peut être partiellement basée sur des solutions précédemment éprouvées (patrimoine), mais elle est aussi toujours une instance dans un processus de conception continu qui s'adapte constamment aux nouveaux changements.

Par conséquent, l'adaptabilité concerne en fin de compte la capacité d'un système à répondre aux changements environnementaux, culturels et de conception. Plus son environnement contextuel exige de changements, plus la conception doit être modifiée pour permettre de nouvelles fonctions du système en réponse. L'adaptabilité de la conception d'un système peut alors être mesurée par sa fonctionnalité de conception relative. Il s'agit du nombre de fonctions qu'une géométrie de système peut exécuter compte tenu d'une capacité d'interaction spécifique et d'un niveau d'utilisation des ressources. Dans ce paradigme, l'objectif est donc de faire mieux avec moins. Ainsi, plus les fonctions qu'un système peut exécuter sont nombreuses et de qualité, moins il a besoin de ressources et plus il est adaptable. De plus, dans cette optique, la géométrie du système, y compris la forme, les éléments d'assemblage, etc., est liée de manière critique à la fois à ses fonctions (comportements) et aux ressources qu'il utilise (substance). Dans ce cas, le concept de comportement se réfère au système lui-même, et non au processus de conception tel qu'il pourrait être compris dans le cadre du FBS de Gero (Gero et Kannengiesser, 2004).

L'architecture d'un système évolutif vise à être hautement adaptable au niveau du système et des sous-systèmes. Cela se produit par conception, tant du point de vue matériel que logiciel. Elle est également rendue possible par la nature d'un processus de conception évolutif continu tel qu'il a été présenté précédemment. Ainsi, de multiples instances peuvent être créées simultanément en tant que résultat de la modification des variables clés du système qui définissent ses caractéristiques les plus pertinentes. Par exemple, la conception d'une pièce d'habillement telle qu'une veste de pompier (figure 114) dans le cadre de cette approche porterait sur plusieurs couleurs et matériaux, mais aussi sur les capacités ultérieures de protection thermique et contre les intempéries. Lorsque cette veste est conçue, une base de référence est créée, et elle peut facilement être modifiée pour que les modèles puissent prendre en compte plusieurs tailles (par exemple, petite, moyenne, grande) et différents matériaux (par exemple, couleur, texture, propriétés, réfléchissants, etc.). De cette façon, la conception peut répondre à différentes situations chimiques et thermiques, ainsi qu'à des modifications, des mises

à jour, des améliorations (p. ex. résistance chimique, protection thermique, etc.) et des solutions spéciales comme l'identification, les conditions d'éclairage, etc. (Watkins et Dunne, 2015). Ainsi, le système lui-même est un produit unique, mais il appartient à une collection qui comprend des systèmes aux caractéristiques similaires et aux multiples variations. Cependant, même si le système lui-même ne nécessite pas de variations, cette perspective apporte d'énormes avantages ultérieurs concernant les mises à niveau, la réaffectation du travail et, en fin de compte, l'efficacité du système. La conception de l'adaptabilité s'attaque aux contraintes de mise en œuvre et aux facteurs fonctionnels dès le début du processus de conception, ce qui permet de réduire le coût relatif global du processus de conception (temps, ressources, main-d'œuvre, etc.) et d'améliorer les performances du système lui-même. Si cette approche est largement diffusée dans la culture d'une organisation, tous les efforts initiaux nécessaires pour mettre ce processus en ligne seront compensés par l'apport de nouveaux niveaux d'adaptabilité dans les lignes de produits, les équipes et les projets.

Un tel effort de conception pour créer une unité est réparti sur de multiples instanciations de cette espèce d'architecture et porte sur de multiples variables de conception représentées par un cadre en réseau de caractéristiques et de variables, plutôt que par une liste linéaire ou même une matrice d'exigences. Le produit final est le résultat de la pondération de ces besoins qui sont souvent interdépendants ou même opposés entre eux (figure 115). En d'autres termes, dans cette perspective évolutive, les relations entre les variables ne sont pas nécessairement constantes et elles peuvent varier dans le temps en raison de changements dans le contexte externe (par exemple, les facteurs de stress de la conception). Si cette approche ouverte et adaptable pouvait être rationalisée, même lorsqu'il s'agit d'une solution unique (avec ou sans patrimoine), alors non seulement les variations et les mises à niveau de ces solutions seraient plus faciles et moins coûteuses à réaliser, mais elles pourraient être incluses comme des apports patrimoniaux utiles pour des solutions plus récentes ou même sans rapport. Ainsi, la capture et la validation de ces relations font à bien des égards partie du matériel génétique du système et du processus. Ainsi, dans le cadre de cette approche, l'architecture d'un tel système pourrait être définie du point de vue d'un cadre évolutif, comme un réseau adaptable de variables interconnectées qui évolue en permanence (figure 115), plutôt que comme une structure hiérarchique statique (figure 116). L'architecture du système est alors définie par les connexions entre les variables les plus pertinentes, et l'adaptabilité consiste à gérer les changements dans les nœuds sélectionnés du réseau (lignes bleues) dans un cadre évolutif. Plus un tel réseau est fluide, plus le besoin d'une architecture de système adaptable est important pour réduire les coûts et améliorer l'efficacité et la capacité de toute conception de système.

Toutefois, dans la perspective de ce cadre, l'adaptabilité du système peut être repoussée plus loin pour [1] mettre l'accent sur la conception de solutions plus efficaces qui utilisent moins de ressources, ainsi que [2] transformer l'incertitude inhérente aux phases de conception, de mise en œuvre et d'exploitation en un avantage. En substance, plus l'architecture d'un système peut traiter de zones du réseau avec moins de ressources, plus elle devient efficace et plus elle peut gérer d'incertitudes sans augmenter ses efforts ou ses coûts de reconception, de mise à niveau ou d'interconnexion. En outre, plus cette opération est effectuée tôt dans l'effort de conception, plus le processus devient efficace et plus le système a de chances d'être optimisé.

En ce qui concerne cette efficacité, la conception initiale de la veste de pompier pourrait exiger qu'elle soit stratégiquement redessinée ou améliorée, en ajoutant des conditions telles que des manches détachables, une technologie intelligente intégrée (par exemple, des capteurs, des chauffages, etc.), ainsi que d'autres variables de mode culturel plus complexes. Ces paramètres ne faisaient pas partie de la conception initiale et n'étaient donc pas inclus dans les exigences et l'évaluation initiales de l'adaptabilité. Cependant, ces conditions sont déterminées par les utilisations futures prévisibles, les contraintes de fabrication et les changements du marché qui repoussent les limites d'un système futur. Si cette architecture a été conçue avec l'adaptabilité en tête (évolutive), alors une partie de la définition de ses exigences (lignes bleues) inclura également des "nœuds" ouverts, ou des zones pour des variables futures possibles ou incertaines.

Par exemple, une nouvelle technique de couture pourrait permettre d'ajouter facilement des lignes de fermeture à glissière aux manches fixes afin que les manches puissent être retirées. Si cela s'applique à l'architecture elle-même, cela affecte également la méthode de conception. Envisager cet aspect dès le début de la mise en œuvre peut permettre d'écarter des solutions trop uniques, et donc non adaptables. Bien sûr, cela peut sembler très inefficace et peut-être une source de complexité inutile. Cependant, si elle est bien gérée, elle peut être la clé de sa réussite et permettre de faire des bonds en avant dans les performances. Le coût de l'inclusion de nœuds ouverts généraux (en vert), s'il est effectué correctement, est

minime par rapport aux avantages de la réutilisation de cet effort de conception pour des modifications ultérieures. À l'instar de ce que fait la nature avec l'évolution, l'ouverture et le caractère aléatoire de la définition du système (génétique) ouvrent la voie à l'adaptabilité en raison de l'incertitude de la réalité. En d'autres termes, plus le potentiel d'adaptabilité est insufflé tôt dans la conception, meilleure sera la gestion des risques (Costikyan, 2013) et moins le système sera coûteux à moyen et long terme.

Dans le cadre de cette approche de l'adaptabilité évolutive, la sollicitation du système est également une stratégie visant à atteindre la fiabilité et la résilience du système. Si une grande partie ou la totalité des multiples relations décrivant l'architecture du système dans ce cadre sont prises en compte par un système, sa conception répond à des exigences connues et très probablement aussi connues. Plus un système est adaptable aux changements par le biais d'une approche "mieux avec moins", plus ce système est capable de faire face aux incertitudes et plus il devient résilient. En outre, même si cette approche est partiellement mise en œuvre, elle fournit une base solide pour de futures options et extensions de l'espace commercial de la conception.

Cependant, l'intégration et l'ajout d'exigences peuvent également conduire à une hyper-intégration rendant difficiles les mises à niveau, réparations ou mises à jour futures. Il ne faut pas oublier que si une augmentation initiale de la complexité implique un plus grand effort et davantage de variables (et d'exigences), elle se traduit en fin de compte par des efforts de conception beaucoup plus efficaces et une utilisation moindre des ressources disponibles. C'est essentiellement ce que fait la nature, puisque le point de référence n'est pas l'effort de conception vers une conception ponctuelle ou une architecture unique, mais le système (organisme) en tant que partie instanciée d'une évolution continue (espèce).

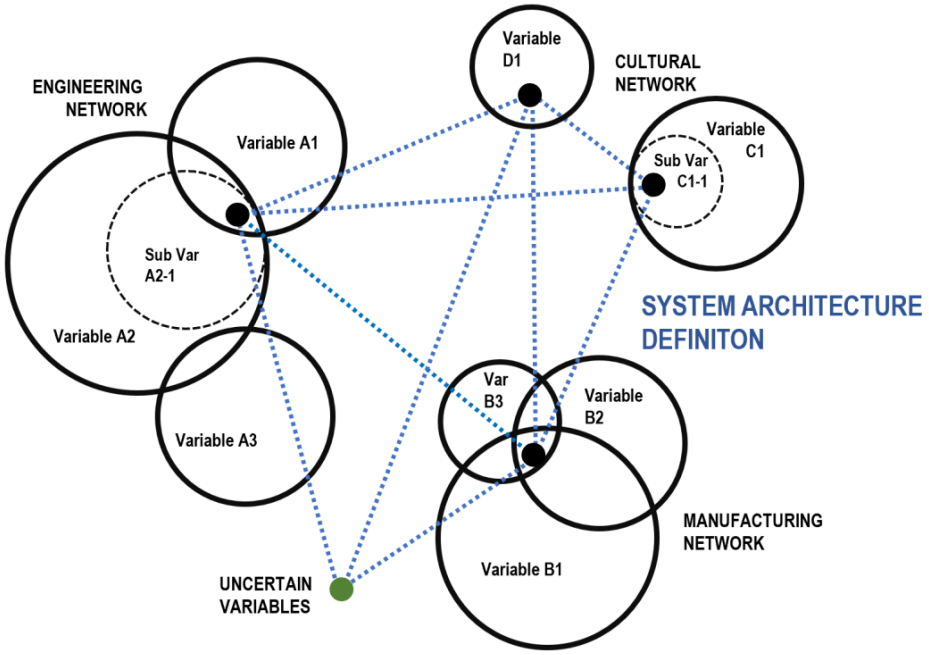



*Figure 188. Représentation visuelle de la définition d'une architecture basée sur un réseau évolutif de variables.*

Ensuite, pour repousser les limites de l'architecture d'un système vers des niveaux plus élevés de performance et d'adaptabilité, il faut suivre un processus minutieux qui repose sur [1] l'équilibre entre les besoins et les ressources, et [2] une connexion synergique entre les sous-systèmes et les disciplines au sein d'un réseau d'exigences. Le chapitre 5 décrira ce processus en détail. En outre, l'introduction d'un haut niveau d'adaptabilité dans la conception a également l'avantage de mieux gérer l'incertitude. Dans une architecture évolutive, le système est considéré comme une solution ouverte, c'est-à-dire une famille de solutions plutôt qu'un point de conception verrouillé, qui se répercute sur plusieurs niveaux tels que les sous-systèmes, les composants, les pièces et même les stratégies (par exemple, la fabrication, le marketing, etc.). Ainsi, l'incertitude de conception s'accumule dans le système sous forme de probabilité, de faisabilité et de disponibilité de

paramètres statistiques qui doivent être saisis, suivis et utilisés pour des optimisations ultérieures.

La portée de cette clé de voûte critique varie sur une plage définie par ces niveaux :

- **Inadaptable (aucune adaptabilité**). Ces conceptions de systèmes présentent le nombre minimum de fonctions avec le nombre maximum d'éléments. Elles ne peuvent pas gérer efficacement des niveaux élevés d'incertitude de conception, et elles ont tendance à graviter vers des solutions uniques de conception ponctuelle, rigides et souvent limitées.
- **Adaptable (adaptabilité équilibrée).** Les conceptions de systèmes de cette catégorie présentent un équilibre entre le nombre de fonctions et leurs éléments constitutifs. Elles tendent vers des séries courtes et des solutions personnalisables limitées.
- **Évolutif (adaptabilité maximale).** À l'autre extrémité du spectre, ces conceptions présentent un nombre maximal de fonctions avec un nombre minimal d'éléments et de composants. Elles gèrent efficacement des niveaux élevés d'incertitude de conception et gravitent vers des solutions ouvertes ou des familles de solutions répondant très bien à des exigences ouvertes.

Le concept d'adaptabilité évolutive peut être appliqué à toute conception d'architecture de système, qu'il soit physique, numérique ou virtuel. Cependant, il est particulièrement pertinent pour les architectures de systèmes complexes et intelligents basés sur le matériel. La nature particulière de ces systèmes complexes qui intègrent des géométries physiques complexes, des fonctions commandées par des actionneurs et des opérations commandées par des données met certainement en évidence de multiples aspects imbriqués de l'approche évolutive, comme le présentera le chapitre suivant.

#### *8.4.1.2. Réactivité*

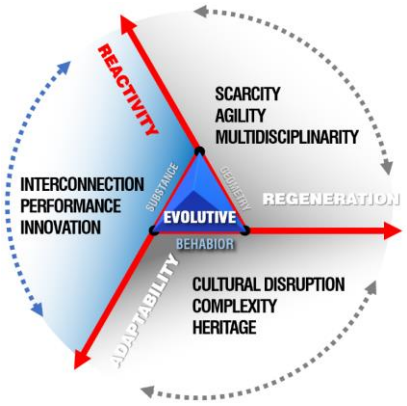



*Figure 189. La réactivité dans le tétraèdre évolutif de la conception de l'architecture du système.*

La réactivité est la deuxième clé de voûte du tétraèdre évolutif de la conception du système (**Error! Reference source not found.**). Ce concept est particulièrement lié à l'interconnexion, à la performance du système et à l'innovation.

Du point de vue de l'évolution, l'architecture d'un système est dynamique et présente une nature multidisciplinaire pour offrir des caractéristiques de haute performance. Par exemple, un système mécanique évolutif pourrait être une voiture de course qui optimise les performances thermiques et la réduction de la masse. En outre, une configuration physique et adaptable répond à l'évolution des exigences de conception, avec des fonctions clés de contrôle et de gestion associées. Dans cet exemple mécanique, la gestion des actionneurs électromécaniques dans l'ensemble, pourrait permettre des améliorations et des ajustements au cours des différentes phases de la course pour améliorer les performances, ainsi que des mises à niveau basées sur les données recueillies au fil du temps. Par conséquent, la conception physique, les commandes des actionneurs et les décisions fondées sur les données sont combinées au sein d'une architecture évolutive adaptable complexe pour réagir aux changements environnementaux ou de conception. Dans ce cas, la réactivité est essentielle pour tenir compte de la synergie dynamique des composants du système, ainsi que de sa capacité à gérer son adaptabilité à travers les multiples réalités d'un système (physique, numérique, virtuelle, basée sur des données, etc.) La réactivité est également liée à la nature transitoire de la complexité évolutive du système, en tant que développement continu entre le système et son environnement. Le matériel (géométrie), le logiciel (comportement) et les ressources (substance) sont tous intégrés dans la capacité du système à interagir avec les changements intégratifs externes et internes.

Par conséquent, l'interaction transitoire du système est la mesure de la réactivité du système et est définie comme le nombre (et la complexité) des réactions que les comportements du système peuvent fournir compte tenu d'une géométrie spécifique du système et de l'utilisation des ressources. Plus l'architecture d'un système est capable d'interactions avec l'environnement, plus elle est réactive. Moins le système a besoin d'interactions pour gérer les changements externes, ou

en d'autres termes, plus il est intelligent et adaptable, plus l'architecture du système est efficace. En substance, l'objectif principal de ce principe est de rendre le système plus intelligent (plus réactif) avec moins.

De nos jours, les architectures de systèmes complexes modernes dans tous les domaines sont de plus en plus robotisées par nature. Cela signifie qu'elles combinent de plus en plus de logiciels, de matériel et de données, par le biais d'un certain type de gestion, d'évaluation et de contrôle intelligents (Chen et al., 2018). Par exemple, une voiture moderne compte aujourd'hui plusieurs millions de lignes de code (Desjardins et McCandless, 2017), ce qui est une tendance croissante alors que l'autonomie commence à devenir une capacité standard de toute voiture à l'avenir (Towns end, 2020). Il en va de même pour les apps ou les logiciels, ainsi que pour les téléphones, les véhicules, les appareils électroménagers et de nombreux autres objets qui nous entourent aujourd'hui. Dans le même temps, la quantité d'informations utilisées dans nos systèmes ne cesse d'augmenter, de sorte qu'une autre tendance technologique croissante apporte la connectivité entre tous ces systèmes, comme l'internet des objets (IoT). Tout cela dépeint un monde à court terme de dispositifs et de capteurs interconnectés partout (Soro et al., 2019). Ainsi, l'infusion croissante de comportements et de contrôles logiciels dans tout système matériel évolue vers une capacité intelligente omniprésente et croissante (figure 118) pour chacun de ces systèmes. En tant que telles, les règles de conception intrinsèques de tout matériel ou matériel piloté par des robots du système vont changer. Par exemple, dans le cadre de cette approche, une maison gèrera elle-même son éclairage ou sa consommation d'énergie en fonction de l'interaction de l'utilisateur, tandis qu'une voiture se conduira toute seule en modifiant sa vitesse, les profils de sa suspension et son couple en fonction de l'état de la route et des stimuli environnementaux. Notre monde construit par l'homme devient plus intelligent et, soudain, la performance thermique, la fatigue mécanique ou la longévité du système seront déterminées par cette capacité inhérente. Ainsi, une approche comme celle-ci offrira de grandes opportunités dans cet équilibre entre la rareté des ressources et la complexité du système.

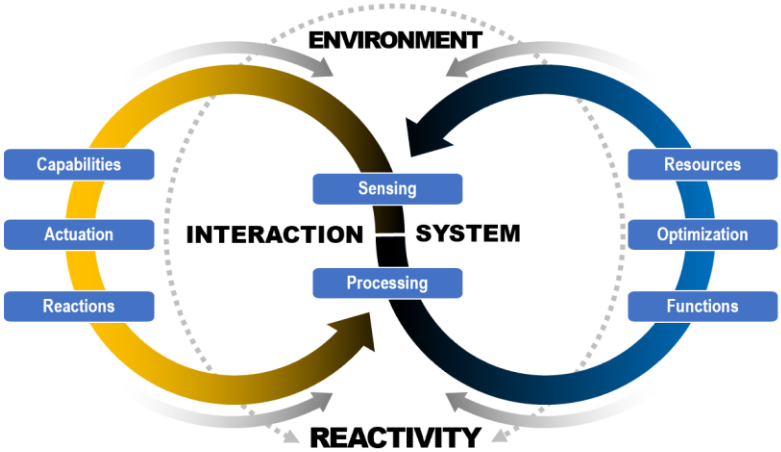



*Figure 190. Réactivité évolutive, conception de systèmes et flux simultané d'interactions entre systèmes.*

Ces systèmes intelligents nécessitent un effort de conception tout aussi intelligent pour exploiter, améliorer et mettre à niveau la capacité de réactivité elle-même, mais aussi un moyen de faire correspondre l'adaptabilité du matériel, l'efficacité du système et l'interactivité entre les systèmes. En d'autres termes, une approche évolutive implique la capacité de préparer et de concevoir un flux constant d'informations et d'interactions entre les systèmes, les composants et leur environnement. Ces approches évolutives croissantes vont modifier les modèles commerciaux et industriels (Kranz, 2017), en affectant la manière dont nous concevons, construisons, fabriquons et utilisons les objets qui nous entourent.

Comme les sections précédentes l'ont mentionné, les organismes (systèmes) dans la nature ne sont pas isolés. Ils sont en interaction constante avec leur environnement, avec d'autres organismes et avec eux-mêmes, quel que soit le volume d'informations et le véhicule de cet échange ou de cette étude. Cette interaction avec l'environnement entraîne : [1] les efforts de conception du système, y compris les réseaux de définition comme le montre la figure 118, ainsi que [2] les processus d'interaction permettant au système de réagir et de s'adapter.

La Figure 190 montre graphiquement comment ce flux simultané affecte chaque processus. En général, le fonctionnement d'un système interactif dans l'environnement lui permet à la fois d'envoyer et de recueillir des données, qui sont utilisées pour effectuer des variations et des changements. Ceux-ci dépendent des capacités du système (par exemple, des pièces mobiles), ce qui permet de réagir au mieux à ces stimuli. Cependant, ce processus a également des conséquences sur la définition du système et sur ses efforts de conception ultérieurs, permettant également des changements dans la conception qui pourraient améliorer ses performances en fonction de chaque nouvelle situation (héritage continu). Comme le présente ce petit résumé, l'architecture du système doit alors être conçue pour permettre ce processus, en plaçant la réactivité et l'adaptabilité au cœur de sa définition.

Au-delà, les contraintes de fabrication et d'exploitation doivent également être intégrées. Actuellement, les architectures de systèmes hautement réactifs, comme les voitures autonomes, présentent un niveau différent d'autonomie et de comportements induits par les données dans le cadre du processus de conception. Le processus de conception lui-même doit donc être modifié, optimisé et évolué en fonction de la gestion de l'information inhérente à cette caractéristique clé. Dans le cadre de cette approche, plus une architecture de système avec une base intelligente est utilisée, plus le processus de conception changera en fonction de cette boucle de données de retour. Cela devient possible grâce à une architecture de données-matériel intégrée et simultanée (figure 119).

La portée de cette clé de voûte critique varie progressivement sur une plage définie par ces points clés :

- **Passif (pas de réactivité).** Ces systèmes présentent une capacité d'interaction minimale avec un nombre maximal d'éléments fonctionnels et d'utilisation des ressources. Ils ne peuvent pas faire face à de nombreux changements externes ou inattendus, et ont tendance à s'orienter vers des solutions simples et de faible technicité.
- **Actif (interaction équilibrée).** Les systèmes présentent ici un équilibre entre l'interactivité et la complexité du système et les ressources requises tout au long de leur cycle de vie. Les systèmes programmables, modulaires et évolutifs ont leur place ici.
- **Réactif (réactivité totale).** Ces systèmes présentent une capacité d'interaction maximale avec un nombre minimal d'éléments fonctionnels et d'utilisation des ressources. Il s'agit de systèmes hautement intelligents tels que la robotique avancée, les architectures pilotées par l'IA, les systèmes autonomes, etc. Ils peuvent gérer efficacement de nombreux changements externes et inattendus avec une grande interactivité. Ils gravitent autour de solutions high-tech, biologiques et logicielles.

Cette clé de voûte évolutive ne s'applique pas seulement aux systèmes très complexes et de haute technologie, mais elle peut également être identifiée dans des conceptions aussi simples qu'une veste d'aventure. Une telle veste évolutive pourrait simplement avoir des manches amovibles et des ouvertures ou des poches qui pourraient être ajustées à la main pour des raisons de gestion thermique. Ce principe évolutif de réactivité est applicable à tous les secteurs techniques et créatifs. Bien que cette approche soit axée sur les architectures de systèmes basés sur le matériel, elle peut également être appliquée aux architectures logicielles ou virtuelles nécessitant à la fois des interactions et une adaptabilité.

#### *8.4.1.3. Régénération : Performance et durabilité des ressources*

Enfin, la régénération est la dernière clé de voûte d'une architecture de système évolutive (Figure 191). De même, elle s'attaque à tous les facteurs de stress généraux mentionnés dans la section 4.1 en tant que dernier fondement du tétraèdre évolutif. Parmi eux, la régénération est particulièrement liée à la rareté des ressources, à l'agilité et à la multidisciplinarité.

La régénération concerne l'utilisation, la gestion et la restauration des ressources tout au long de leur cycle de vie, quelles qu'elles soient. Il peut s'agir de l'énergie, des matériaux de construction, du code informatique, des composants mécaniques ou de la disponibilité de la main-d'œuvre, entre autres. Les ressources ne doivent pas nécessairement être physiques et ne doivent pas non plus être créées par l'homme, mais elles ont toutes une substance. On entend par substance ce dont est constituée l'architecture du système ou la ressource nécessaire à son fonctionnement. Ce concept concerne donc l'optimisation du cycle de vie des ressources d'un système dans un environnement externe donné. Les sources utilisées pour fabriquer le système et leur gestion font partie d'un processus de conception évolutif. La prise en compte des ressources doit se faire sur l'ensemble du cycle de vie, de la production au recyclage, y compris : [1] l'énergie, [2] les matériaux, [3] les personnes ou la main-d'œuvre, [4] les données et [5] le codage ou la programmation.

Par conséquent, une mesure clé derrière ce concept constitutif d'une architecture de système évolutive est la consommation et l'utilisation des ressources pour une géométrie de système et une capacité de réactivité données. En d'autres termes, il s'agit du concept d'utilisation des ressources dans ce contexte. Il s'agit de la consommation de ressources par le système tout au long de son cycle de vie, de la conception au déclassement. En outre, elle prend en compte à la fois toutes les ressources utilisées pour concevoir le système, pour le développer (évo-devo), utilisées par le système lui-même, et par les relations avec son environnement, qu'il soit physique, numérique, ou les deux (éco-évo-devo).

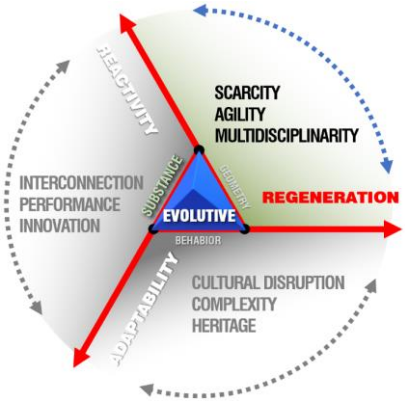



*Figure 191. La régénération dans le tétraèdre évolutif de la conception de l'architecture du système.*

Dans cette perspective, un système évolutif vise non seulement à être durable mais aussi à devenir positif en termes de ressources (par exemple, en produisant plus d'énergie qu'il n'en consomme) ou régénérateur (Lyle, 1996). La première a des implications évidentes sur une approche du berceau au berceau, et il ne s'agit pas seulement de pollution ou de pénurie, mais d'efficacité à travers la conception, la mise en œuvre, les opérations, et jusqu'au déclassement (Bhamra et Lofthouse, 2016). Cette prise en compte des ressources peut être négative (le système ne fait que consommer), neutre (le système est durable) ou positive (le système reconstitue les ressources). Il peut également y avoir plusieurs niveaux parmi ceux-ci, qui sont appliqués au niveau du système ainsi qu'au niveau d'un composant ou d'un sous-système.

Ainsi, la gestion des ressources dans un système évolutif est liée au concept d'éco-évo-évo-cycle de vie. Il s'agit d'une approche d'évo-devo qui examine le processus de développement du système lui-même en considérant l'écosystème environnemental et le cycle de vie du système du point de vue des ressources (**Error! Reference source not found.**). Dans cette perspective, des concepts tels que le recyclage durable (Bhamra et Lofthouse, 2016) et le cradle-to-cradle (McDonough et Braungart, 2010) sont intégrés dans la compréhension et l'optimisation de l'utilisation des ressources à travers plusieurs phases :

- **Dans cette phase**, la conception consiste à prendre en compte à la fois [1] toutes les ressources nécessaires pour créer la conception (par exemple, la main-d'œuvre, les outils, les puissances de calcul, le papier, etc.), et [2] les ressources nécessaires au fonctionnement du système.
- **Mise en œuvre.** La fabrication physique et numérique de l'architecture d'un système nécessite des ressources directes et indirectes telles que les matériaux, l'outillage et le codage. Dans cette phase, il est essentiel de prendre en compte toutes les pertes dues aux inefficacités et autres étapes intermédiaires. Cette phase doit également aborder l'intégration, le transport et l'installation.
- **Opérations**. Cette phase concerne toutes les ressources nécessaires pour exploiter, maintenir et même mettre à niveau le système. De plus, l'exploitation du système est essentielle dans cette phase, tant du point de vue actif que passif, car elle affecte toutes les autres phases du cycle de vie. Entre autres ressources, la gestion de la main-d'œuvre et le codage sont suivis ici.
- **Mise hors service**. Enfin, cette dernière phase prend en compte les ressources concernant la réaffectation, le recyclage ou la réutilisation des systèmes à la fin de leur durée de vie. Cette phase critique va au-delà de la durabilité du système à tout niveau d'utilisation des ressources et relie la fin du cycle de vie au processus de conception initial.

Dans cette perspective et tout au long du cycle de vie d'un système évolutif, la relation entre le système et son environnement est toujours considérée comme un élément constitutif de la conception, indépendamment de toute exigence de conception donnée. Cette relation détermine la durabilité du système, la position de la conception vis-à-vis de la rareté des ressources (section 2.1) et son coût de mise en œuvre. Elle conditionne et sollicite également la conception du système et la méthodologie de conception pour obtenir de meilleures performances. Par exemple, si la conception permet d'infuser, d'utiliser ou d'échanger des matériaux et des sources d'énergie parmi d'autres contraintes, cela augmentera l'adaptabilité de la conception du système et potentiellement les performances du système à long terme.

De même que dans l'approche éco-évo, l'étude et la conception du système se font toujours à la lumière de sa relation avec son environnement changeant. Tout système est donc défini par la conception du système et son contexte. Ce

contexte peut être l'ensemble dans lequel il réside (par exemple, une pièce mécanique), l'environnement naturel (par exemple, un bâtiment), son cadre logiciel (par exemple, une application), etc.

Ainsi, cette troisième clé de voûte est une caractéristique essentielle pour toute architecture de système, mais elle est particulièrement pertinente pour les architectures évolutives dans des environnements axés sur la rareté. Comme indiqué précédemment, la rareté de l'énergie est une contrainte globale générale pour l'humanité, et l'efficacité énergétique est particulièrement critique pour des opérations plus longues et encore plus abordables. L'utilisation de l'énergie est liée à l'utilisation des ressources naturelles, y compris l'extraction des matériaux, le traitement, le prototypage, la fabrication et jusqu'au recyclage (Johnson et Gibson, 2014). Il faut toujours en tenir compte pour des raisons de rareté, de coût et de fiabilité, car elle est essentielle pour tous les facteurs de stress généraux, de conception et même culturels (chapitre 2).

Par exemple, les matériaux recyclables à base de cellulose qui sont disponibles dans la région sont essentiels pour créer une approche évolutive vers l'impression de produits tels qu'un magazine, comme le montre la figure 122. Habituellement, la sélection finale du matériau et du fournisseur pour l'impression intervient à la fin du processus de conception. Mais une approche évolutive prend en compte ces détails clés dès le début du processus, et inclut les sources locales, les alternatives, les schémas de recyclage et les contraintes de fabrication. En outre, l'approche doit tenir compte de la manière dont l'édition peut reconstituer les arbres et l'énergie utilisés pendant sa conception, son impression et sa livraison. Cela conduit à gérer les encres, les formats, les fournisseurs et le transport, ainsi que les approches marketing, les aspects environnementaux et les autres contraintes sociales.

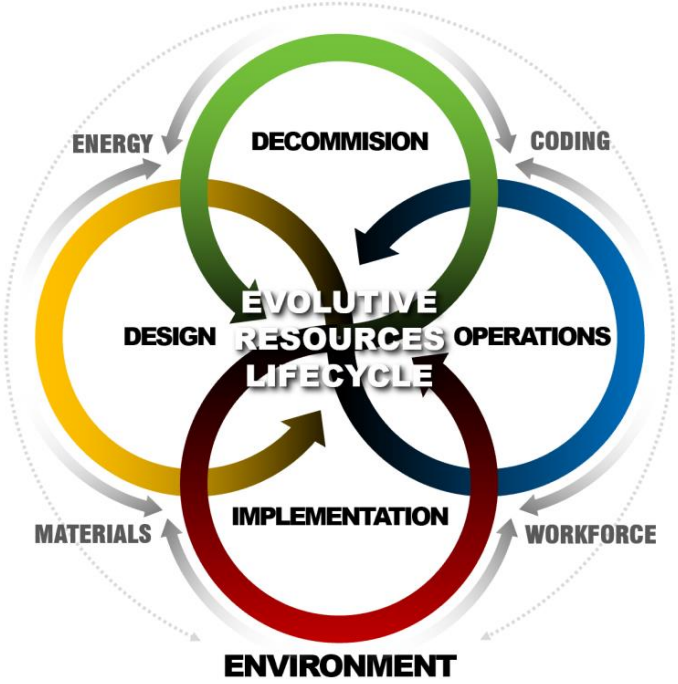



*Figure 192. Cycle de vie complet des ressources évolutives dans le cadre du processus de conception des systèmes évolutifs.*

Tous ces aspects sont pris en compte pour rendre le produit final plus riche, plus adaptable et mieux adapté à son contexte. Une telle approche exige un effort supplémentaire à la fois pour le concepteur et pour le processus de conception, et elle peut certainement devenir écrasante. Cependant, la clé consiste à évaluer laquelle de ces variables et connexions dans une telle interaction système-environnement est critique. Le chapitre 5 présentera ce processus et sa méthode.

Viser un surplus dans le système offre plusieurs avantages du point de vue de la conception, car [1] cela met l'accent sur les exigences de conception permettant une plus grande adaptabilité, [2] cela crée des marges de performance et [3] cela met en œuvre des principes environnementaux clés ayant des conséquences économiques, sociales et de conservation importante. Par conséquent, une architecture de système évolutive pourrait souvent présenter des métiers de conception

clés qui permettent et utilisent ces principes. Ces principes comprennent notamment : une architecture de système multifonctionnelle, la réutilisation, la facilité de mise à niveau, la recyclabilité, la réduction de la masse, la réduction des coûts, l'augmentation du retour sur investissement, etc. En substance, le principe de conception qui sous-tend cette clé de voûte consiste à faire plus avec moins du point de vue des ressources. La gamme de régénération varie progressivement dans une fourchette définie par :

- **Épuisement (consommation nette négative de ressources, consommateur).** Ces conceptions de système présentent la consommation maximale de ressources et aucune stratégie de réapprovisionnement. Ils ont tendance à présenter des niveaux de performance inférieurs, une moindre adaptabilité et une moindre réactivité. Ils s'orientent également vers des solutions non recyclables, jetables et non durables. Un exemple de ce domaine pourrait être les systèmes thermomécaniques dont l'empreinte carbone de fabrication est élevée.
- **Durable (consommation de ressources neutre).** Les conceptions de systèmes de cette catégorie présentent un équilibre entre la consommation et la reconstitution des ressources. Il s'agit notamment des systèmes durables et des solutions neutres en carbone.
- **Régénérateur (consommation de ressources positive nette, pro-sommateur).** À l'opposé, ces conceptions ont une consommation minimale de ressources et une stratégie de réapprovisionnement complet. Ainsi, ils présentent également les niveaux de performance les plus élevés, avec une plus grande adaptabilité et une plus grande réactivité du système. Ces systèmes gravitent vers des solutions nettes positives et régénératives telles que les systèmes électromécaniques basés sur la séquestration du CO2.

Le concept de régénération en tant que clé de voûte d'un système évolutif pourrait être appliqué et observé dans de nombreux domaines, notamment thermomécanique, numérique, informatique et biologique, entre autres. Par conséquent, dans le cadre de cette approche, l'énergie, la matière et l'information (données) sont les multiples facettes de la même réalité en tant que substance des systèmes complexes. Il s'agit d'une manière fondamentalement holistique d'envisager toute architecture de système, indépendamment de sa complexité ou de son échelle.

Des exemples de conception durable qui aspirent à être intégrés à la nature sont de plus en plus visibles dans de multiples secteurs tels que l'habillement, les produits de consommation et les maisons (Kwinter, 2017), parmi beaucoup d'autres. Néanmoins, concevoir et produire pour l'abondance des ressources, plutôt que pour le rationnement des ressources disponibles (McDonough et Braungart, 2013) est ce à quoi s'oppose la régénération. Cela signifie essentiellement [1] de concevoir une architecture de système qui produit plus de ressources qu'elle n'en consomme (Mang et al., 2016), ou [2] d'avoir un schéma fonctionnel intégré en boucle fermée afin que le système restaure, renouvelle et transforme toute énergie et ressource utilisée (Burke, 1999, Colozza et Maloney, 2003). Ceci est particulièrement applicable au développement de systèmes de production d'énergie, de projets de grande envergure et d'architectures orientées vers les infrastructures (Hemenway, 2015). Parmi d'autres exemples à plus petite échelle, nous pourrions identifier les bâtiments durables, les systèmes énergétiques régénératifs (Alotaibi et al., 2020), ou les systèmes de production alimentaire à base de plantes, pour n'en citer que quelques-uns. Cette approche est donc une tendance croissante en raison des facteurs de stress liés à la rareté et de la complexité croissante des systèmes.

#### *8.4.1.4. Facteurs de conception évolutive*

Les sections précédentes ont présenté l'approche de l'architecture des systèmes évolutifs en réponse aux facteurs de stress de la conception globale et aux lacunes méthodologiques. Ces clés de voûte fondamentales caractérisent tout système évolutif dans la grande catégorie des systèmes complexes généraux. La figure 123 résume cette approche en mettant en évidence les trois clés de voûte à la base du tétraèdre de la conception évolutive : adaptabilité, réactivité et régénération. Cependant, il est nécessaire de définir pleinement un système évolutif afin d'aborder des facteurs de conception plus spécifiques derrière ces principes clés de voûte.

Ces clés de voûte reposent en partie sur les composantes architecturales de base ancestrales de tout bâtiment, qui ont été décrites par Vitruve sous l'Empire romain (Vitruvius, 2012). Il s'agit de la structure (firmitas), de la fonction (utilitas) et de la perception (venustas). Mais ces principes doivent être adaptés à l'époque actuelle. Il est pertinent de souligner que, contrairement à d'autres interprétations modernes telles que le modèle FVS de Gero (Gero et Kannengiesser, 2014), ces principes et les moteurs de conception ultérieurs se rapportent au système lui-même et non au processus de développement

de la conception qui le sous-tend, car il s'agit d'une approche pratique.

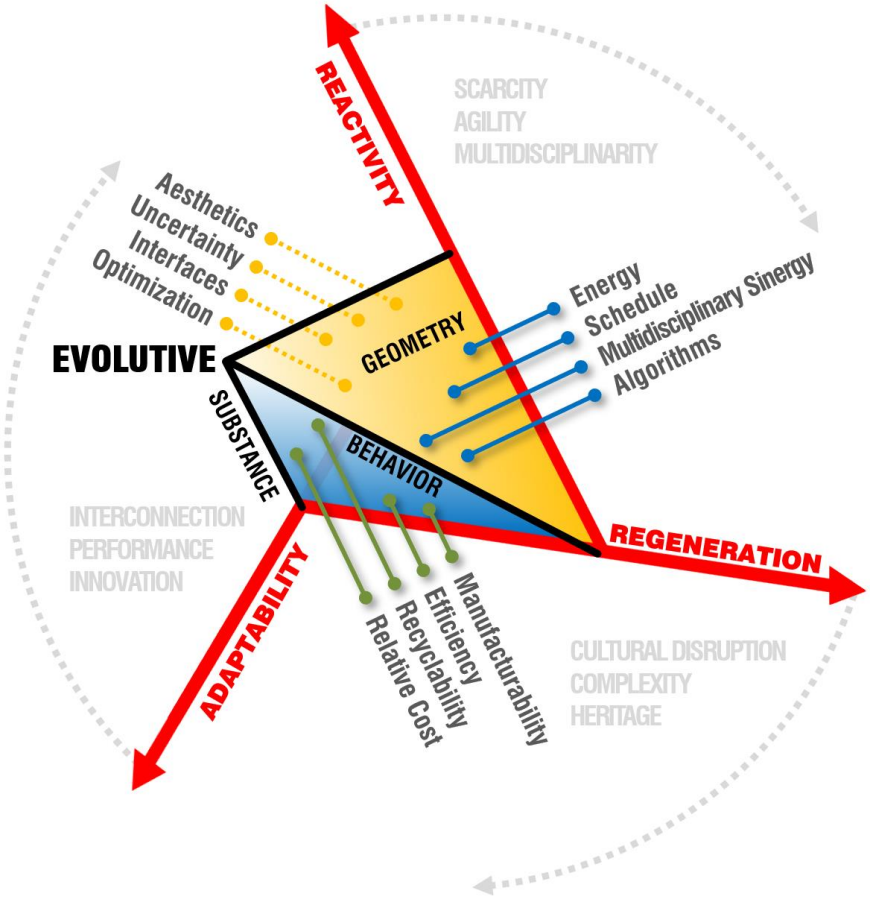



*Figure 193. Les moteurs de la conception du système comme faces du tétraèdre évolutif.*

À partir de la base décrite du tétraèdre évolutif, trois faces ou plans représentent ces groupes de moteurs de conception : [1] géométrie, [2] comportement, et [3] substance. Chacun de ces plans est opposé à sa clé de voûte plus directe (adaptabilité, réactivité et régénération), qui sont les vecteurs entre ces plans. Comprendre ces vecteurs tout en tenant compte des relations connues et inconnues entre eux est essentiel pour les concepteurs et les processus de conception afin de produire de bonnes architectures complexes. Les sections suivantes décrivent en détail ces facteurs de conception pour toute architecture de système évolutif donnée.

Les descriptions suivantes sont basées sur [1] l'étude des besoins généraux concernant les systèmes complexes, [2] les lacunes en matière de conception et d'ingénierie des systèmes identifiées dans la section 3, [3] les caractéristiques clés des systèmes évolutifs, adaptatifs et évolutifs, et enfin [2] près de deux décennies de pratique de la conception de systèmes complexes dans de multiples domaines techniques.

#### *8.4.1.5. Conclusion*

En réponse au chapitre 2, la section 4.1 a présenté les nouveaux facteurs de stress qui affecteront la pratique de l'ingénierie de conception de systèmes au cours des prochaines décennies, ainsi que les besoins subséquents de gérer une plus grande complexité dans ces efforts. L'étude des facteurs qui sous-tendent cette nouvelle réalité, ainsi que l'examen approfondi des techniques de pointe de l'ingénierie de conception et de l'ingénierie des systèmes, font apparaître plusieurs lacunes qui définissent en elles-mêmes un nouveau sous-ensemble croissant de systèmes complexes. Ces lacunes sont

les suivantes [1] la pertinence de la géométrie comme terrain d'entente entre les disciplines qui conditionnent la capacité du système, [2] la prise en compte des fonctions fondamentales ou des "comportements" de base du système en question qui n'est plus statique ou incapable de s'adapter, et [3] l'importance d'aborder le besoin et l'utilisation de toutes les ressources nécessaires pour produire, utiliser, gérer et réaffecter le système. Ces lacunes limitent non seulement le résultat du système mais aussi la méthodologie elle-même. En d'autres termes, les nouveaux besoins nécessitent de nouvelles méthodes et de nouvelles normes. C'est le point de départ de l'approche évolutive, qui répond au besoin d'adaptabilité dans la conception du système, ainsi qu'à la nature complémentaire et évolutive des méthodes inspirées de la nature qui permettent de relever ces défis à partir d'une approche rapide, axée sur les données, autoorganisée et multidisciplinaire.

Ces points sont souvent trouvés et combinés à travers l'intuition et l'instinct d'architectes et d'ingénieurs en chef talentueux dans de multiples domaines techniques et artistiques. Ainsi, cette recherche vise à créer une approche de base pour explorer le plein potentiel de ces approches afin de permettre la quantification, la qualification et, plus important encore, l'optimisation de nouvelles architectures, en particulier celles basées sur le matériel sans héritage ou générations précédentes.

Cela implique tout d'abord d'étudier la nature et les caractéristiques particulières des architectures de systèmes évolutifs dans le contexte des systèmes d'architectures complexes basées sur le matériel, et ensuite de développer une méthodologie pour permettre de tels systèmes et compenser les lacunes des techniques actuelles de pointe.

Sur la base des facteurs de stress généraux de plus en plus présents dans les scénarios de pénurie décrits au chapitre 2, et des lacunes dans les méthodologies de conception, l'approche évolutive présente trois clés de voûte constitutives des principes de base : adaptabilité, réactivité et régénération. Ces principes ont été décrits en détail dans la section 4.2. Ils caractérisent les architectures de systèmes évolutifs tout en fournissant les fondements de la méthodologie de conception ultérieure. Ces clés de voûte sont entrelacées (section 4.4) par une série de moteurs de conception synergiques qui cartographient le cycle complet des systèmes capables de réagir et de s'adapter à tout changement dans leur contexte et entre les composants. Ils ont été décrits et regroupés dans la section 4.3 autour de ces trois grands principes. En outre, ces moteurs traitent également de l'utilisation et de la gestion de toutes les ressources tout au long des phases de conception et du cycle de vie du système.

Bien que cela soit développé plus en détail au chapitre 5, la section 4.4 présente graphiquement en détail les relations entre ces moteurs dans un système de référence tridimensionnel. Un tel système de référence est basé sur la mesure des fonctions, l'utilisation des ressources et les interactions des systèmes, en réponse aux trois clés de voûte évolutives, ainsi qu'aux trois domaines décrivant tout système général dans ce contexte : géométrie, comportement et substance. Les architectures de systèmes évolutifs sont physiques, numériques, virtuelles, ou une combinaison de tous ces éléments. Il s'agit essentiellement de systèmes hautement adaptables, réactifs et durables ou régénératifs, comme on peut en trouver dans certains systèmes robotiques, architecturaux, aérospatiaux et organiques.

Par essence, les systèmes évolutifs s'inspirent de la nature et visent à atteindre le même niveau de performance, d'efficacité et d'adaptation. De plus, la façon dont cette nouvelle classe d'architectures est conçue infuse des principes de base éprouvés par des millénaires d'évolution naturelle sur la planète. Des forces simples mais très puissantes décrivent à la fois le système (produit) et la technique (méthodes), telles que : [1] les informations génétiques et patrimoniales qui déterminent l'adaptabilité et la sélection, [2] les mises en œuvre et les conceptions multifonctionnelles optimisées, [3] une approche continue du système en constante évolution, [4] la pertinence du contexte ou de l'environnement pour la conception du système, y compris les aspects culturels, techniques, physiques, numériques et virtuels, etc. Ces

Après cette description introductive des architectures de systèmes évolutifs, les sections suivantes développeront cette recherche en ce qui concerne les techniques et les méthodologies qui les permettent (chapitre 5), ainsi qu'un exemple simplifié (chapitre 6) qui les illustre. Bien que la phase de conception soit une étape clé dans un processus de développement en réseau, les phases d'optimisation et de mise en œuvre sont intimement liées et seront également brièvement présentées comme des perspectives à l'origine de cette nouvelle façon de voir un système basé sur le matériel, à l'ombre de la conception ultime du système, la vie naturelle.

## 8.5. Méthodologie de conception d'une architecture de système évolutive (chapitre 5)

Les architectures de systèmes évolutifs, telles que décrites précédemment au chapitre 4, sont un sous-ensemble de systèmes complexes réagissant à de multiples facteurs de stress externes sur la base des principes clés d'adaptabilité, de réactivité et de régénération. Ces principes sont fondés sur des approches adaptatives et évolutives, et ils comblent certaines lacunes importantes dans les techniques actuelles d'ingénierie et de conception de systèmes. En substance, l'outil conditionne le résultat et la manière de relever un défi. Cependant, pour tirer le meilleur parti de ces lacunes (sections 3.4 et 4.1), ainsi que pour fournir un moyen plus efficace de développer des architectures évolutives, une méthodologie ultérieure doit être créée. Ce chapitre présentera une approche vers un tel processus développée dans le cadre de cette recherche, la conception d'architecture de système évolutive (eSARD).

Cette méthode n'est ni fermée ni rigide. Elle présente une voie fondamentale qui pourrait et devrait être étendue et adaptée à tout besoin particulier requis par les concepteurs, les équipes, les machines, les flux de travail, les secteurs et les domaines industriels, parmi beaucoup d'autres. Ainsi, une approche de conception évolutive devrait être applicable à tout développement d'architecture de conception de système, indépendamment du domaine d'application. Cette méthode présente les objectifs généraux et interdépendants suivants :

- Développer une méthode d'ingénierie de conception efficace qui fournit des architectures de système évolutives matures sans héritage, qui couvre l'ensemble du cycle de vie de la conception, qui optimise le temps et les ressources, et qui permet la possibilité de solutions à pas de géant. En d'autres termes, il s'agit d'une méthode allégée permettant de trouver des solutions novatrices sans héritage.
- Jeter les bases d'une approche systémique de l'ES qui sert également de méthodologie de conception (ESD), et vers de nouvelles infusions de méthodologies d'aide informatique renforcées par des méthodes axées sur les données (par exemple, des flux de travail d'IA).
- Créer également les bases d'un schéma organisationnel et managérial, servant à la fois les approches DE et SE pour gérer le calendrier, les ressources et la main d'œuvre, ainsi que toute technologie et support machines nécessaires.

Les sections suivantes présenteront en détail le développement de cette méthode à travers ses objectifs, ses principes, sa base, ses flux de travail, ses outils et ses environnements. Cependant, cette recherche se concentre sur la partie fondamentale de la conception et de l'ingénierie des systèmes. Elle ne présente donc que des indications de base sur les aspects d'optimisation et de mise en œuvre des applications SE, ainsi que sur d'autres aspects organisationnels et managériaux de l'écosystème complet de la méthodologie évolutive.

### 8.5.1. Approche du processus de conception

Le développement de l'approche eSARD commence par le tétraèdre de conception évolutive qui caractérise le système d'architecture évolutive (Figure 127). En plus d'autres caractéristiques générales des systèmes complexes, les architectures de systèmes évolutifs présentent trois principes caractéristiques majeurs ou clés de voûte, comme expliqué au chapitre 4 : adaptabilité, régénération et réactivité (ARR). Cependant, ces principes généraux ne décrivent que des caractéristiques d'architecture de haut niveau, de sorte qu'un processus de conception doit aborder les trois domaines descriptifs du système, tels que la géométrie, le comportement et la substance (GBS). Enfin, en tant que méthode pratique, eSARD s'attaque également à l'échelle de tous les détails du système de conception, de mise en œuvre et d'exploitation (DIO).

Comme les points précédents l'ont présenté, le développement d'architectures de systèmes complexes en ce début de siècle est conditionné par la pénurie croissante potentielle de ressources due à de multiples facteurs et à des niveaux croissants de complexité des systèmes. L'équilibre entre les besoins et les ressources est en train de changer, exigeant souvent l'infusion et l'intégration d'outils nouveaux et perturbateurs qui complètent les méthodes plus traditionnelles. Du point de vue d'un quasi 4e révolution industrielle (Machado et Davim, 2020) aux nouveaux flux de travail collaboratifs homme-machine (Daugherty et Wilson, 2018), tout indique un changement de paradigme. De telles transitions se sont produites à un rythme beaucoup plus rapide dans les domaines des logiciels et des systèmes informatiques que dans les

développements de mise en œuvre matérielle (chapitre 3). C'est dans ce contexte que s'inscrit la présente recherche sur l'ingénierie de conception de systèmes, qui vise notamment à combler deux lacunes critiques dans la conception d'architectures complexes basées sur le matériel :

- Comment concevoir plus efficacement vers l'optimisation et la mise en œuvre d'architectures plus performantes présentant des caractéristiques évolutives ?
- Comment faire face au manque d'héritage et à la complexité croissante de la multidisciplinarité dans de tels processus ?

Une approche de conception de systèmes évolutifs commence par une perspective de cycle complet, qui aborde simultanément la conception, la mise en œuvre et les opérations afin de permettre des performances de système plus élevées et des architectures de niveau système plus efficaces en s'appuyant sur des connexions synergiques entre les disciplines et les sous-systèmes (figure 128). Cette méthodologie évolutive est particulièrement utile lorsque la conception est soumise à un manque important d'héritage (first-of-a-kind), à des contraintes de temps, ainsi qu'à un large éventail de sous-systèmes ou de technologies réalisables et pourtant nouveaux qui doivent être infusés pour la première fois.

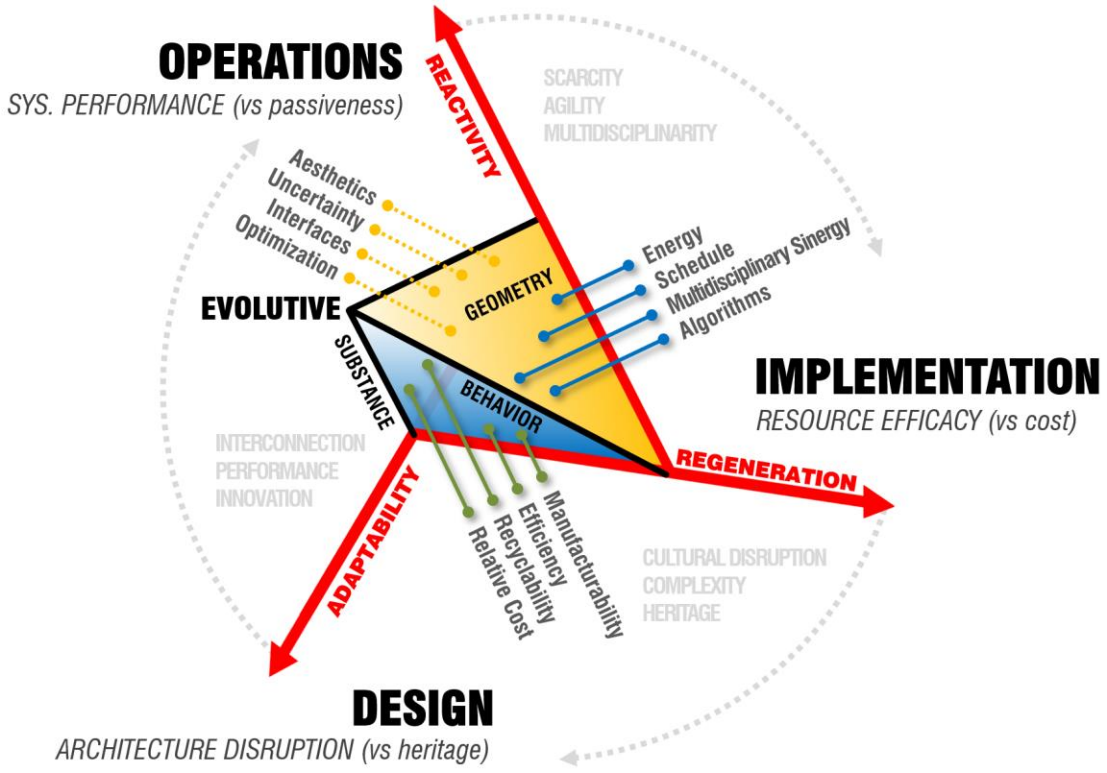



*Figure 194. Tétraèdre de conception évolutive définissant les phases clés de la méthodologie.*

Alors que le chapitre 3 a identifié les lacunes critiques dans l'état de l'art actuel des techniques DE et SE, le chapitre 4 a souligné et présenté en détail la caractérisation des systèmes d'architecture évolutive. Ainsi, cette thèse s'attaque aux lacunes et aux caractéristiques présentées par ces principes de conception d'architecture (le quoi), tandis qu'elle développe une méthodologie autour d'elle (le comment). Inspirée par la méthodologie de conception de la nature (Kliman, 2016) et la pratique plus holistique ou multidisciplinaire de l'architecture (Jarzombek et Prakash, 2011), cette approche applique des méthodes éprouvées, voire anciennes, à de nouveaux domaines de mise en œuvre.

Du point de vue des méthodes, cette approche a appliqué certains aspects de l'ingénierie des systèmes évolutifs (Braha et al., 2006) en informatique au domaine des implémentations matérielles. En tant que telle, plutôt que des méthodes linéaires et monodisciplinaires ou même parallèles, cette approche a un schéma axé sur le réseau, embrassant et combinant

à l'extrême les pratiques d'ingénierie simultanée et collaborative. En outre, cette méthodologie ne concerne pas seulement des disciplines quantifiables (par exemple, la conception mécanique) soutenues par des paramètres analytiques, mais aussi des sujets uniquement qualifiables (par exemple, l'esthétique) basés sur la conception géométrique, ainsi que des flux de travail d'exigences ouvertes ou changeantes.

Dans ce contexte, une approche évolutive (eSARD) ne se concentre pas sur des solutions de conception en un point unique. Elle aborde plutôt le développement de l'architecture du système à partir d'un schéma de solution continu, tout en traitant l'optimisation, la mise en œuvre (y compris la gestion) et les opérations ultérieures du point de vue de la géométrie, du comportement (fonctions) et de la substance (ressources) (GBS). Cette approche est agnostique par rapport aux applications et aux outils, et vise également à insuffler des niveaux plus élevés d'adaptabilité dans la méthodologie elle-même, tant du point de vue de la conception (géométrie) que des SE (abstraction).

Du point de vue du produit, de l'artefact et de l'architecture du système, un processus de conception de système évolutif vise à produire une architecture de système évolutive pouvant être mise en œuvre. Celle-ci présente plusieurs caractéristiques complémentaires par rapport aux systèmes plus traditionnels basés sur le matériel, telles que : [1] une grande adaptabilité du système, [2] une base réactive intelligente, et [3] une stratégie de ressources régénératrices ou durables. Du concept à la mise en œuvre, l'approche évolutive s'attaque aux écarts de maturité du système et de ses parties. Elle s'appuie ensuite sur les points communs et les synergies entre les disciplines, les sous-systèmes et les parties prenantes. La faisabilité et la capacité fonctionnelle de l'architecture en question déterminent l'approche, tout en gardant à l'esprit l'efficacité globale en termes de ressources, d'agilité et d'adaptabilité.

Cette méthodologie comble les lacunes des techniques d'ingénierie de conception (Pahl et al., 2007) et d'ingénierie des systèmes (INCOSE, 2020b) actuellement appliquées aux systèmes physiques et matériels. En substance, elle permet également d'établir une nouvelle base théorique pour faire mieux avec moins comme principe clé de la philosophie de conception. Par conséquent, le développement de ce processus est basé sur : [1] des analyses documentaires approfondies, [2] des recherches, des prototypes et des activités pratiques, et enfin [3] plusieurs décennies d'expérience professionnelle validée en tant qu'architecte et architecte système dans de multiples domaines industriels à travers le monde, dont près d'une décennie de pratique au Jet Propulsion Laboratory de la NASA pour le développement d'architectures de systèmes complexes. Néanmoins, cette approche est développée à partir d'une perspective de recherche fondamentale, elle est donc complètement agnostique par rapport aux outils, au domaine d'application et à toute technologie spécifique. En résumé, il s'agit d'une approche universelle de la conception de systèmes (DSE).

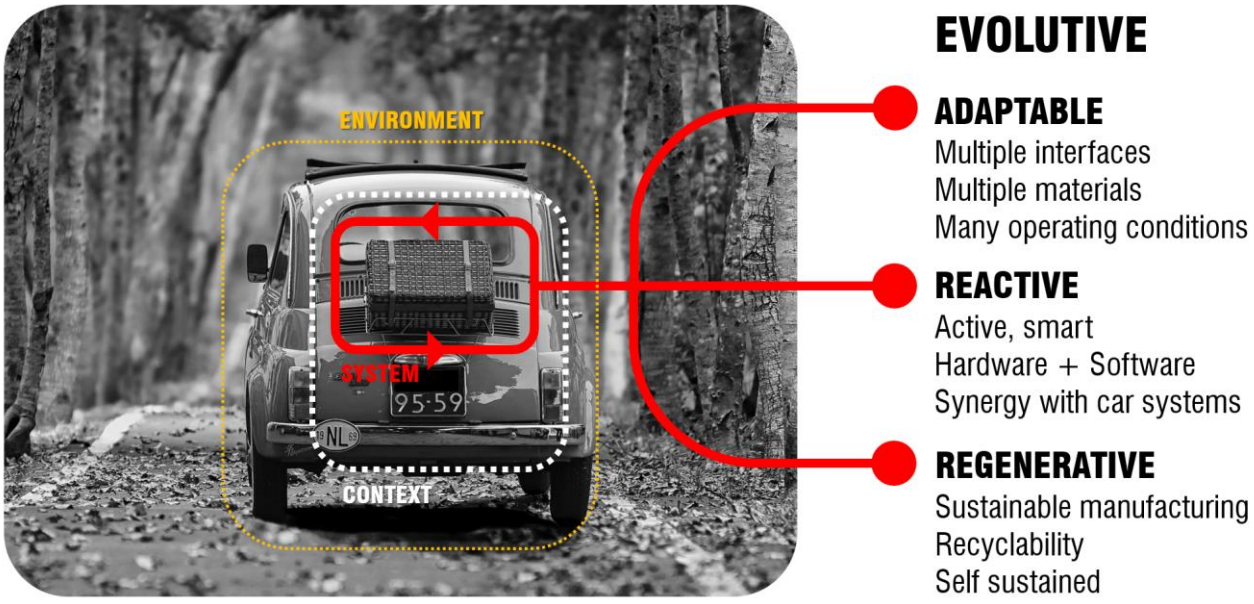



*Figure 195. Exemples de l'approche eSARD appliquée à un composant additionnel pour une conception de voiture existante.*

Le développement d'une pièce complémentaire externe pour la carrosserie d'un modèle de voiture existant pourrait être un bon exemple de cette approche. Il pourrait s'agir, par exemple, d'un support de bagage supplémentaire pour le coffre (figure 129). La pièce elle-même n'a pas beaucoup d'héritage puisqu'elle est assez unique et ne fait pas partie de la conception originale. Cependant, elle nécessite d'accroître ses performances par rapport aux solutions précédentes en raison de l'augmentation de la tolérance à la vitesse, des normes de confort et de la protection de l'environnement. En outre, une approche évolutive appliquée à ce problème prendrait en compte les points suivants :

- **Adaptabilité (géométrie**). Le composant doit s'adapter passivement à différents paramètres de conduite, conditions environnementales et interfaces mécaniques. De multiples options de finition et de matériaux feraient partie de l'espace commercial.
- **Réactivité (comportement).** Cet aspect pourrait permettre l'éclairage et le contrôle aérodynamique actif. Il devrait également pouvoir être suivi par GPS en cas de perte, de vol ou de chute de la voiture. Ainsi, les batteries, les capteurs et les composants actifs sont intégrés.
- **Régénération (substance).** La fabrication et le cycle de vie complet du système doivent être entièrement durables.

L'entreprise qui développe cette pièce pourrait ne vouloir répondre qu'aux exigences d'un modèle spécifique. Toutefois, lorsqu'une approche évolutive est appliquée, le concepteur et le flux de travail de conception doivent s'adapter à davantage de choses. Ainsi, au lieu que la conception soit uniquement applicable à un seul cas, elle est réalisée en tenant compte de nombreuses autres contraintes possibles probables ou réalisables afin de trouver des solutions plus synergiques et optimisées. Il ne s'agit pas d'une contrainte excessive, mais d'un effort pour trouver une meilleure solution.

### 8.5.2. Modèle de l'hélice de conception évolutive

Le chapitre 4 a exposé des principes rationnels et fondamentaux concernant la définition d'une architecture de système évolutive ou eSAR. En outre, le chapitre 3 a identifié les lacunes des techniques de pointe en matière de SE et de DE, y compris celles qui appliquent des principes évolutifs. Enfin, le chapitre précédent a présenté ce que la méthodologie de conception évolutive doit poursuivre en termes d'objectifs et de principes dans une perspective combinée d'ED et de SE.

L'étape suivante consiste donc à développer le processus eSARD, qui présente une nature en réseau abordant les trois nœuds d'activité clés tels que [1] la conception du système, [2] l'optimisation opérationnelle du système et [1] la mise en œuvre du système (DOI). Le premier nœud de cette méthode représente l'objectif principal de cette recherche, mais tous les nœuds sont intimement liés puisqu'ils doivent être développés simultanément. Cette section définit les détails spécifiques du flux de travail eSARD, y compris le cadre, les jalons, les outils, la dynamique et les routines pour une méthodologie d'ingénierie des systèmes de conception (DSE) hautement adaptable.

La figure 144 présente une description graphique du modèle hélicoïdal eSARD (eSARD_he) où trois secteurs (triangles) sont répartis sur un schéma en spirale décrivant différentes portes de conception (jalons), routines (outils) et chemins utilisés pour couvrir les trois domaines ARR d'un système (adaptabilité, réactivité et régénération). La représentation d'un système unique à un moment donné est présentée par la spirale de la figure 145. Les vecteurs ARR (flèches rouges) créent la structure définissant les secteurs DOI, les phases, les jalons, etc. pour tout système donné. Les itérations multiples au sein d'un même système sont représentées par des translations dans le secteur, de manière similaire aux techniques SE incrémentales ou IID (section 0).

Si l'on considère ces changements continus dans de multiples instances au sein d'une espèce commune (conception continue évolutive), l'évolution du système pourrait être représentée comme une hélice tridimensionnelle, comme le montre la figure 145. Cette hélice peut également être référencée dans le cadre évolutif décrit à la section 5.4 (figure 134). Dans ce cadre, toute conception ponctuelle (par exemple, le violet B) représente une famille d'instances ou de solutions. Le processus de la spirale 2D d'eSARD (Figure 144) peut être appliqué à chacun de ces points, avec pour objectif de décrire, développer et gérer une nouvelle solution qui est référencée dans ce système de coordonnées. Ainsi, le graphique en spirale 2D représente le processus au niveau de la solution, tandis que l'hélice 3D représente le travail au niveau de l'espèce, et les deux échelles peuvent être comparées et connectées dans ce cadre évolutif. Les spirales 2D et 3D sont des processus en réseau, de sorte qu'à l'intérieur de ceux-ci, toutes les variables, étapes, routines et outils connexes sont liés tout au long de leur flux de travail. Par exemple, les validations de la fabrication au niveau de la mise en œuvre peuvent également conditionner les écarts de maturité de la conception et vice versa. Comme tout se passe simultanément, il est essentiel de

disposer de données correctes. Le flux de jalons aide à guider et à valider le processus, tout en permettant l'infusion de méthodes et d'outils traditionnels dans des organisations plus axées sur le patrimoine.

La nature itérative des approches de conception plus traditionnelles (Johnson et Gibson, 2014) fait également partie de l'activité dans chacun de ces secteurs, ainsi que de toutes les interactions entre eux. Cependant, cette méthode inclut d'autres aspects du cycle de vie tels que la durabilité ou le déclassement. Les clés de voûte de l'ARR (adaptabilité, interactivité et régénération) sont directement liées aux trois secteurs décrivant une architecture de système évolutive : géométrie, comportement et substance (GBS). La méthode qui sous-tend cette approche est liée à d'autres théories de conception, comme le présente la section 4.1 sur la base des conclusions du chapitre 3. Il s'agit notamment des techniques SE et DE telles que le DE systématique (énergie, matériau, signal - Pahl et al., 2007), le FBS (fonction, comportement et structure - Gero et Kannengiesser, 2004), et d'autres méthodes évolutives présentées dans le tableau 17. Néanmoins, cette approche ne vise pas à créer un simple processus de conception général, mais plutôt à créer une méthode holistique et adaptable qui aborde toutes les caractéristiques particulières des architectures de systèmes évolutifs basés sur le matériel tout au long de leur cycle de vie.

Dans le cadre de ce flux de travail, les données patrimoniales pertinentes deviennent également un aspect essentiel servant de validation, de contexte et de comparaison pour les nouveaux systèmes. Toutefois, ces données ne peuvent jamais limiter les efforts de conception. Ainsi, le patrimoine est évalué de manière indépendante puisque l'objectif d'une méthode eSARD est de développer une solution d'architecture évolutive qui dépasse très probablement les performances des systèmes précédents tout en devenant souvent la première du genre. La section suivante présente en détail ces étapes de conception.

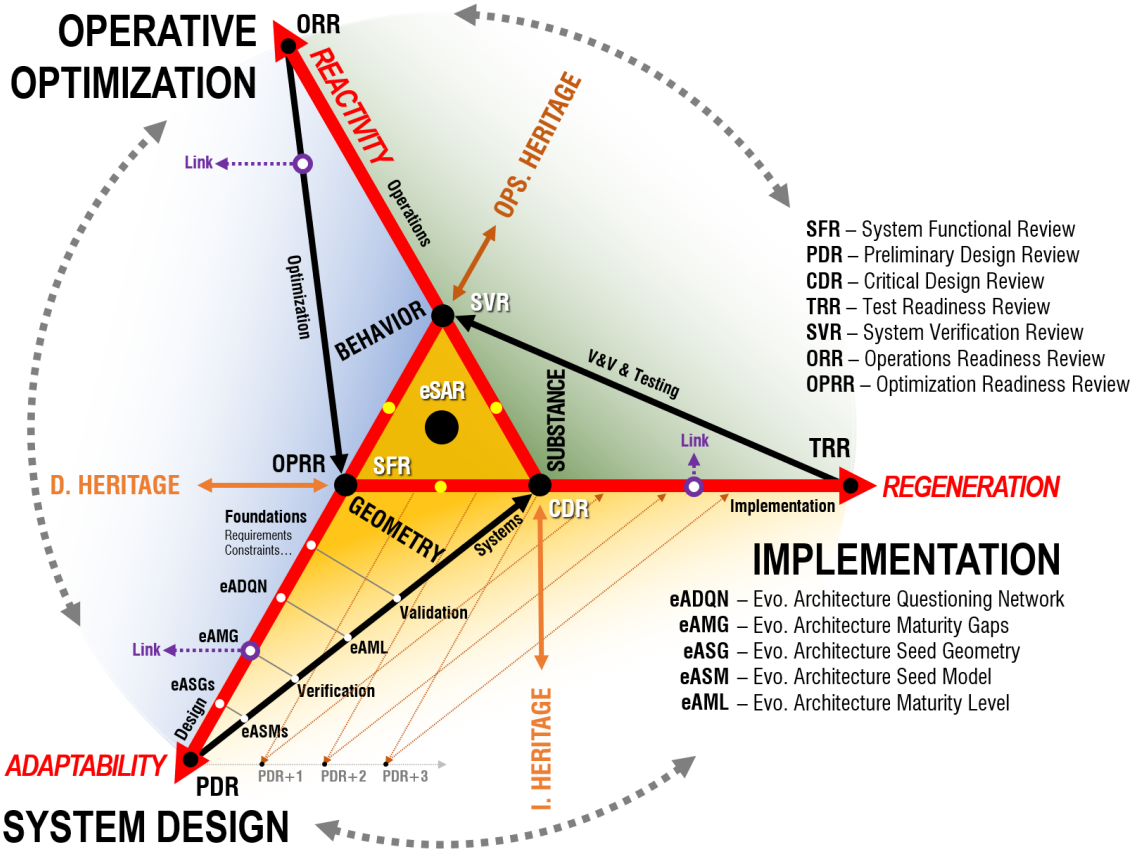



*Figure 196. Processus évolutif de conception de systèmes en réseau, présentant les trois nœuds d'activité ARR traitant de la conception, des opérations et de la mise en œuvre.*

### 8.5.3. Conclusion (chapitre 5)

Les méthodologies de conception actuelles sont principalement basées sur une approche "diviser pour mieux régner", mettant en œuvre une approche de conception de système "en ligne" ou séquentielle. Bien que de nombreuses méthodes prennent en compte plusieurs disciplines, elles ne sont généralement pas développées d'un point de vue intégré. En réponse aux caractéristiques particulières des architectures de systèmes évolutifs présentées au chapitre 4 et résumées sous les principes d'adaptabilité, de réactivité et de régénération (ARR), l'approche eSARD est développée pour aborder à la fois le processus global et les principes, objectifs et outils clés qui sous-tendent sa méthodologie évolutive. Cette méthode s'inspire des mécanismes évolutifs (section 5.1) et de certaines lacunes des techniques de pointe en matière d'ED et de SE (chapitre 4) pour créer une approche plus efficace, plus rapide et plus performante qui répond aux besoins spécifiques des architectures de systèmes eSAR (section 5.2). En outre, cette méthodologie intègre des outils et des méthodes provenant d'autres techniques de DES, tout en créant et en modifiant des outils existants pour fournir un flux de travail dynamique et hautement adaptable abordant le processus de conception du système à tous les niveaux.

Les principes de base de l'ARR fournissent un cadre de référence pour la méthode eSARD qui est basée sur le tétraèdre de conception évolutive. Cette construction permet d'aborder le flux de travail de conception à différents niveaux, notamment [1] les caractéristiques de haut niveau de l'architecture du système (ARR) décrivant les besoins et les capacités spécifiques des systèmes eSAR, [2] les détails du système de géométrie, de comportement et de substance (GBS) où un flux de travail de conception basé sur le matériel définit, conceptualise et met en œuvre tout système, et enfin [3] le flux de travail de conception détaillé associé à toutes ces échelles en considérant les sujets du système de conception, de mise en œuvre et d'exploitation (DOI) nécessaires pour mettre en œuvre complètement toute conception de l'architecture du système. Ainsi, cette méthode s'adresse à un système qui est développé à travers les échelles et les phases du cycle de vie comme le présentent la section 5.2 en général et la section 5.3 en particulier. Le processus eSARD introduit une approche holistique qui aborde le processus complet du cycle de vie de la conception en traitant toutes les échelles du système de manière simultanée, synergique et efficace. Cette approche est également basée sur les méthodes actuelles et sur de nouveaux outils spécifiquement conçus pour les architectures de systèmes évolutifs (tableau 25).

La section 5.4 a présenté les objectifs de conception les plus critiques d'eSARD dans le cadre de référence évolutif et autour des domaines de la conception adaptable, de la performance des systèmes réactifs et enfin de l'utilisation régénérative des ressources. Ces objectifs conduisent à une approche globale qui minimise les coûts, prend en compte l'influence culturelle, permet de multiples voies de conception simultanées pour finalement fournir suffisamment de détails pour faire d'eSARD une méthodologie compétitive (section 5.4.4).

Les objectifs de conception constituent la base des principes de conception clés (section 5.5) utilisés dans l'ensemble de la méthodologie eSARD pour guider tous les efforts et activités de conception. Ces principes sont les suivants [1] "faire mieux avec moins", en résolvant un problème par la conception (adaptabilité, section 5.5.1), [2] concevoir "plus intelligemment avec moins", par des solutions et des opérations continues (réactivité, section -), et [3] faire "plus avec moins", en optimisant l'utilisation de toutes les ressources tout au long du cycle de vie de la conception et du système (régénération, section 5.5.3).

Une fois que tous les aspects fondamentaux de l'approche eSARD ont été exposés, ce chapitre a présenté le modèle et le diagramme subséquent utilisés pour décrire, organiser, gérer et mettre en œuvre toutes les multiples étapes, activités, jalons, produits et outils utilisés simultanément dans un flux de travail eSARD (section 5.6). Il s'agit du modèle d'hélice eSARD ou eSARD_he (Figure 144). Ce modèle fournit une représentation simplifiée en deux dimensions d'une activité et d'un flux de travail de conception fortement mis en réseau, simultané et, à certains égards, tridimensionnel (section 5.6.1). Ce modèle est organisé autour de trois secteurs basés sur tous les domaines de la DOI (conception, mise en œuvre et opérations) qui se déroulent toujours simultanément. Chaque secteur est par essence un modèle Vee, les trois secteurs étant intégrés autour d'une spirale qui décrit le cycle complet du système. Ce diagramme présente une série de jalons ou de portes de conception dans chaque sommet, qui ne cessent d'accroître la maturité et la définition du système à travers les phases du cycle de vie et les secteurs de développement (GBS). Les bords de ces secteurs abordent des domaines de conception critiques et interconnectés, tels que : la conception, l'ingénierie des systèmes, la mise en œuvre, la vérification et les tests, les opérations et l'optimisation complète du système. Dans chaque secteur, il existe une série d'outils et

d'activités utilisés pour créer les produits nécessaires à l'examen critique et aux étapes clés, mais surtout pour aborder, étudier et développer la conception de l'architecture du système en question. La section 5.6.1 présente tous les éléments clés de ce diagramme, tandis que la section 5.6.2 décrit en détail le premier secteur de conception du modèle et les opérations qui s'y déroulent. Enfin, les sections 5.6.3 et 5.6.4 présentent d'autres sujets concernant le secteur de la mise en œuvre et le secteur opérationnel, respectivement. En outre, ce niveau d'interconnexion nécessite des outils pour évaluer l'état de développement entre les secteurs, ce qui est fait par le biais de boucles de vérification qui ont été décrites dans la section 5.6.5. Enfin, une telle approche nécessite non seulement un modèle pour décrire les opérations, mais aussi un cadre d'espace de travail qui permet une utilisation et une infusion efficaces de la méthode sur le plan physique (équipes), virtuel (collaboration) et numérique (données). C'est le cadre évolutif 3C (concurrent, collaboratif et communicatif) décrit dans la section 5.6.6.

Dans le cadre du développement de la méthodologie eSARD, ce chapitre a également introduit une description de son flux de conception ultérieur (section 5.7) et a présenté en détail tous les outils clés de conception évolutive utilisés dans le premier secteur. Cette approche est basée sur l'identification des lacunes les plus pertinentes et les plus synergiques dans la conception d'un système par le biais d'une série de questions ou eADQNs (section 5.8) sélectionnées et hautement multidisciplinaires. Ces questions portent sur les aspects qui conditionnent la faisabilité, l'efficacité et la mise en œuvre d'un concept de système évolutif. La plus critique d'entre elles devient une lacune de maturation du système ou eAMG (section 5.9) qui définit le point de départ du processus de conception eSARD. À partir de là, les premières solutions commencent à se former grâce à une série de modèles et d'outils de conception ultérieurs appelés géométries d'amorçage ou eASG (section 5.10). Ces conceptions conduisent ensuite à la création de modèles de système ou eASM (section 0) à l'aide d'une série d'outils SE modifiés qui amènent la solution à un certain niveau de maturité et de définition, mesuré par des niveaux de maturité évolutifs ou eAML (section 5.12). Comme pour le TRLS et le CMLS, ces niveaux permettent d'organiser, de comparer et d'évaluer les solutions de conception qui sont développées dans le cadre d'une série de cycles de conception rapides et synchrones (section 0). Enfin, ce chapitre fournit une série de mesures et de principes de comparaison pour évaluer l'avancement de la conception et comparer les efforts entre les techniques, les cadres et la solution système (section 5.14).

Grâce à une approche multidisciplinaire, cette activité de recherche et la pratique associée complètent les tendances actuelles de l'état de l'art en matière de méthodologie de conception et d'ingénierie des systèmes tout en ouvrant une voie nouvelle et complémentaire, notamment vers les architectures de systèmes complexes basés sur le matériel. Néanmoins, cette approche est indépendante du domaine technique d'application et considère les liens entre les perspectives basées sur le matériel et le logiciel.

## 8.6. Cas d'étude : Architecture évolutive de micro-habitat (chapitre 6)

Ce chapitre applique plusieurs des méthodes et outils expliqués dans les sections précédentes à un cas d'étude générique. Le sujet de cette étude est un système matériel, qui nécessite la prise en compte de plusieurs sujets évolutifs dans les domaines de l'adaptabilité, de la réactivité et de la régénération. L'objectif de ce chapitre n'est pas de présenter une solution entièrement détaillée pour le défi de conception en question, mais plutôt de présenter les aspects clés du flux de travail et de la méthodologie eSARD. En outre, le chapitre souligne les principales différences avec d'autres méthodes, ainsi que certaines mesures potentielles à utiliser à différentes phases du processus. Le processus eSARD utilisé ici n'est pas un processus rigide, cette recherche ne présente donc qu'une base qui peut être ultérieurement modifiée, personnalisée, améliorée, réduite ou complétée par d'autres méthodologies en fonction de l'application.

### 8.6.1. Le domaine de l'architecture des micro-habitats

La conception de l'architecture des habitats en général est un domaine complexe et potentiellement très évolutif pour les systèmes complexes basés sur le matériel. Parmi les nombreuses applications et pratiques de conception figurent les bâtiments, les maisons, les habitats techniques, les abris, etc. Parmi eux, il existe un sous-ensemble spécifique d'architectures de systèmes qui est très intéressant à titre d'exemple, les habitats à petite échelle ou micro-habitats (Horden, 2010). Ces petits bâtiments comprennent des abris hors réseau, des avant-postes de recherche, des retraites en montagne,

des points de vue, des abris d'urgence, des cabanes dans les arbres et des maisons de jeux, entre autres. Toutefois, ce domaine est complexe, multidisciplinaire et assez spécialisé (figure 174). Il présente également un long héritage au fil des siècles en ce qui concerne la fonctionnalité, la mise en œuvre et les approches de conception. Parmi leurs nombreuses caractéristiques génériques, les suivantes sont assez communes dans ce domaine :

- Taille compacte. Ces habitats ont tendance à être très petits, avec une conception et une structure modulaire.
- Transportabilité. Ces systèmes d'architecture sont souvent portables, transportables et préfabriqués.
- Matériaux avancés. En raison de leur application particulière, de leur taille et de leur nature expérimentale, ces habitats présentent de nouveaux matériaux et de nouvelles techniques de fabrication qui ne sont pas souvent utilisés ou envisagés dans d'autres constructions de plus grande taille.
- Masse légère. En raison de tous les points précédents, ces constructions ont tendance à être très légères, car elles font appel à de multiples techniques peu communes et de haute technologie, telles que les structures gonflables, les panneaux légers, les tensegrities, les nouveaux schémas structurels, etc.

Par conséquent, tous ces micro-habitats en général, et plus particulièrement ces abris techniques nécessitant des capacités hors réseau et une grande portabilité, sont liés à des caractéristiques évolutives de base (ARR), comme le résument les points suivants :

- **Adaptabilité**. Ces habitats doivent pouvoir s'adapter aux multiples changements de conditions météorologiques, aux différents besoins des utilisateurs et aux schémas d'utilisation pendant toute la durée de vie du système. En outre, les architectures de ces systèmes doivent répondre aux problèmes de disponibilité des matériaux de construction, de connaissances et de compétences de la main-d'œuvre, de faisabilité du transport, et bien d'autres problèmes encore.
- **Réactivité**. Les systèmes d'habitation de ce type, même s'ils sont temporaires, doivent réagir au niveau le plus élémentaire à toutes les conditions environnementales. Par exemple, ils doivent fournir et conserver la chaleur par temps froid, fournir des mécanismes de refroidissement dans les climats chauds, gérer les espaces ouverts et fermés en fonction des conditions d'utilisation, etc.
- **Régénération**. Les habitats durables hors réseau doivent également produire de l'énergie et gérer les déchets dans de multiples conditions. Si les performances et les capacités du système sont susceptibles de changer, les conséquences de ces changements affecteront l'ensemble du système.

En outre, le domaine des micro-habitats fait partie d'un domaine architectural plus large, celui des systèmes d'architecture d'habitation, qui est en phase avec tous les facteurs de stress de la conception décrits au chapitre 2, tels que : le changement climatique, les besoins énergétiques, la complexité, la nécessité d'améliorer les performances, la pression du patrimoine et la raréfaction des ressources due à la croissance démographique, entre autres. Ainsi, ce domaine des micro-habitats constitue un parfait terrain d'étude en raison de la complexité de sa nature et de son contexte d'exploitation.

### 8.6.2. Conclusion (chapitre 6)

L'exemple choisi pour cette étude a montré le besoin de plusieurs des caractéristiques clés qui définissent une architecture de système évolutive (eSAR). Ceci a été fait après un premier regard à travers la perspective d'une approche eSARD. Dès le début, les domaines clés du système ARR (adaptabilité, réactivité et régénération) ont été abordés car ils nécessitaient une conception et une analyse pour une solution système réussie. Cependant, cette approche a également mis en évidence la nécessité d'évaluer d'autres solutions patrimoniales directes et indirectes, ainsi que les avantages d'une exploration plus approfondie de toutes les exigences initiales.

Ainsi, la première étape dans cet exemple a consisté à mettre en place un environnement évolutif 3C, abordant des sujets collaboratifs, communicatifs et simultanés qui comprenaient des outils, des agents et des connexions préliminaires, entre autres modèles. Cela a non seulement donné le ton au flux de travail du processus de conception, mais a également accéléré toutes les activités de travail ultérieures. Ainsi, tant les outils que l'environnement dans lequel ils fonctionnent ont nécessité une certaine conception au-delà de la solution système elle-même, ce qui a fait une grande différence. C'est un aspect pertinent de l'approche eSARD qui diffère des autres méthodes. L'analyse des solutions patrimoniales d'un point de vue des systèmes, des composants et des technologies a permis de créer un ensemble d'informations très utiles à utiliser ultérieurement lors de toute activité de conception dans les multiples phases et secteurs du processus.

L'approche eSARD (section 6.4.2) a abordé l'activité de conception dans cet exemple d'un point de vue combiné impliquant tous les domaines du DOI (conception, opérations et mise en œuvre). Bien que cet exemple n'ait abordé que le premier, le tableau 35 présente le déroulement complet de l'activité. Une fois l'environnement et les études patrimoniales étudiés, le processus eSARD a présenté plusieurs phases fournissant des conclusions clés qui sont résumées dans les points suivants :

- **Exigences, DOI et principes GBS (section 6.4.3).** L'étude de ces sujets dans le cadre de l'approche eSARD apporte une perspective très holistique et interconnectée pour évaluer un défi de conception (Tableau 36). Cela a conduit à la création d'exigences secondaires issues de cette étude, qui n'étaient pas initialement fournies par le client mais qui sont devenues critiques pour trouver une solution réalisable. La solution envisagée ici n'est pas destinée à un seul EPH, mais plutôt à une famille d'EPH en tenant compte d'éventuelles mises à niveau et modifications futures. L'application d'une telle contrainte de conception a permis de simplifier les solutions et de découvrir des voies de conception qui réduiront les étapes de fabrication, les coûts et la masse ultérieurement.
- **Les techniques de questionnement dynamique utilisant les eADQN (section 6.5)** permettent de trouver la question DOI la plus critique conditionnant la faisabilité du système et le flux de travail de conception en général (figure 178). Dans ce cas, la compacité du système, les opérations manuelles et les améliorations environnementales possibles se sont avérées être les aspects les plus critiques pour cette entreprise. Ces outils ont permis d'évaluer et de renforcer l'importance de l'adaptabilité du système pour sa réussite.
- **Écarts de maturité (eAMG).** L'utilisation des techniques eAMG (section 6.6) a permis d'identifier l'écart le plus critique pour cette conception, qui est également le sujet le plus pertinent intégrant des disciplines clés, telles que l'ingénierie mécanique (mécanismes), la gestion thermique, et les sujets de l'expérience utilisateur : le style de conditionnement et l'aménagement intérieur. Ainsi, la compactibilité de la géométrie de l'habitat était un écart de maturation clé (tableau 37) et plus important que la réduction de la masse ou le coût. Sans réponse à cette lacune, la solution n'est pas réalisable, mais en même temps, résoudre ce problème d'un point de vue multidisciplinaire permet automatiquement d'aborder trois disciplines majeures. Ainsi, un sujet synergique intègre de multiples points de vue, et se concentrer sur lui permet de trouver des solutions meilleures et plus complètes beaucoup plus rapidement.
- **Géométries de semences évolutives (eASG).** Un tel écart dans l'effort de conception (section 0) a permis de former une géométrie conceptuelle (figure 180) abordant simultanément plusieurs voies de conception et questions disciplinaires. Ces géométries ont pris en compte des sujets mécaniques, la performance thermique, les techniques de fabrication, les sujets SE et le style, entre autres.
- **Modèles de systèmes évolutifs (eASM).** Enfin, la dernière étape dans le secteur de la conception DOI est la création d'un modèle de système initial qui capture les paramètres quantifiables et qualifiables, tout en créant des liens avec les modèles eASG (section 6.8). Dans ce cas, l'outil choisi pour capturer le système était un eGBSEL évolutif. Ce modèle basé sur un tableur est renforcé par des hyperliens et couvre tous les domaines de développement du système ARR et GBS, permettant de créer une vue beaucoup plus large et détaillée de l'eSAR en question. Il servira de base aux études paramétriques ultérieures.

En substance, l'utilisation d'une méthodologie eSARD a permis de trouver une solution très efficace en termes de calendrier (temps), d'outils et d'agents (section 6.7.2). Cependant, l'essentiel est que cette approche a permis de mieux aborder l'activité de conception d'un point de vue conceptuel, logique, pratique et programmatique. Alors que les prochaines étapes et phases (section 6.9) apporteront certainement toute la puissance de cette approche à d'autres domaines du DOI tels que la mise en œuvre et les opérations, toutes les améliorations et les efficacités fournies par cette technique sont mises en évidence par le produit et le processus.

# 8.7. Conclusion : Pistes de conception de systèmes d'architecture évolutive (chapitre 7)

## 8.7.1. Discussion

Le point de départ de cette recherche était les systèmes complexes basés sur le matériel (CHS) en général, et un sous-ensemble de ceux-ci définis comme des architectures de systèmes évolutifs (eSAR). En outre, les méthodologies de conception et d'ingénierie des systèmes permettant de tels systèmes faisaient partie des objectifs de la recherche. Les chapitres précédents ont élaboré le chemin utilisé tout au long de cette thèse pour répondre aux trois questions de recherche initiales. Les réponses et les discussions à ces questions sont résumées dans les points suivants.

- **Quelles sont les nouvelles caractéristiques et les besoins de conception complémentaires que présentent ces systèmes ultra-complexes dans des environnements où les ressources sont limitées ?**
  - L'analyse globale des tendances mondiales présentée au chapitre 2 a montré que plusieurs facteurs de stress liés à la conception influencent la pratique des SE et de l'ED pour développer des systèmes complexes (SHC). Ces facteurs affectent l'équilibre de base entre les besoins des systèmes et les ressources disponibles (figure 6) et ne cessent d'imposer le besoin d'adaptabilité dans toute approche robuste qui s'y attaque. Ainsi, de nouvelles situations exigent de nouvelles approches. Les facteurs de stress les plus importants sont les suivants.
    - La rareté des ressources, telles que l'énergie, les matériaux et même la main-d'œuvre, est déterminée par le climat, l'économie et la compétitivité, entre autres raisons, ce qui oblige les systèmes à devenir plus efficaces et à utiliser moins de ressources.
    - La complexité des systèmes s'accroît, en particulier lorsque les systèmes matériels (CHS) continuent d'être enrichis de logiciels, de données et d'autres capacités interactives (par exemple, la robotique appliquée aux produits de consommation).
    - De meilleures performances des systèmes doivent être obtenues à des vitesses plus rapides en raison des contraintes du marché.
    - La multidisciplinarité croissante de la conception des systèmes et de leur gestion doit être prise en compte de nos jours.
    - L'agilité des processus du système et la capacité de réutilisation de l'effort de travail sont des facteurs critiques pour le coût.
    - Les systèmes complexes (SHC) sont de plus en plus en réseau et présentent davantage de liens entre les sous-systèmes et leur propre environnement contextuel, qu'il soit physique, numérique, virtuel ou une combinaison de ces éléments.
    - L'héritage de la conception technique influence progressivement l'équilibre entre les nouvelles solutions et la posture de risque.
    - L'innovation dans les nouveaux systèmes évolue vers la perturbation plutôt que vers des changements partiels ou incrémentaux.
    - Les perturbations culturelles, les nouveaux outils et les flux de travail affectent les activités de conception tant au niveau des produits que des processus.
  - Dans ce contexte changeant, les architectures de systèmes évolutifs sont identifiées comme un sous-ensemble de systèmes basés sur le matériel (CHS) en réponse aux facteurs de stress actuels et futurs de la conception (chapitre 4). Ces systèmes hautement complexes et indépendants du terrain sont déterminés par les facteurs de stress précédents et les nouveaux besoins de conception. Ils présentent trois caractéristiques clés ou ARR, qui deviennent également le fondement d'une méthodologie de conception ultérieure.
    - **Adaptabilité.** Toute architecture de système est conçue comme un processus évolutif continu dans lequel les définitions géométriques et analytiques du système (quantifiables ou non) ne cessent de changer et d'évoluer. Toute solution antérieure (héritage) peut devenir un élément de construction validé. Globalement,

la géométrie du système est toujours complétée par des capacités fonctionnelles du système et des schémas de mise en œuvre concernant l'utilisation des ressources.
    - **Réactivité**. Un système évolutif n'est pas passif et interagit avec son contexte environnemental et ses composants. Ainsi, outre la géométrie du système, ses descriptions fonctionnelles et logiques sont essentielles pour la complétude comportementale du système (réactions), tout comme pour son adaptabilité et les processus associés.
    - **Régénération**. L'utilisation, le recyclage, la réaffectation et la régénération de toutes les ressources utilisées par le système et son processus de conception font également partie de son processus de développement qui repousse les limites de sa durabilité.
- Des situations nouvelles et changeantes définissent de nouveaux besoins pour les systèmes et méthodes à venir. Les systèmes complexes exigent de plus en plus la combinaison des pratiques de l'ingénierie de conception et de l'ingénierie des systèmes, ce qui apporte une réponse à cette question à la lumière des principes évolutifs de l'ARR. Il s'agit également d'un début pour d'autres questions de recherche, notamment en ce qui concerne les interactions entre le matériel et les logiciels, et leurs combinaisons virtuelles.

- **Quels principes pourraient améliorer les processus de conception et d'ingénierie des systèmes traditionnellement séquentiels afin de réaliser plus rapidement, mieux et plus efficacement de tels systèmes complexes multidisciplinaires ?**
    - Un examen approfondi de la littérature a présenté les principales lacunes pertinentes et liées entre elles dans les méthodes de DE et de SE pour le développement de SHC lorsqu'elles sont étudiées d'un point de vue géométrique et analytique (chapitre 3), telles que :
        - La synergie. Les méthodes d'ED ont tendance à aborder plusieurs disciplines de manière séquentielle ou parallèle.
        - Continuité. Les méthodes DSE sont axées sur la conception de points ou de solutions discrètes plutôt que sur des familles de solutions (espèces).
        - La qualification des paramètres est largement répartie entre elles mais les aspects qualifiables sont difficiles ou impossibles.
        - Géométrie. Les méthodologies SE ne traitent pas bien les informations géométriques, et les processus SE se débattent avec des définitions de systèmes complexes qui pourraient être personnalisés ou améliorés.
        - Le cycle de vie complet du système est souvent absent des méthodes.
        - La flexibilité des méthodes d'ESD est souvent très complexe ou absente. Les outils ont tendance à retravailler les solutions, souvent pour des raisons culturelles.
        - Les perturbations ne sont souvent pas encouragées par la méthode elle-même.
        - Les flux de travail rapides sont souvent un défi pour tous les efforts.
        - Les liens de connectivité du système ont tendance à manquer dans les efforts de l'ESD.
    - Certaines de ces lacunes ont été comblées par les techniques évolutionnistes (section 0) depuis le début des années 1950. Celles-ci étaient inspirées par la nature et principalement appliquées à l'informatique plutôt qu'au SHC et à d'autres applications axées sur le matériel. Parmi les principes et contributions clés (section 4.1.1), plusieurs peuvent être mis en évidence, car les solutions évolutionnaires sont [1] continues (espèces), [2] multidimensionnelles et multidisciplinaires. Leur approche de la conception est [3] agile, [4] évolutive, [5] en réseau, [6] axée sur l'héritage (génétique), [7] axée sur l'environnement (coévolution), et [8] elle tient compte du processus de développement complet (éco-évo-devo) du système (organisme).
    - Ainsi, l'approche évolutive unit les principes adaptatifs et évolutifs (chapitre 4). Cela permet de combler et de créer des synergies entre les lacunes identifiées de l'ED et de l'ES grâce à des axiomes et des principes qui traitent des caractéristiques de l'ARR.
    - Les caractéristiques des architectures de systèmes évolutives (eSAR) ont été abordées au chapitre 4. Le processus de conception associé développé ou eSARD (chapitre 5) a directement abordé cette question. Il a été résumé par le tétraèdre de conception évolutive (Figure 182) qui repose sur trois niveaux de base du système

(GBS), à savoir :

- La géométrie (adaptabilité) comprend les aspects géométriques, l'esthétique, l'incertitude, les interfaces et l'optimisation.
- Le comportement (réactivité) considère les aspects fonctionnels tels que l'énergie, le calendrier, la synergie et les algorithmes.
- La substance (régénération) prend en compte les sujets de mise en œuvre de natures multiples (physique, numérique, virtuelle ou une combinaison des deux), ainsi que les sujets de fabricabilité, d'efficacité, de recyclabilité et de coût relatif.

- Sur ces points, la méthode eSARD permet un cycle de développement complet pour les architectures de systèmes évolutifs qui est basé sur trois clés de voûte du réseau ou principes de système tels que la conception, les opérations et la mise en œuvre (DOI). L'objectif de cette approche est de créer une méthode de conception qui permet de réaliser plus efficacement des architectures de système évolutives, réactives et régénératives, en traitant et en abordant les facteurs de stress mondiaux. Elle repose principalement sur une perspective synergique qui aborde chaque phase du cycle de vie et chaque défi de conception à partir des synergies entre les disciplines et les composants, plutôt que sur l'approche classique "diviser pour mieux régner". Ces principes permettent de gérer plus rapidement et plus efficacement les relations entre les facteurs de conception complexes (figure 124).
- Les lacunes des méthodologies DE et SE les plus récentes sont entrelacées et complétées par les principes eSARD en réponse à cette question de recherche. Ces principes sont dérivés à la fois des facteurs de stress environnementaux mondiaux et des nouveaux besoins des systèmes. Ils élargissent les perspectives plus traditionnelles pour répondre à tous les besoins clés d'un eSAR, tout en ouvrant de nouvelles possibilités de recherche vers l'application de l'approche évolutive à d'autres domaines techniques, de conception et de développement.

- **Comment une méthode de conception qui prend en compte les questions précédentes pourrait-elle être utilisée pour développer plus efficacement des systèmes complexes dans un tel environnement, lorsqu'il n'y a pas d'héritage direct et que la performance du système est une nécessité ?**
  - Le développement efficace d'architectures évolutives nécessite une méthode de conception à cycle complet, synergique et en réseau. Cela signifie un flux de travail hautement adaptable qui gère des efforts de conception continus, qui tient compte des liens multiples entre les outils et les étapes, et qui intègre des éléments du patrimoine tout au long du processus. La méthode eASRD (Figure 183) est développée à partir de trois domaines de développement universels et interconnectés dans le cadre de cette thèse :
    - La conception (géométrie) aborde les aspects organisationnels, de conception, de systèmes, d'esthétique et de validation.
    - La mise en œuvre (substance) s'attaque à la fabrication ou au codage, à la validation, à la vérification et aux essais.
    - L'optimisation opérationnelle (comportement) comprend les opérations du système et l'optimisation globale du système.
  - Les méthodes eSARD visent la perturbation (adaptabilité), l'intelligence (réactivité) et l'efficacité (régénération des ressources) du système (section 5.4) et dans un cadre de référence mesurable (figure 135). Cela répond également à la nécessité de concevoir mieux, plus intelligemment et plus largement, tout en obtenant plus avec moins (ressources, pièces, coût).
  - Par conséquent, un flux de travail eSARD devrait être conçu pour augmenter la capacité ARR du système (Figure 139) du point de vue de l'individu, de l'équipe, de la machine ou d'un système hybride, tout en tirant le meilleur parti des méthodes SOA actuelles, des principes évolutifs et des caractéristiques du système ARR. Ainsi, ce processus est organisé autour d'une hélice qui se présente :
    - Trois secteurs DOI continus, reproductibles et en réseau qui abordent tous les domaines et outils de développement.

- Des jalons et des étapes de vérification tout au long du cycle de vie : SFR, PDR, CDR, TRR, SVR, ORR, et OPRR.
- Outils permettant d'identifier dynamiquement les lacunes synergiques (eADQN) pour une meilleure capacité du système (eAMGs). Ils permettent également d'avoir une vision plus large du processus de conception (eASGs) et de la définition du système (eASMs). Il en résulte des délais de conception plus courts, une meilleure utilisation du patrimoine relatif et de meilleures performances du système avec les mêmes paramètres.

- En remettant en question la nature interrogative du processus de conception et les liens entre toutes les phases traditionnelles, cette question de recherche trouve une nouvelle voie qui est rendue possible par de nouveaux outils utilisant les capacités actuelles. Cela ouvre également la voie à une approche plus fluide mais gérable de la conception d'architectures de systèmes complexes basés sur le matériel (eSAR).

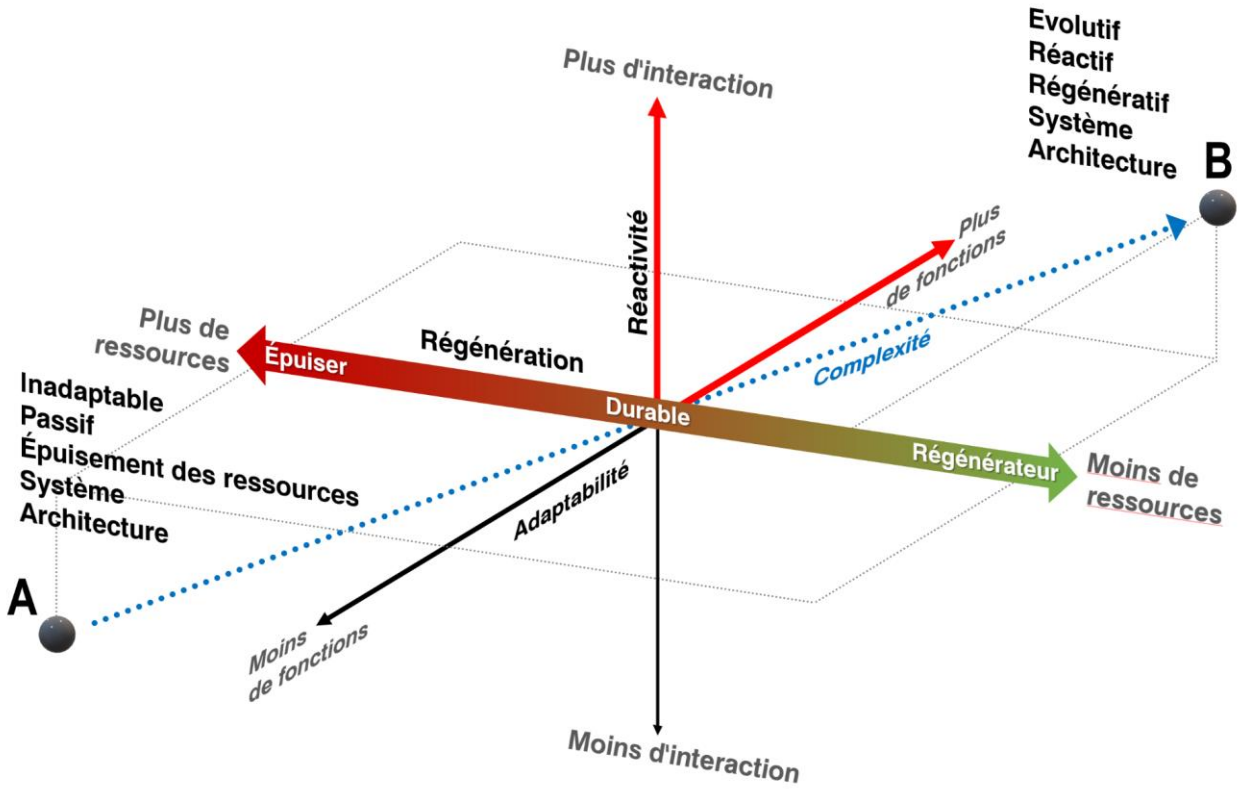



*Figure 197. Cadre de référence évolutif pour les architectures de systèmes évolutifs complexes (eSAR) basé sur les principes ARR.*

## 8.7.2. Conclusion (chapitre 7)

Le monde est en perpétuel changement, mais peut-être aujourd'hui plus que jamais en raison de raisons climatiques, économiques et sociales plus importantes qui se produisent simultanément à l'échelle mondiale. La pratique de la conception de systèmes complexes est affectée par ces nouveaux facteurs de stress contextuels croissants, comme l'a présenté le chapitre 2. Aujourd'hui, les systèmes et les services ont besoin d'une augmentation de leurs capacités en raison de facteurs de stress communs tels que la compétitivité, la rapidité de mise sur le marché et la compétitivité, entre autres. Cependant, il existe également un nombre croissant d'autres facteurs de stress mondiaux ayant une grande influence sur ces activités de conception. Il s'agit notamment de : [1] la rareté croissante des ressources, [2] le besoin généralisé de meilleures performances des systèmes dans tous les domaines techniques, [3] l'influence de l'héritage culturel sur la main-d'œuvre et

les flux de travail, et [4] l'infusion de techniques axées sur les données, parmi de nombreuses autres perturbations technologiques ayant une influence directe sur les systèmes complexes basés sur le matériel (CHS).

Trois questions de recherche (section 1.2) ont délimité l'hypothèse initiale de cette thèse, à savoir comment les mécanismes de conception inspirés de l'évolution naturelle sont appliqués à la conception et à la mise en œuvre physique de systèmes complexes pour permettre des processus de développement plus efficaces et justifier des architectures de systèmes complexes (SHC) plus adaptables. Néanmoins, cette thèse est fondée sur des années de recherche, des analyses documentaires approfondies, une expérience pratique dans de nombreux domaines techniques et une connaissance approfondie des techniques de pointe en matière d'ingénierie de conception (EC) et d'ingénierie des systèmes (IS).

Par conséquent, ces facteurs de stress de conception soulignent les changements à venir dans le rapport entre les besoins des systèmes et les ressources disponibles (section 2.3). En outre, cela signifie également que les futurs systèmes complexes et leurs méthodologies de conception associées devront s'adapter. Ceci est particulièrement significatif pour le développement de systèmes complexes basés sur le matériel, en raison du besoin croissant d'exigences multidisciplinaires et de l'influence des technologies perturbatrices qui changent la façon dont nous pouvons concevoir, mettre en œuvre et même exploiter ces systèmes.

Dans ce contexte, cette thèse de doctorat et la recherche associée sont basées sur des années de pratique qui ont débuté par l'étude de ces nouveaux besoins, ainsi que sur les principales lacunes identifiées dans l'état actuel de l'ingénierie de conception et des techniques d'ingénierie des systèmes (chapitre 3). La nécessité de corréler des définitions géométriques complexes avec des descriptions de systèmes plus avancés et en réseau a mis en évidence que ces lacunes entre elles ont tendance à être complémentaires et manquent d'adaptabilité face à une nouvelle réalité basée sur le changement, l'incertitude et les développements rapides.

Cette étude approfondie des besoins, des lacunes et des capacités a conduit au développement de la perspective évolutive en tant que point de vue hybride entre l'application de principes adaptables aux produits du système et les méthodes évolutives à leurs processus de développement. Les principes évolutifs extraits de la nature ont été étudiés (section 0), ainsi que leurs applications initiales, principalement en informatique depuis les années 50, par le biais des algorithmes génétiques et des techniques de programmation évolutive. Le résultat est un sous-ensemble de SHC avec certaines caractéristiques particulières et une méthode de développement de système subséquente.

Les **architectures de systèmes évolutifs (eSAR)** constituent un type de SCH en réponse aux facteurs de stress généraux précédents (chapitre 4) et en raison de l'application de principes évolutifs à la conception du système. Plus précisément, elles présentent trois caractéristiques essentielles.

- **L'adaptabilité,** qui est liée à la capacité du système à changer et à adapter à la fois sa géométrie et son comportement. Selon cette approche, ces systèmes sont dynamiques et leur développement est continu par nature. Cette caractéristique du système concerne à la fois les aspects géométriques du système et l'évaluation des fonctions du système.
- **Réactivité.** Ces systèmes sont intelligents et interagissent avec l'environnement, leur nature et leur schéma opérationnels doivent donc être abordés dès le début. Ce principe concerne principalement les interactions entre les systèmes.
- **Régénération**. Enfin, il ne s'agit pas seulement du système mais de sa mise en œuvre (physique, numérique, hybride). Ainsi, la substance du système, l'utilisation des ressources, la recyclabilité et la durabilité sont prises en compte à tous les niveaux.

L'approche évolutive des systèmes complexes est fondée sur une série de facteurs de conception clés associés à ces caractéristiques. Ceux-ci sont intégrés dans un cadre de référence de conception tridimensionnel ou tétraèdre évolutif (figure 182) fournissant des coordonnées pour définir ces systèmes. Dans cet espace, deux niveaux de développement tels que les principes de conception (DOI) et les détails des systèmes (GBS) sont interconnectés du point de vue du réseau de cette approche.

**La méthode eSARD** a été créée en réponse à ces caractéristiques uniques de système qui nécessitent une approche de développement qui englobe ces principes fondamentaux. Cette approche est basée sur [1] la recherche de lacunes et de liens synergiques entre les exigences, les technologies, les sous-systèmes et les solutions patrimoniales, entre autres,

[2] l'étude du système à partir de plusieurs points de vue RAR dans une perspective en réseau, et [3] des objectifs et des principes de conception spécifiques pour développer de meilleurs systèmes eSAR complexes plus rapidement, plus facilement, plus économiquement et plus efficacement. La méthode eSARD est multi-niveaux, multidisciplinaire, et hautement efficace. L'approche évolutive des systèmes complexes est fondée sur une série de facteurs de conception clés associés à ces caractéristiques. Ils sont intégrés dans un cadre de référence de conception tridimensionnel ou tétraèdre évolutif (figure 182) qui fournit des coordonnées pour définir ces systèmes. Dans cet espace, deux niveaux de développement tels que les principes de conception (DOI) et les détails des systèmes (GBS) sont interconnectés du point de vue du réseau de cette approche.

La méthode eSARD a été créée en réponse à ces caractéristiques uniques de système qui nécessitent une approche de développement qui englobe ces principes fondamentaux. Cette approche est basée sur [1] la recherche de lacunes et de liens synergiques entre les exigences, les technologies, les sous-systèmes et les solutions patrimoniales, entre autres, [2] l'étude du système à partir de plusieurs points de vue RAR dans une perspective en réseau, et [3] des objectifs et des principes de conception spécifiques pour développer de meilleurs systèmes eSAR complexes plus rapidement, plus facilement, plus économiquement et plus efficacement. La méthode eSARD est multi-niveaux, multidisciplinaire et hautement adaptable par nature.

En outre, son approche de la conception (chapitre 5) permet d'aborder beaucoup plus de sujets pertinents pour la faisabilité et l'intégrité du système, tout en traitant simultanément plus de disciplines. Cette méthode englobe les nouvelles technologies et techniques, en complément d'autres outils et approches traditionnels qui peuvent être utilisés par des individus, des équipes, des machines et des flux de travail hybrides. Le cœur de cette approche consiste à trouver les bonnes questions, en intensifiant à la fois la profondeur et la portée de tout questionnement initial. Cela permet de trouver, d'un point de vue véritablement multidisciplinaire, tous les obstacles réels à la conception de tout système. Enfin, les méthodes eSARD apportent suffisamment de flexibilité au processus pour changer, évoluer et varier rapidement l'échelle et le niveau de l'effort, depuis les domaines de haut niveau jusqu'aux détails du système, afin d'obtenir des conceptions de systèmes réalisables et performantes. Par conséquent, ce processus (section 5.7) améliore les méthodes traditionnelles avec une série d'outils et de techniques inspirés de l'évolution qui permettent une méthodologie plus efficace et mesurable. Parmi les outils les plus pertinents de cette méthode, les réseaux de questionnement évolutifs (eADQNs) décrits dans la section 5.8 permettent de trouver les écarts de maturité critiques ou eAMGs (section 5.9) qui deviennent le point de départ multidisciplinaire et multiniveau du processus de conception. Ces écarts sont critiques car ils permettent de trouver les sujets de conception les plus synergiques qui conditionnent la faisabilité du système, ainsi que l'efficacité du processus de développement puisque leur nature multidisciplinaire permet d'aborder plusieurs perspectives à la fois.

Un flux de travail eSARD évalue en permanence toutes les différentes facettes de la conception, de la mise en œuvre et de l'exploitation (DOI) des systèmes et des familles de systèmes (espèces). Cela permet d'adopter une approche à la fois holistique et pratique de toute activité d'ESD. Le début du processus est abordé par des géométries d'amorçage évolutives interconnectées ou eASG (section 5.10) qui conduisent à des modèles de systèmes évolutifs (eASM) qui décrivent et définissent le système. Parmi les outils de ces modèles figurent les croquis de systèmes évolutifs (eSD, section 0) et les listes d'équipements GBS (eGBSEL, section 5.11.2), qui sont des améliorations évolutives des outils classiques et constituent une méthode plus ouverte, plus adaptable et plus fiable. Cette approche est soutenue par des connexions et des liens entre tous les secteurs de développement, comme le décrit le diagramme en hélice d'eSARD (section 5.6), et par l'évaluation et l'infusion ultérieures d'apports patrimoniaux.

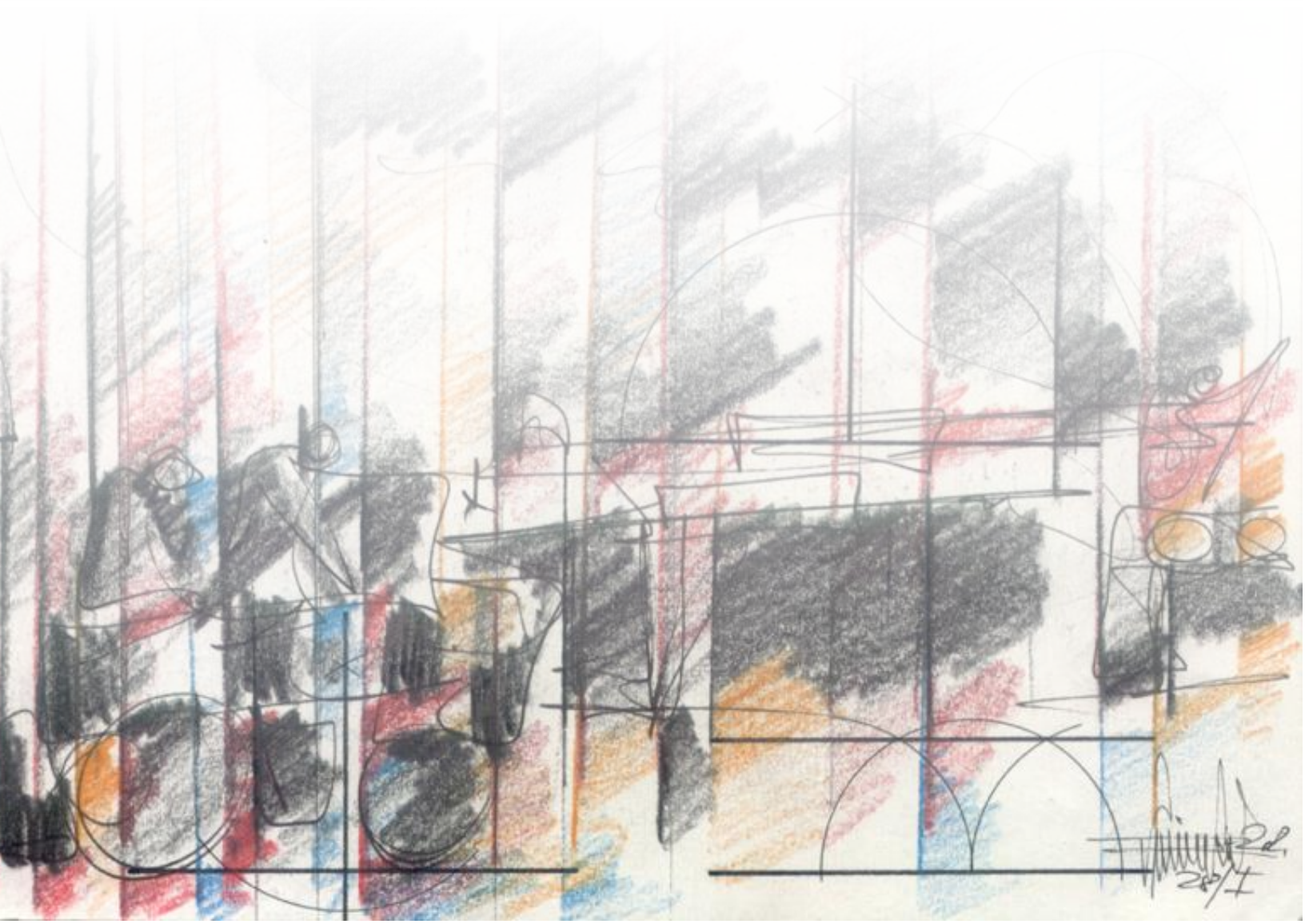
Cette approche évolutive présente également un moyen de mesurer et d'évaluer le développement d'un système grâce à l'utilisation des niveaux de maturité de l'architecture (eAML), comme décrit dans la section 5.12. Ainsi, des cycles de conception rapides, facilités et contrôlés peuvent être gérés et menés efficacement dans ce cadre de conception. L'approche eSARD intègre un cadre de collaboration, de concordance et de collaboration (3C) à sa pratique, ce qui garantit son adaptabilité aux nouveaux outils, capacités et agents de conception à l'avenir. En substance, cette approche présente une méthode universelle pour les SHC qui est adaptée aux eSAR.

# END MATTER

## Reference and Appendix

## CHAPTER 9

*“Haz las preguntas correctas si quieres encontrar las respuestas correctas”.*
***Vanessa Redgrave***



# References

Abraham, A., Jain, L.C., Goldberg, R. (Eds.), 2005. Evolutionary multiobjective optimization: theoretical advances and applications, Advanced information and knowledge processing. Springer, New York.

Abramson, A., 2007. The History of Television, 1942 to 2000. McFarland.

Abruzzo, E., Ellingsen, E., Solomon, J.D., 2007. Models: 306090 11. Princeton Architectural Press.

Agkathidis, A., 2016. Generative Design: Form-finding Techniques in Architecture. Laurence King Publishing.

Albers, A., Wintergerst, E., 2014. The Contact and Channel Approach (C&C2-A): Relating a System's Physical Structure to Its Functionality, in: Chakrabarti, A., Blessing, L.T.M. (Eds.), An Anthology of Theories and Models of Design. Springer London, London, pp. 151–171. https://doi.org/10.1007/978-1-4471-6338-1_8

Alexiou, K., Johnson, J., Zamenopoulos, T., 2009. Embracing Complexity in Design. Routledge.

Allbee, B., 2018. Hands-On Software Engineering with Python: Move beyond basic programming and construct reliable and efficient software with complex code. Packt Publishing Ltd.

Allen, P., Maguire, S., McKelvey, B., 2011. The SAGE Handbook of Complexity and Management. SAGE.

Alotaibi, S., Alotaibi, F., Ibrahim, O.M., 2020. Solar-assisted steam power plant retrofitted with regenerative system using Parabolic Trough Solar Collectors. Energy Rep. 6, 124–133. https://doi.org/10.1016/j.egyr.2019.12.019

Altshuller, 1984. Creativity As an Exact Science. CRC Press.

Altshuller, G., 2002. 40 Principles: TRIZ Keys to Innovation. Technical Innovation Center, Inc.

Álvarez, A., Ritchey, T., 2015. Applications of General Morphological Analysis 4, 40.

Alves, R.M. de B., Nascimento, C.A.O. do, Biscaia, E.C., 2009. 10th International Symposium on Process Systems Engineering. Elsevier.

America, P., van de Laar, P., Muller, G., Punter, T., van Rooijen, N., Rutgers, J., Watts, D., 2010. Architecting for Improved Evolvability, in: Van de Laar, P., Punter, T. (Eds.), Views on Evolvability of Embedded Systems, Embedded Systems. Springer Netherlands, Dordrecht, pp. 21–36. https://doi.org/10.1007/978-90-481-9849-8_2

An, J., Rau, R., 2019. Finance, technology and disruption. Eur. J. Finance 0, 1–12. https://doi.org/10.1080/1351847X.2019.1703024

Ang, Y., Yin, S., 2008. Intelligent Complex Adaptive Systems. IGI Global.

Arnold, J.E., Clancey, W.J., 1959. Creative Engineering: Promoting Innovation by Thinking Differently.

Arthur, W., 2002. The emerging conceptual framework of evolutionary developmental biology. Nature 415, 757–764. https://doi.org/10.1038/415757a

Ashby, W.R., 1991. Principles of the Self-Organizing System, in: Facets of Systems Science. Springer US, Boston, MA, pp. 521–536. https://doi.org/10.1007/978-1-4899-0718-9_38

Ashlock, D., 2006. Evolutionary Computation for Modeling and Optimization. Springer Science & Business Media.

Asimov, M., 1976. Introduction to Design. Photoduplication Service, Library of Congress, Washington, D.C.

Astels, D., 2003. Test-driven Development: A Practical Guide. Prentice Hall PTR.

Aughenbaugh, J., Paredis, C., 2004. The Role and Limitations of Modeling and Simulation in Systems Design. https://doi.org/10.1115/IMECE2004-59813

Autodesk, 2020. Generative Design for Manufacturing With Fusion 360 | Autodesk.

Baba, N., Jain, L.C., Howlett, R.J., 2001. Knowledge-based Intelligent Information Engineering Systems & Allied Technologies: KES'2001. IOS Press.

Babers, C., 2015. The Enterprise Architecture Sourcebook, Volume 1, Second Edition. Lulu.com.

Backhouse, C.J., Brookes, N.J., 1996a. Concurrent Engineering: What's Working where. Gower Publishing, Ltd.

Backhouse, C.J., Brookes, N.J., 1996b. Concurrent Engineering: What's Working where. Gower Publishing, Ltd.

Bade, R., Parkin, M., 2012. Essential Foundations of Economics, 6 edition. ed. Pearson, Boston.

Badiru, A.B., 2019. Systems engineering models: theory, methods, and applications, Systems innovation series. Taylor & Francis, a CRC title, part of the Taylor & Francis imprint, a member of the Taylor & Francis Group, the academic division of T&F Informa, plc, Boca Raton.

Badiru, A.B., 2013. Handbook of Industrial and Systems Engineering. CRC Press.

Baeck, T., Fogel, D.B., Michalewicz, Z., 2018. Evolutionary Computation 1: Basic Algorithms and Operators. CRC Press.

Bahill, A.T., Madni, A.M., 2017. Tradeoff Decisions in System Design. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-43712-5

Bahill, A.T., Madni, A.M., 2016. Tradeoff Decisions in System Design. Springer.

Bansal, A.K., 2013. Introduction to Programming Languages. CRC Press.

Barnard, P., May, J., Duke, D., Duce, D., 2000. Systems, interactions, and macrotheory. ACM Trans. Comput.-Hum. Interact. 7, 222–262. https://doi.org/10.1145/353485.353490

Barnes, Y., 2018. 8 things to know about global real estate value. Savills Impacts. URL https://www.savills.com/impacts/market-trends/8-things-you-need-to-know-about-the-value-of-global-real-estate.html (accessed 6.14.20).

Barr, M., Massa, A., 2006. Programming Embedded Systems: With C and GNU Development Tools. O'Reilly Media, Inc.

Bártolo, P.J., Soares de Lemos, A.C., International Conference on Advanced Research in Virtual and Rapid Prototyping, Centro para o Desenvolvimento Rápido e Sustentado do Produto (Eds.), 2012. Innovative developments in virtual and physical prototyping: proceedings of the 5th International Conference on Advanced Research and Rapid Prototyping, Leiria, Portugal, 28 September - 1 October, 2011 ; [the 5th International Conference on Advanced Research in Virtual and Physical Prototyping (VR@P 2011)]. CRC Press/Balkema, Boca Raton, Fla.

Bathe, K.-J., 2006. Finite Element Procedures. Klaus-Jurgen Bathe.

Batson, R.G., 1986. N2 Chart Key Features. Wikipedia.

Bauer, M., Mösle, P., Schwarz, M., 2009. Green Building: Guidebook for Sustainable Architecture. Springer Science & Business Media.

Beard, D.C., 2020. Shelters, Shacks and Shanties. Library of Alexandria.

Becattini, N., Cascini, G., Rotini, F., 2015. OTSM-TRIZ Network of Problems for Evaluating the Design Skills of Engineering Students. Procedia Eng. 131, 689–700. https://doi.org/10.1016/j.proeng.2015.12.356

Beck, K., 2003. Test-driven Development: By Example. Addison-Wesley Professional.

Beck, K., Andres, C., 2004. Extreme Programming Explained: Embrace Change. Pearson Education.

Bekey, G.A., Goldberg, K.Y., 2012. Neural Networks in Robotics. Springer Science & Business Media.

Bell, G., 1996. Selection: The Mechanism of Evolution. Springer Science & Business Media.

Benevolo, L., 1977. History of Modern Architecture. MIT Press.

Bentley, P. (Ed.), 1999. Evolutionary design by computers. Morgan Kaufmann Publishers, San Francisco, Calif.

Bentley, P.J., Wakefield, J.P., 1996. Conceptual Evolutionary Design by a Genetic Algorithm 14.

Benton, T., Cohen, J.-L., 2019. Le Corbusier Le Grand. Phaidon Press.

Berdonosov, V., Redkolis, E., 2010. ON CLASSIFICATION OF COMPUTER-AIDED SOFTWARE ENGINEERING SYSTEMS (CASE). Sch. Notes Komsomolsk-Na-Amure State Tech. Univ. 1, 12–25. https://doi.org/10.17084/2010.IV-1(4).2

Beyer, H.-G., 2013. The Theory of Evolution Strategies. Springer Science & Business Media.

Bhamra, T., Lofthouse, V., 2016. Design for Sustainability: A Practical Approach. CRC Press.

Birolini, A., 2007. Reliability Engineering: Theory and Practice. Springer Science & Business Media.

Bittner, K., Spence, I., 2006. Managing Iterative Software Development Projects. Addison-Wesley Professional.

Blessing, L.T.M., Chakrabarti, A., 2009. DRM, a Design Research Methodology, 2009 edition. ed. Springer, Dordrecht ; London.

Blokdyk, G., 2019. Technology Readiness Level a Complete Guide - 2020 Edition. Emereo Pty Limited.

Blokdyk, G., 2018. Systematic Inventive Thinking A Complete Guide. Emereo Pty Limited.

Blokdyk, G., 2017. Iterative and Incremental Development: Practical Design Techniques. CreateSpace Independent Publishing Platform.

Boardman, J., Sauser, B. (Eds.), 2013. Systemic Thinking: Building Maps for Worlds of Systems. John Wiley & Sons, Inc., Hoboken, NJ, USA. https://doi.org/10.1002/9781118721216

Boardman, J., Sauser, B., 2008. Systems Thinking: Coping with 21st Century Problems, 1st edition. ed. CRC Press, Boca Raton, FL.

Boardman, J., Sauser, B., 2006. System of Systems - the meaning of of, in: 2006 IEEE/SMC International Conference on System of Systems Engineering. Presented at the 2006 IEEE/SMC International Conference on System of Systems Engineering, IEEE, Los Angeles, California, USA, pp. 118–123. https://doi.org/10.1109/SYSOSE.2006.1652284

Boccaletti, S., 2010. Handbook on Biological Networks. World Scientific.

Boehm, B., Koolmanojwong, S., Lane, J.A., Turner, R., 2012. Principles for Successful Systems Engineering. Procedia Comput. Sci. 8, 297–302. https://doi.org/10.1016/j.procs.2012.01.063

Boehm, B., Lane, J.A., Koolmanojwong, S., Turner, R., 2014. The Incremental Commitment Spiral Model: Principles and Practices for Successful Systems and Software. Addison-Wesley Professional.

Boehm, B.W., 1988. A spiral model of software development and enhancement. Computer 21, 61–72. https://doi.org/10.1109/2.59

Boje, C., Guerriero, A., Kubicki, S., Rezgui, Y., 2020. Towards a semantic Construction Digital Twin: Directions for future research. Autom. Constr. 114, 103179. https://doi.org/10.1016/j.autcon.2020.103179

Bonanomi, M.M., 2019. Digital Transformation of Multidisciplinary Design Firms: A Systematic Analysis-Based Methodology for Organizational Change Management. Springer.

Bonjour, É., Micaëlli, J.-P., 2010. Design Core Competence Diagnosis: A Case From the Automotive Industry. IEEE Trans. Eng. Manag. 57, 323–337. https://doi.org/10.1109/TEM.2009.2036838

Borgnakke, C., Sonntag, R.E., 2013. Fundamentals of thermodynamics, 8 [edition]. ed. Wiley, Hoboken, NJ.

Borky, J.M., Bradley, T.H., 2018. Effective Model-Based Systems Engineering. Springer.

Boswell, M.R., Greve, A.I., Seale, M.T.L., 2019. Climate Action Planning: A Guide to Creating Low-Carbon, Resilient Communities, Revised edition. ed. Island Press, Washington, DC.

Bozorg-Haddad, O., Solgi, M., Loáiciga, H.A., 2017. Meta-heuristic and Evolutionary Algorithms for Engineering Optimization. John Wiley & Sons.

Brackney, L., Parker, A., Macumber, D., Benne, K., 2018. Building Energy Modeling with OpenStudio: A Practical Guide for Students and Professionals. Springer.

Bradley, A.R., 2011. Programming for Engineers: A Foundational Approach to Learning C and Matlab. Springer Science & Business Media.

Braha, D., Minai, A.A., Bar-Yam, Y., 2007. Complex Engineered Systems: Science Meets Technology. Springer.

Braha, D., Minai, A.A., Bar-Yam, Y. (Eds.), 2006. Complex Engineered Systems: Science Meets Technology, 2006 edition. ed. Springer, Berlin ; New York.

Brandon, P.S., Kocatürk, T., 2009. Virtual Futures for Design, Construction and Procurement. John Wiley & Sons.

Brooks, D.R., Wiley, E.O., Wiley, E.O., 1988. Evolution As Entropy. University of Chicago Press.

Brown, T., 2013. Engineering economics and economic design for process engineers.

Brown, T., 2009. Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. Harper Collins.

Brusa, E., Calà, A., Ferretto, D., 2017. Systems Engineering and Its Application to Industrial Product Development. Springer.

Buchholz, S., Briggs, B., Sharma, S., 2020. Tech Trends 2020. Deloitte Consulting LLP.

Budde, R., Kautz, K., Kuhlenkamp, K., Züllighoven, H., 2012. Prototyping: An Approach to Evolutionary System Development. Springer Science & Business Media.

Buede, D.M., 2009. The engineering design of systems: models and methods, 2nd ed. ed, Wiley series in systems engineering and management. John Wiley & Sons, Hoboken, N.J.

Burge, D.S., 2009. The Systems Engineering Tool Box 15.

Burke, A., 1999. High Energy Density Regenerative Fuel Cell Systems for Terrestrial Applications 18.

Burke, C.S., Pierce, L.G., Salas, E., 2006. Understanding Adaptability: A Prerequisite for Effective Performance within Complex Environments. Emerald Group Publishing.

Burns, S.A., 2002. Recent Advances in Optimal Structural Design. ASCE Publications.

Burnstein, I., Homyen, A., Grom, R., Carlson, C.R., 1998. A Model to Assess Testing Process Maturity 5.

Butlin, R., Bridle, J., Schluter, D., 2009. Speciation and Patterns of Diversity. Cambridge University Press.

Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, J., Bonabeau, E., Theraula, G., 2003. Self-organization in Biological Systems. Princeton University Press.

Carmichael, T., Hadžikadić, M., 2019. The Fundamentals of Complex Adaptive Systems, in: Carmichael, T., Collins, A.J., Hadžikadić, M. (Eds.), Complex Adaptive Systems, Understanding Complex Systems. Springer International Publishing, Cham, pp. 1–16. https://doi.org/10.1007/978-3-030-20309-2_1

Carnergi Mellon, S., 2017. Software Architecture [WWW Document]. URL https://www.sei.cmu.edu/our-work/projects/display.cfm?customel_datapageid_4050=21328&customel_datapageid_4050=21328 (accessed 12.30.20).

Cash, P., Stanković, T., Štorga, M., 2016. Experimental Design Research: Approaches, Perspectives, Applications. Springer.

Cask05, 2006. House of Quality.

Casti, J., Karlqvist, A., 2003. Art and Complexity. Elsevier.

Cavallucci, D., 2017. TRIZ – The Theory of Inventive Problem Solving: Current Research and Trends in French Academic Institutions. Springer.

Ceschin, F., 2013. Sustainable Product-Service Systems: Between Strategic Design and Transition Studies. Springer Science & Business Media.

Chakrabarti, A., 2019. Research into Design for a Connected World: Proceedings of ICoRD 2019 Volume 1. Springer.

Chakrabarti, A., Blessing, L.T.M. (Eds.), 2014. An Anthology of Theories and Models of Design. Springer London, London. https://doi.org/10.1007/978-1-4471-6338-1

Chakrabarti, C.G., Ghosh, K., 2011. Biological Evolution: Entropy, Complexity and Stability. J. Mod. Phys. 02, 621–626. https://doi.org/10.4236/jmp.2011.226072

Chang, K.-H., 2015. e-Design: computer-aided engineering design. Academic Press, Amsterdam ; New York.

Charlesworth, B., Charlesworth, D., 2017. Evolution: A Very Short Introduction. Oxford University Press.

Chaturvedi, D.K., 2010. Modeling and simulation of systems using MATLAB and Simulink. CRC Press, Boca Raton.

Chen, C.-Y., Hwang, R.-C., 2009. A new variable topology for evolutionary hardware design. Expert Syst. Appl. 36, 634–642. https://doi.org/10.1016/j.eswa.2007.09.017

Chen, P., Han, J., 2002. Facilitating system-of-systems evolution with architecture support, in: Proceedings of the 4th International Workshop on Principles of Software Evolution - IWPSE '01. Presented at the the 4th international workshop, ACM Press, Vienna, Austria, p. 130. https://doi.org/10.1145/602461.602489

Chen, Z., Mendes, A., Yan, Y., Chen, S. (Eds.), 2018. Intelligent Robotics and Applications: 11th International Conference, ICIRA 2018, Newcastle, NSW, Australia, August 9–11, 2018, Proceedings, Part II, Lecture Notes in Computer Science. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-97589-4

Childs, P.R.N., 2013. Mechanical Design Engineering Handbook. Butterworth-Heinemann.

Chua, C.K., Leong, K.F., Lim, C.S., 2010. Rapid Prototyping: Principles and Applications. World Scientific.

Chung, S.H., Bayer, T.J., Cole, B., Cooke, B., Dekens, F., Delp, C., Lam, D., 2012. Model-based systems engineering approach to managing mass margin.

CIO Council, 2013. FEA Consolidated Reference Model.

Clarke, J., 2007. Energy Simulation in Building Design. Routledge.

Clements, P. (Ed.), 2011. Documenting software architectures: views and beyond, 2nd ed. ed, SEI series in software engineering. Addison-Wesley, Upper Saddle River, NJ.

Clerc, M., 2013. Particle Swarm Optimization. John Wiley & Sons.

CMMI Product Team, 2018. CMMI for Development, Version 1.3 4037844 Bytes. https://doi.org/10.1184/R1/6572342.V1

Cody, B., 2017. Form Follows Energy, 1 edition. ed. Birkhäuser, Basel.

Cohn, M., 2010. Succeeding with Agile: Software Development Using Scrum. Pearson Education.

Colozza, A.J., Maloney, T., 2003. Initial Design and Construction of a Mobil Regenerative Fuel Cell System 36.

Conrad, M., 2012. Adaptability: The Significance of Variability from Molecule to Ecosystem. Springer Science & Business Media.

Cooper, K.G., 2001. Rapid prototyping technology selection and application. Marcel Dekker, New York.

Cooper, P., 2005. Speciation in the computing sciences: digital forensics as an emerging academic discipline, in: Proceedings of the 2nd Annual Conference on Information Security Curriculum Development - InfoSecCD '05. Presented at the the 2nd annual conference, ACM Press, Kennesaw, Georgia, p. 19. https://doi.org/10.1145/1107622.1107628

Corser, R., 2012. Fabricating Architecture: Selected Readings in Digital Design and Manufacturing. Chronicle Books.

Cortright, J., Mayer, H., 2001. High Tech Specialization: A Comparison of High Technology Centers.

Costikyan, G., 2013. Uncertainty in Games. MIT Press.

Cottrell, J.A., Hughes, T.J.R., Bazilevs, Y., 2009. Isogeometric Analysis: Toward Integration of CAD and FEA. John Wiley & Sons.

Couto, M.G., 2016. 150 Best Tiny Home Ideas. HarperCollins.

Crawley, E., Cameron, B., Selva, D., 2016. System Architecture: Strategy and Product Development for Complex Systems. Pearson.

Cross, N., 2011. Design thinking: understanding how designers think and work, Engl. ed. ed. Berg, Oxford.

Cross, N., 2008. Engineering Design Methods: Strategies for Product Design, 4 edition. ed. Wiley, Chichester, England ; Hoboken, NJ.

Curedale, R., 2013. Design Thinking: Process and Methods Manual. Design Community College.

Darke, J., 1979. The primary generator and the design process. Des. Stud. 1, 36–44. https://doi.org/10.1016/0142-694X(79)90027-9

Darwin, C., 1845. Voyage of the Beagle, Engraving [WWW Document]. URL http://darwin-online.org.uk/converted/published/1845_Beagle_F14/1845_Beagle_F14_fig07.jpg (accessed 6.26.20).

Darwish, A., 2011. Business Process Mapping: A Guide to Best Practice. Writescope Publishers.

Daszko, M., 2018. Pivot, Disrupt, Transform: How Leaders Beat the Odds and Survive. Diversion Books, Place of publication not identified.

Daugherty, P.R., Wilson, H.J., 2018. Human + Machine: Reimagining Work in the Age of AI. Harvard Business Press.

De Bono, E., 1993. Serious creativity: using the power of lateral thinking to create new ideas, 1st pbk. ed. ed. HarperBusiness, New York, N.Y.

de la Tour, A., Portincaso, M., Goeldel, N., Chaudhry, U., Tallec, C., Gourevitch, A., 2020. Deep Tech: The Great Wave of Innovation.

De la Tour, A., Soussan, P., Harlé, N., Chevalier, R., Duportet, X., 2017. From Tech to Deep Tech. Boston Consult. Group 52.

Deane, P. M., Deane, Phyllis M., 1979. The First Industrial Revolution. Cambridge University Press.

Defense Acquisition University, 2005. Systems Engineering Fundamentals, January 2001, Illustrated edition. ed. Defense Acquisition University, Fort Belvoir.

Dehlinger, J., 2009. Architecture - Design Methods - Inca Structures. Festschrift for Jean-Pierre Protzen. kassel university press GmbH.

Dejtiar, F., 2018. Etymology in Architecture: Tracing the Language of Design to its Roots [WWW Document]. ArchDaily. URL https://www.archdaily.com/898648/etymology-in-architecture-tracing-the-language-of-design-to-its-roots (accessed 6.21.20).

Demartini, G., 2015. Hybrid human–machine information systems: Challenges and opportunities. Comput. Netw., Crowdsourcing 90, 5–13. https://doi.org/10.1016/j.comnet.2015.05.018

Dempsey, I., O'Neill, M., Brabazon, A., 2009. Foundations in Grammatical Evolution for Dynamic Environments. Springer Science & Business Media.

Desjardins, J., McCandless, D., 2017. How Many Millions of Lines of Code Does It Take? [WWW Document]. Vis. Capital. URL https://www.visualcapitalist.com/millions-lines-of-code/ (accessed 6.28.20).

Deutsch, R., 2011. BIM and Integrated Design: Strategies for Architectural Practice, 1 edition. ed. Wiley.

Dickerson, C., Mavris, D.N., 2016. Architecture and Principles of Systems Engineering. CRC Press.

Dieckmann, U., Doebeli, M., Metz, J.A.J., Tautz, D., 2004. Adaptive Speciation. Cambridge University Press.

Dieter, G., Schmidt, L., 2012. Engineering Design, 5 edition. ed. McGraw-Hill Education, New York.

Dobzhansky, T., 1973. Nothing in Biology Makes Sense except in the Light of Evolution. Am. Biol. Teach. 35, 125–129. https://doi.org/10.2307/4444260

Dori, D., 2016. Model-Based Systems Engineering with OPM and SysML. Springer.

Dori, D., 2011. Object-Process Methodology: A Holistic Systems Paradigm. Springer Science & Business Media.

Dorigo, M., Dorigo, D. de R.D.F.M., Stützle, T., 2004. Ant Colony Optimization. MIT Press.

Dorst, K., Cross, N., 2001. Creativity in the design process: co-evolution of problem–solution. Des. Stud. 22, 425–437. https://doi.org/10.1016/S0142-694X(01)00009-6

Douglass, B.P., 2016. Agile systems engineering. Morgan Kaufmann, Waltham, MA.

Droste, M., 2015. The Bauhaus: 1919-1933: reform and avant-garde. Taschen, Köln.

Duato, J., Yalamanchili, S., Ni, L., 2003. Interconnection Networks. Morgan Kaufmann.

DuBrin, A.J., 2011. Essentials of Management. Cengage Learning.

Duehr, K., Heimicke, J., Breitschuh, J., Spadinger, M., Kopp, D., Haertenstein, L., Albers, A., 2019. Understanding Distributed Product Engineering: Dealing with Complexity for Situation- and Demand-Oriented Process Design. Procedia CIRP 84, 136–142. https://doi.org/10.1016/j.procir.2019.04.200

Dukas, R., 1998. Cognitive Ecology: The Evolutionary Ecology of Information Processing and Decision Making. University of Chicago Press.

Dukkipati, R.V., 2008. Matlab: An Introduction With Applications. New Age International.

Dumitrescu, D., Lazzerini, B., Jain, L.C., Dumitrescu, A., 2000. Evolutionary Computation. CRC Press.

Durham, W.H., 1991. Coevolution: Genes, Culture, and Human Diversity. Stanford University Press.

Dutchguilder, 2007. Iterative Development illustration.

Dym, C.L., 2013. Engineering Design: A Project-Based Introduction, 4 edition. ed. Wiley, New York.

Eastman, C., Teicholz, P., Sacks, R., Liston, K., 2011. BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors. John Wiley & Sons.

Eastman, C.M., 2012. Design for X: Concurrent engineering imperatives. Springer Science & Business Media.

Eckstein, J., 2013. Agile Software Development in the Large: Diving Into the Deep. Pearson Education.

Eder, W.E., Hosnedl, S., 2010. Introduction to Design Engineering: Systematic Creativity and Management. CRC Press.

Eggert, R.J., 2010. Engineering Design : Second Edition, 2nd edition. ed. High Peak Press, Meridian, Idaho.

Eggert, R.J., 2005. Engineering design. Pearson/Prentice Hall, Upper Saddle River, N.J.

EIA, 2019. International Energy Outlook 2019. Energy Inf. Adm. 85.

Eiben, A.E., Smith, J.E., 2007. Introduction to Evolutionary Computing. Springer Science & Business Media.

Eisner, H., 2011. Systems Engineering: Building Successful Systems. Morgan & Claypool Publishers.

Ekman, U., Bolter, J.D., Diaz, L., Sondergaard, M., Engberg, M., 2015. Ubiquitous Computing, Complexity and Culture. Routledge.

Elam, K., 2001. Geometry of Design: Studies in Proportion and Composition. Princeton Architectural Press.

Elangovan, U., 2020. Product Lifecycle Management (PLM): A Digital Journey Using Industrial Internet of Things (IIoT). CRC Press.

Epstein, D., 2019. Range: Why Generalists Triumph in a Specialized World. Penguin.

Erdogmus, P., 2018. Particle Swarm Optimization with Applications. BoD – Books on Demand.

Estefan, J.A., 2008. Survey of Model-Based Systems Engineering (MBSE) Methodologies. INCOSE MBSE Initiat. 70.

Etlin, R.A., 1996. Symbolic Space: French Enlightenment Architecture and Its Legacy. University of Chicago Press.

European Union, 2019. Deep-tech Innovations: When the Digital Technologies Meet the Real World. Publications Office of the European Union.

Evangeline, P., 2020. The Digital Twin Paradigm for Smarter Systems and Environments: the Industry Use Cases. Elsevier.

Evans, A., 2010. RESOURCE SCARCITY, CLIMATE CHANGE AND THE RISK OF VIOLENT CONFLICT. World Bank 24.

Evans, D., Back, W. de, 2011. Evolutionary robotics, in: Roberts, S.C. (Ed.), Applied Evolutionary Psychology. Oxford University Press, pp. 414–425. https://doi.org/10.1093/acprof:oso/9780199586073.003.0025

Evans, J.H., 1959. BASIC DESIGN CONCEPTS. J. Am. Soc. Nav. Eng. 71, 671–678. https://doi.org/10.1111/j.1559-3584.1959.tb01836.x

Evers, B., Thoenes, C., Kunstbibliothek (Berlin, Germany) (Eds.), 2015. Architectural theory: from the Renaissance to the present, Bibliotheca universalis. Taschen, Köln.

Fairley, D., Forsberg, K., 2020. System Life Cycle Process Models: Vee - SEBoK [WWW Document]. URL https://www.sebokwiki.org/wiki/System_Life_Cycle_Process_Models:_Vee (accessed 8.3.20).

Faisander, A., Adcock, R., 2020. System Design - SEBoK [WWW Document]. URL https://www.sebokwiki.org/wiki/System_Design (accessed 6.21.20).

Faisandier, A., Roedler, G., Adcock, R., 2020. System Architecture - SEBoK [WWW Document]. Syst. Archit. URL https://www.sebokwiki.org/wiki/System_Architecture (accessed 6.21.20).

Fanmuy, G., Goubault, E., Krob, D., Stephan, F., 2016. Complex Systems Design & Management: Proceedings of the Seventh International Conference on Complex Systems Design & Management, CSD&M Paris 2016. Springer.

Farid, A.M., Suh, N.P. (Eds.), 2016. Axiomatic Design in Large Systems: Complex Products, Buildings and Manufacturing Systems, 1st ed. 2016 edition. ed. Springer, New York, NY.

Farsi, M., Daneshkhah, A., Hosseinian-Far, A., Jahankhani, H., 2019. Digital Twin Technologies and Smart Cities. Springer.

Feoktistov, V., 2007. Differential Evolution: In Search of Solutions. Springer Science & Business Media.

Fernández, F., Hidalgo, J.I., Lanchares, J., Sánchez, J.M., 2004. A methodology for reconfigurable hardware design based upon evolutionary computation. Microprocess. Microsyst. 28, 363–371. https://doi.org/10.1016/j.micpro.2004.03.017

Fernandez, J.L., Hernandez, C., 2019. Practical Model-Based Systems Engineering. Artech House.

Finkenthal, M., 2008. Complexity, Multi-disciplinarity, and Beyond. Peter Lang.

Fiorineschi, L., Frillici, F.S., Rissone, P., 2015. A Comparison of Classical TRIZ and OTSM-TRIZ in Dealing with Complex Problems. Procedia Eng. 131, 86–94. https://doi.org/10.1016/j.proeng.2015.12.350

Fishel, H., 2018. Fidget!: 101 Ways to Boost Your Creativity and Decrease Your Stress. Simon and Schuster.

Flanagan, D.P., Ortiz, S.O., Alfonso, V.C., 2013. Essentials of Cross-Battery Assessment. John Wiley & Sons.

Fogel, G.B., Fogel, D.B., 1995. CONTINUOUS EVOLUTIONARY PROGRAMMING: ANALYSIS AND EXPERIMENTS. Cybern. Syst. 26, 79–90. https://doi.org/10.1080/01969729508927488

Ford, N., Parsons, R., Kua, P., 2017. Building Evolutionary Architectures: Support Constant Change. O'Reilly Media, Inc.

Forrest, S., 1993. Genetic algorithms: principles of natural selection applied to computation. Science 261, 872–878. https://doi.org/10.1126/science.8346439

Forsberg, K., 2020. System Life Cycle Process Models: Iterative - SEBoK [WWW Document]. URL https://sebokwiki.org/wiki/System_Life_Cycle_Process_Models:_Iterative (accessed 8.3.20).

Forsberg, K., Mooz, H., 2003. System Engieering for Faster, Cheaper, Better.

Forsberg, K., Mooz, H., 1992. The Relationship of Systems Engineering to the Project Cycle. Eng. Manag. J. 4, 36–43. https://doi.org/10.1080/10429247.1992.11414684

Forsberg, K., Mooz, H., Cotterman, H., 2005. Visualizing Project Management: Models and Frameworks for Mastering Complex Systems. John Wiley & Sons.

Fowler, M., 2018. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

Fox, C.W., Roff, D.A., Fairbairn, D.J., 2001. Evolutionary Ecology: Concepts and Case Studies. Oxford University Press.

Fox, M., 2016. Interactive Architecture: Adaptive World. Chronicle Books.

Frampton, K., 2020. Modern Architecture: A Critical History. Thames & Hudson.

Francis, S., 2019. Bubbletecture: Inflatable Architecture and Design. Phaidon Press.

Freeman, L.C., 1977. A Set of Measures of Centrality Based on Betweenness. Sociometry 40, 35. https://doi.org/10.2307/3033543

Freiberg, J., 2014. Bramante's Tempietto, the Roman Renaissance, and the Spanish Crown. Cambridge University Press.

Frey, D.D., Fukuda, S., Rock, G., 2011. Improving Complex Systems Today: Proceedings of the 18th ISPE International Conference on Concurrent Engineering. Springer Science & Business Media.

Fried, J., Hansson, D.H., 2010. Rework, 1 edition. ed. Currency, New York.

Friedenthal, S., Moore, A., Steiner, R., 2008. A practical guide to SysML: the systems modeling language. Elsevier/Morgan Kaufmann, Burlington, Mass.

Friedman, D.P., Wand, M., 2008. Essentials of Programming Languages. MIT Press.

Frodeman, R., Klein, J.T., Pacheco, R.C.S., 2017. The Oxford Handbook of Interdisciplinarity. Oxford University Press.

Gacek, C., Abd-Allah, A., Clark, B., Boehm, B., 1995. On the Definition of Software System Architecture 12.

Gaševic, D., Djuric, D., Devedžic, V., 2006. Model Driven Architecture and Ontology Development. Springer Science & Business Media.

Gengnagel, C., Kilian, A., Palz, N., Scheurer, F., 2011. Computational Design Modeling: Proceedings of the Design Modeling Symposium Berlin 2011. Springer Science & Business Media.

Gennaro, A., Marc, L., A, M.R., 2011. Biological Evolution: Facts and Theories : a Critical Appraisal 150 Years After "The Origin of Species." Gregorian Biblical BookShop.

Georgiev, G.Y., Smart, J.M., Martinez, C.L.F., Price, M.E., 2019. Evolution, Development and Complexity: Multiscale Evolutionary Models of Complex Adaptive Systems. Springer.

Gero, J.S., 2011. Analysing Design Protocols: Development of Methods and Tools 10.

Gero, J.S., Kannengiesser, U., 2014. The Function-Behaviour-Structure Ontology of Design, in: Chakrabarti, A., Blessing, L.T.M. (Eds.), An Anthology of Theories and Models of Design. Springer London, London, pp. 263–283. https://doi.org/10.1007/978-1-4471-6338-1_13

Gero, J.S., Kannengiesser, U., 2004. The situated function–behaviour–structure framework. Des. Stud. 25, 373–391. https://doi.org/10.1016/j.destud.2003.10.010

Gershenson, C., 2007. Design and Control of Self-organizing Systems. CopIt ArXives.

Gevorkian, P., 2009. Alternative Energy Systems in Building Design (GreenSource Books). McGraw Hill Professional.

GFAB, 2010. SysML diagrams collage.

Gilmore, C., 2014. Materials Science and Engineering Properties. Cengage Learning.

Godfrey, S., 2008. File:Characteristics of Capability Maturity Model.svg.

Goetsch, D.E., Rickman, R.L., Chalk, W.S., 2015. Technical Drawing for Engineering Communication. Cengage Learning.

Gombrich, E.H., 1995. The Story of Art. Prentice-Hall.

Gordon, W.J.J., 1961. Synectics, the Development of Creative Capacity. Collier-Macmillan.

Gorod, A., White, B.E., Ireland, V., Gandhi, S.J., Sauser, B., 2014. Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering. CRC Press.

Gove, R., Uzdzinski, J., 2013. A Performance-Based System Maturity Assessment Framework. Procedia Comput. Sci. 16, 688–697. https://doi.org/10.1016/j.procs.2013.01.072

Grady, J.O., 1995. System Engineering Planning and Enterprise Identity. CRC Press.

Gräßler, I., Pöhler, A., 2017. Implementation of an Adapted Holonic Production Architecture. Procedia CIRP 63, 138–143. https://doi.org/10.1016/j.procir.2017.03.176

Green, D.G., Liu, J., Abbass, H.A., 2013. Dual Phase Evolution. Springer Science & Business Media.

Greene, M.T., Gonzalez, R., Papalambros, P.Y., McGowan, A.-M., 2017. Design Thinking vs. Systems Thinking for Engineering Design: What's the Difference? 10.

Gros, C., 2015. Complex and Adaptive Dynamical Systems: A Primer. Springer.

Gross, B., Bohnacker, H., Laub, J., Lazzeroni, C., 2018. Generative Design: Visualize, Program, and Create with JavaScript in p5.js. Chronicle Books.

Grösser, S.N., 2012. Co-Evolution of Standards in Innovation Systems: The Dynamics of Voluntary and Legal Building Codes. Springer Science & Business Media.

Gumus, B., 2005. AXIOMATIC PRODUCT DEVELOPMENT LIFECYCLE.

Guzzi, P.H., Roy, S., 2020. Biological Network Analysis: Trends, Approaches, Graph Theory, and Algorithms. Elsevier.

Gwaur, 2016. Brainstomring Process.

Haberfellner, R., de Weck, O., Fricke, E., Vössner, S., 2019. Systems Engineering: Fundamentals and Applications. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-13431-0

Haddow, P.C., Tyrrell, A.M., 2011. Challenges of evolvable hardware: past, present and the path to a promising future. Genet. Program. Evolvable Mach. 12, 183–215. https://doi.org/10.1007/s10710-011-9141-6

Haik, Y., Shahin, T.M., Sivaloganathan, S., 2010. Engineering Design Process, 002 edition. ed. Cengage Learning.

Hall, A.D., 1962. A Methodology for Systems Engineering. Van Nostrand.

Hall, B.K., 2012a. Evolutionary Developmental Biology. Springer Science & Business Media.

Hall, B.K., 2012b. Evolutionary Developmental Biology (Evo-Devo): Past, Present, and Future. Evol. Educ. Outreach 5, 184–193. https://doi.org/10.1007/s12052-012-0418-x

Hall, B.K., Olson, W.M., 2003. Keywords and Concepts in Evolutionary Developmental Biology. Harvard University Press.

Hamilton, M., Hackler, W., 2009. Universal Systems Language: Lessons Learned from Apollo. Computer 41, 34–43. https://doi.org/10.1109/MC.2008.541

Hancke, T., Besant, C.B., Ristic, M., Husband, T.M., 1990. Human-Centred Technology. IFAC Proc. Vol., IFAC/IFIP/IMACS Symposium on skill Based Automated Production, Vienna, Austria, 15-17 November 23, 59–66. https://doi.org/10.1016/S1474-6670(17)52137-0

Harari, Y.N., 2018. Sapiens: A Brief History of Humankind, Reprint edition. ed. Harper Perennial, New York.

Harvey, H., Orvis, R., Rissman, J., 2018. Designing Climate Solutions: A Policy Guide for Low-Carbon Energy. Island Press.

Hatchuel, A., Weil, B., 2002. Fondements et usages d'une théorie unifiée de la conception. Ecole Mines Paris 24.

Hatchuel, P., Le Masson, P., Weil, B., 2004. CK Theory in Practice: Lessons from Industrial Applications.

Hatley, D., Hruschka, P., Pirbhai, I., 2013. Process for System Architecture and Requirements Engineering. Addison-Wesley.

Hawkins, J., 2019. Brainstorming: Become a Brainstorming Facilitator by Learning These Techniques. Scribl.

Hemenway, T., 2015. The Permaculture City: Regenerative Design for Urban, Suburban, and Town Resilience. Chelsea Green Publishing.

Hemsath, T.L., Bandhosseini, K.A., 2017. Energy Modeling in Architectural Design. Routledge.

Henchoz, N., Mirande, Y., 2014. Design for Innovative Technology: From disruption to acceptance. PPUR Presses polytechniques.

Henderson, P., 2012. Systems Engineering for Business Process Change: New Directions: Collected Papers from the EPSRC Research Programme. Springer Science & Business Media.

Herbst, J., Karagiannis, D., 2000. Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. Intell. Syst. Account. Finance Manag. 9, 67–92. https://doi.org/10.1002/1099-1174(200006)9:2<67::AID-ISAF186>3.0.CO;2-7

Herron, J.C., Freeman, S., 2013. Evolutionary Analysis, 5 edition. ed. Pearson, San Francisco, CA.

Hettema, D., 2013. Lifecycle Modeling Language (LML) SPECIFICATION. LML Steer. Comm. 64.

Hibbs, C., Jewett, S., Sullivan, M., 2009. The Art of Lean Software Development: A Practical and Incremental Approach. O'Reilly Media, Inc.

Highsmith, J., 2013. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Addison-Wesley.

Highsmith, J.A., Highsmith, J., 2002. Agile Software Development Ecosystems. Addison-Wesley Professional.

Hingston, P.F., Barone, L.C., Michalewicz, Z., 2008. Design by Evolution: Advances in Evolutionary Design. Springer.

Hirschheim, R., Klein, H.K., Lyytinen, K., 1995. Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations. Cambridge University Press.

Hirsch-Kreinsen, H., Jacobson, D., 2008. Innovation in Low-tech Firms and Industries. Edward Elgar Publishing.

Hitchins, D.K., 2008. Systems Engineering: A 21st Century Systems Methodology. John Wiley & Sons.

Hitchins, D.K., 2003. Advanced Systems Thinking, Engineering, and Management. Artech House.

Ho, L.C., 2004. Coevolution of Black Holes and Galaxies: Volume 1, Carnegie Observatories Astrophysics Series. Cambridge University Press.

Holland, J.H., Holland, P. of P. and of E.E. and C.S.J.H., Holland, S.L. in H.R.M., 1992. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press.

Hopkinson, N., Hague, R.J.M., Dickens, P.M. (Eds.), 2006. Rapid manufacturing: an industrial revolution for the digital age. John Wiley, Chichester, England.

Horden, R., 2010. Micro Architecture: Lightweight, Mobile, Ecological Buildings for the Future. by Richard Horden. Thames & Hudson, London.

Horden, R., 2008. Micro Architecture: Lightweight, Mobile and Ecological Buildings for the Future. Thames & Hudson.

Horning, J., 2009. Simple Shelters: Tents, Tipis, Yurts, Domes and Other Ancient Homes. Bloomsbury Publishing USA.

Horowitz, R., 1999. CREATIVE PROBLEM SOLVING IN ENGINEERING DESIGN. PhD Thesis 166.

Huang, P.M., Darrin, A.G., Knuth, A.A., 2012. Agile hardware and software system engineering for innovation, in: 2012 IEEE Aerospace Conference. Presented at the 2012 IEEE Aerospace Conference, IEEE, Big Sky, MT, pp. 1–10. https://doi.org/10.1109/AERO.2012.6187425

Hubka, V., 2015. Principles of Engineering Design. Elsevier.

Hueber, S.D., Weiller, G.F., Djordjevic, M.A., Frickey, T., 2010. Genes hox.

Hulett, D., 2016. Integrated Cost-Schedule Risk Analysis. CRC Press.

Humphrey, W.S., 1988. Characterizing the software process: a maturity framework. IEEE Softw. 5, 73–79. https://doi.org/10.1109/52.2014

INCOSE, 2020a. Systems Engineering Definition [WWW Document]. URL https://www.incose.org/about-systems-engineering/system-and-se-definition/systems-engineering-definition (accessed 6.21.20).

INCOSE, 2020b. Systems Engineering [WWW Document]. URL https://www.incose.org/systems-engineering (accessed 6.21.20).

INCOSE, 2015. INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities. John Wiley & Sons.

Ingi, b, 2009. A graphical reprensation of a design process using a “Crazy Concept.”

International Organization for Migration, 2014. Outlook on migration, environment and climate change. International Organization for Migration, Geneva.

Isaias, P., Issa, T., 2014. High Level Models and Methodologies for Information Systems. Springer.

Ivannikov, V.P., 1995. PROGRAMMING AND COMPUTER SOFTWARE.

Jack, H., 2013. Engineering Design, Planning, and Management. Academic Press.

Jamshidi, M., 2011. System of Systems Engineering: Innovations for the 21st Century. John Wiley & Sons.

Jarzombek, M.M., Prakash, V., 2011. A Global History of Architecture. John Wiley & Sons.

Jencks, C., 2000. Le Corbusier and the Continual Revolution in Architecture. Monacelli Press.

Jesus, G.T., Jr., M.F.C., 2018. Integration Readiness levels Evaluation and Systems Architecture: A Literature Review. Int. J. Adv. Eng. Res. Sci. 5, 73–84. https://doi.org/10.22161/ijaers.5.4.12

Jodidio, P., 2018. Cabins. Taschen.

Johnson, A., Gibson, A., 2014. Sustainability in engineering design. Elsevier, Amsterdam ; Boston.

Johnson, S.B., 2006. The Secret of Apollo: Systems Management in American and European Space Programs. JHU Press.

Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B., 2020. Characterising the Digital Twin: A systematic literature review. CIRP J. Manuf. Sci. Technol. 29, 36–52. https://doi.org/10.1016/j.cirpj.2020.02.002

Jones, J.C., 1992. Design Methods. John Wiley & Sons.

Jones, J.L., Flynn, A.M., 1993. Mobile robots: inspiration to implementation. A.K. Peters, Wellesley, Mass.

Jong, K.A.D., 2006. Evolutionary Computation: A Unified Approach. MIT Press.

JPL NASA/Caltech, 2017. “Space Fabric” Links Fashion and Engineering [WWW Document]. NASA Jet Propuls. Lab. JPL. URL https://www.jpl.nasa.gov/news/space-fabric-links-fashion-and-engineering (accessed 2.7.21).

Kahn, L., 2012. Tiny Homes: Simple Shelter : Scaling Back in the 21st Century. Shelter Publications.

Kallrath, J., 2004. Modeling Languages in Mathematical Optimization. Springer Science & Business Media.

Kalyanmoy, D., 2008. Evolutionary Design in Engineering, in: Design By Evolution.

Kamrani, A.K., Azimi, M., 2010. Systems Engineering Tools and Methods. CRC Press.

Kamrani, A.K., Azimi, M., Al-Ahmari, A.M., 2016. Methods in Product Design: New Strategies in Reengineering. CRC Press.

Kamrani, A.K., Nasr, E.A., 2010. Engineering Design and Rapid Prototyping. Springer Science & Business Media.

Kan, J.W., Gero, J.S., 2017. Quantitative Methods for Studying Design Protocols, 1st ed. 2017 edition. ed. Springer.

Kao, R.W.Y., 2010. Sustainable Economy: Corporate, Social and Environmental Responsibility. World Scientific.

Kappelman, L., 2009. The SIM Guide to Enterprise Architecture. CRC Press.

Karayanakis, N.M., 1995. Advanced System Modelling and Simulation with Block Diagram Languages. CRC Press.

Karonen, Ii., 2006. Speciation modes.

Kaza, S., Yao, L., Bhada-Tata, P., Van Woerden, F., 2018. What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050. The World Bank. https://doi.org/10.1596/978-1-4648-1329-0

Keane, P., 2018. Generative Design: The Road to Production [WWW Document]. engeering.com. URL https://www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/16973/Generative-Design-The-Road-to-Production.aspx (accessed 7.25.20).

Keister, D., 2008. Teardrops and Tiny Trailers. Gibbs Smith.

Kensek, K.M., 2014. Building Information Modeling. Routledge.

Kepes, F., 2007. Biological Networks. World Scientific.

Kihlander, I., 2009. Decision making in concept phases 58.

Killi, S.W., 2017. Additive Manufacturing. CRC Press, London.

Kimura, F., 2001. Geometric Modelling: Theoretical and Computational Basis towards Advanced CAD Applications. IFIP TC5/WG5.2 Sixth International Workshop on Geometric Modelling December 7–9, 1998, Tokyo, Japan. Springer Science & Business Media.

King, K.N., Offutt, A.J., 1991. A fortran language system for mutation-based software testing. Softw. Pract. Exp. 21, 685–718. https://doi.org/10.1002/spe.4380210704

Kleppe, A.G., Warmer, J., Warmer, J.B., Bast, W., 2003. MDA Explained: The Model Driven Architecture : Practice and Promise. Addison-Wesley Professional.

Kliman, R.M., 2016. Encyclopedia of Evolutionary Biology. Academic Press.

Kolko, J., 2010. Abductive Thinking and Sensemaking: The Drivers of Design Synthesis. Des. Issues 26, 15–28. https://doi.org/10.1162/desi.2010.26.1.15

Košir, M., 2019. Climate Adaptability of Buildings: Bioclimatic Design in the Light of Climate Change. Springer.

Koskela, L., Codinhoto, R., Tzortzopoulos, P., Kagioglou, M., 2014. The Aristotelian Proto-Theory of Design, in: Chakrabarti, A., Blessing, L.T.M. (Eds.), An Anthology of Theories and Models of Design. Springer London, London, pp. 285–303. https://doi.org/10.1007/978-1-4471-6338-1_14

Kossiakoff, A., Biemer, S.M., Seymour, S.J., Flanigan, D.A., 2020. Systems Engineering Principles and Practice. John Wiley & Sons.

Koutromanos, I., 2018. Fundamentals of Finite Element Analysis: Linear Finite Element Analysis. John Wiley & Sons.

Koza, J.R., 1994. Genetic programming as a means for programming computers by natural selection. Stat. Comput. 4, 87–112. https://doi.org/10.1007/BF00175355

Koza, J.R., Keane, Streeter, M., 2005. Routine High-Return Human-Competitive Evolvable Hardware, in: Evolutionary Machine Design: Methodologies and Applications.

Kranz, M., 2017. Building the Internet of Things: implement new business models, disrupt competitors, and transform your industry. John Wiley & Sons, Inc, Hoboken, New Jersey.

Kravtsov, Y.A., Kadtke, J.B. (Eds.), 1996. Predictability of Complex Dynamical Systems. Springer, Berlin ; New York.

Krohn, R., Metcalf, D., 2020. mHealth From Smartphones to Smart Systems, 1st ed. HIMSS Publishing. https://doi.org/10.4324/9780367648008

Kruchten, P., 2004. The Rational Unified Process: An Introduction. Addison-Wesley Professional.

Kuncoro, W., Suriani, W.O., 2018. Achieving sustainable competitive advantage through product innovation and market driving. Asia Pac. Manag. Rev. 23, 186–192. https://doi.org/10.1016/j.apmrv.2017.07.006

Kunda, G., 2009. Engineering Culture: Control and Commitment in a High-Tech Corporation. Temple University Press.

Kwinter, S., 2017. Soupergreen!: Souped-up Green Architecture. Actar Publishers.

Lapham, M.A., Bandor, M., Wrubel, E., 2014. Agile Methods and Request for Change (RFC): Observations from DoD Acquisition Programs: Defense Technical Information Center, Fort Belvoir, VA. https://doi.org/10.21236/ADA609878

Larman, C., 2004. Agile and Iterative Development: A Manager's Guide. Addison-Wesley Professional.

Larman, C., Basili, V.R., 2003. Iterative and incremental developments. a brief history. Computer 36, 47–56. https://doi.org/10.1109/MC.2003.1204375

Larson, E., 2010. Project Management: The Managerial Process, 5 edition. ed. McGraw Hill, New York.

Lawson, B., 2014. How Designers Think: The Design Process Demystified. Elsevier.

Leach, N., Yuan, P.F., 2018. Computational Design. Tongji University Press Co., Ltd, Shanghai.

Lee, E.A., 2020. The Coevolution - the Entwined Futures of Humans and Machines. MIT Press.

Lee, I., Lee, K., 2015. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Bus. Horiz. 58, 431–440. https://doi.org/10.1016/j.bushor.2015.03.008

Lees-Maffei, G., 2013. Writing Design: Words and Objects. Berg.

Leondes, C.T., 2019. Computer-Aided Design, Engineering, and Manufacturing: Systems Techniques and Applications, Volume V, The Design of Manufacturing Systems. CRC Press.

Letcher, T.M., 2008. Future Energy: Improved, Sustainable and Clean Options for our Planet. Elsevier Science.

Levi, P., Kernbach, S., 2010. Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution. Springer Science & Business Media.

Levine, A.S., 1982. Managing NASA in the Apollo Era. Scientific and Technical Information Branch, National Aeronautics and Space Administration.

Levine, W.S., 1996. The Control Handbook. CRC Press.

Lewis, N.C., 2020. Design and Order: Perceptual Experience of Built Form - Principles in the Planning and Making of Place. John Wiley & Sons.

Liker, J.K., Liker, J.K., Ettlie, J.E., Campbell, J.C., 1995. Engineered in Japan: Japanese Technology-management Practices. Oxford University Press.

Lindemann, U., 2009. Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden. Springer-Verlag.

Liou, F.W., 2007. Rapid Prototyping and Engineering Applications: Dekker Mechanical Engineering. Taylor & Francis Ltd., Hoboken.

Liu, D., 2015. System Design Principles and Methods. CRC Press LLC Taylor & Francis Group [distributor, Boca Raton; Abingdon.

Lockwood, T., 2010. Design Thinking: Integrating Innovation, Customer Experience, and Brand Value. Simon and Schuster.

Long, D., Scott, Z., 2011. A Primer for Model-Based Systems Engineering. Lulu.com.

Long, J.H., Aaron, E., Doncieux, S., 2018. Evolvability, Environments, Embodiment, & Emergence in Robotics. Frontiers Media SA.

Long, J.M., 2011. Integration readiness levels, in: 2011 Aerospace Conference. Presented at the 2011 IEEE Aerospace Conference, IEEE, Big Sky, USA, pp. 1–9. https://doi.org/10.1109/AERO.2011.5747629

LUMA Institute, 2012. Innovating for People: Handbook of Human-centered Design Methods. LUMA Institute, LLC.

Lurie, J., Goodman, D.C., Wennberg, J.E., 2002. Benchmarking the Future Generalist Workforce.

Luzeaux, D., Ruault, J.-R., Wippler, J.-L., 2013. Large-scale Complex System and Systems of Systems. John Wiley & Sons.

Lyle, J.T., 1996. Regenerative Design for Sustainable Development. John Wiley & Sons.

Ma, Y., Morosuk, T., Luo, J., Liu, M., Liu, J., 2020. Superstructure design and optimization on supercritical carbon dioxide cycle for application in concentrated solar power plant. Energy Convers. Manag. 206, 112290. https://doi.org/10.1016/j.enconman.2019.112290

Macdonald, A.J., 2019. High Tech Architecture: A Style Reconsidered. The Crowood Press.

Macdonald, E., Salas, R., Espalin, D., Perez, M., Aguilera, E., Muse, D., Wicker, R.B., 2014. 3D Printing for the Rapid Prototyping of Structural Electronics. IEEE Access 2, 234–242. https://doi.org/10.1109/ACCESS.2014.2311810

Machado, C., Davim, J.P., 2020. Industry 4.0: Challenges, Trends, and Solutions in Management and Engineering. CRC Press.

Madani, F., Wang, B., Wang, X., Wang, L., White, C., 2014. Design Structure Matrix. p. 53. https://doi.org/10.1007/978-3-319-02973-3_3

Madu, C.N., 2006. House of Quality (QFD) in a Minute. Chi Publishers Inc.

Madureira, N.L., 2014. Key Concepts in Energy. Springer.

Magrab, E.B., 2014. An Engineer's Guide to Mathematica. John Wiley & Sons.

Magrab, E.B., Magrab, E.B. (Eds.), 2010. Integrated product and process design and development: the product realization process, 2nd ed. ed. CRC Press, Boca Raton, FL.

Mahadevan, B., 2010. Operations Management: Theory and Practice. Pearson Education India.

Maier, M.W., 2009. The Art of Systems Architecting. CRC Press.

Malach-Pines, T. late A., Özbilgin, M.F., 2010. Handbook of Research on High-Technology Entrepreneurs. Edward Elgar Publishing.

Malyszkz, G. png: O. uploader was G. at en wikipediaLater versions were uploaded by A. at en wikipedia derivative work:, 2011. Gantt Chart.

Mang, P., Haggard, B., Regenesis, 2016. Regenerative Development and Design: A Framework for Evolving Sustainability. John Wiley & Sons.

Mang, P., Reed, B., 2017. Update Regenerative Development and Design 2nd edition.

Mankins, J.C., 2009. Technology readiness assessments: A retrospective. Acta Astronaut. 65, 1216–1223. https://doi.org/10.1016/j.actaastro.2009.03.058

Mankins, J.C., 1995. TECHNOLOGY READINESS LEVELS. NASA 5.

Manske, M., 2008. Functional Flow Block Diagram Format.

Marchal, V., Dellink, R., Van Vuuren, D., Clapp, C., Chateau, J., Lanzi, E., Magne, B., Van Vliet, J., 2012. OECD environmental outlook to 2050: the consequences of inaction. Int. J. Sustain. High. Educ. 13, ijshe.2012.24913caa.010. https://doi.org/10.1108/ijshe.2012.24913caa.010

Marcus, A., 2014. Design, User Experience, and Usability: Theories, Methods, and Tools for Designing the User Experience: Third International Conference, DUXU 2014, Held as Part of the HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part I. Springer.

Mareels, I., Polderman, J.W., 1996. Adaptive Systems: An Introduction. Springer Science & Business Media.

Margolin, V., Margolin, S., 2002. A “Social Model” of Design: Issues of Practice and Research. Des. Issues 18, 24–30. https://doi.org/10.1162/074793602320827406

Marin-Sanguino, A., Vera, J., Alves, R., 2019. Foundations of Theoretical Approaches in Systems Biology. Frontiers Media SA.

Markus, M., 2008. Italiano: Roma - Tempietto del Bramante a S. Pietro in Montorio.

Martin, J.N., 1996. Systems Engineering Guidebook: A Process for Developing Systems and Products. CRC Press.

Masson-Delmotte, V., Zhai, P., Porter, H.O., Roberts, D., Skea, J., Shukla, P.R., 2019. Global Warming of 1.5 ºC —.

Massotte, P., Corsi, P., 2015. Operationalizing Sustainability. John Wiley & Sons.

Matulevičius, R., Dijkman, R., 2018. Advanced Information Systems Engineering Workshops: CAiSE 2018 International Workshops, Tallinn, Estonia, June 11-15, 2018, Proceedings.

Maulana, M.I.I.B.M., Al-Ashaab, A., Flisiak, J.W., Araci, Z.C., Lasisz, P.W., Shehab, E., Beg, N., Rehman, A., 2017. The Set-based Concurrent Engineering Application: A Process of Identifying the Potential Benefits in the Surface Jet Pump Case Study. Procedia CIRP 60, 350–355. https://doi.org/10.1016/j.procir.2017.01.026

Mayer, R.J., 2009. IDEF Methods.

Mayfield, J.E., 2013. The Engine of Complexity: Evolution as Computation. Columbia University Press.

Mayhew, P.J., 2006. Discovering Evolutionary Ecology: Bringing Together Ecology and Evolution. OUP Oxford.

Mazzoleni, I., 2013. Architecture Follows Nature-Biomimetic Principles for Innovative Design. CRC Press.

McCalman, J., Paton, P.R.A., Siebert, S., 2015. Change Management: A Guide to Effective Implementation. SAGE.

McCormack, J., 2008. Evolutionary Design in Art, in: Design By Evolution.

McDonough, W., Braungart, M., 2013. The Upcycle: Beyond Sustainability--Designing for Abundance. Macmillan.

McDonough, W., Braungart, M., 2010. Cradle to Cradle: Remaking the Way We Make Things. Farrar, Straus and Giroux.

Mdd, 2008. Function block.

MDD, 2007. Systems Engineering and Verification.

Menges, A., Ahlquist, S., 2011. Computational Design Thinking: Computation Design Thinking, 1 edition. ed. Wiley, Chichester.

Merriam Webster, 2020. Definition of SYSTEM [WWW Document]. URL https://www.merriam-webster.com/dictionary/system (accessed 8.1.20).

Merten, M., Gross, H.-M., 2008. Highly Adaptable Hardware Architecture for Scientific and Industrial Mobile Robots, in: 2008 IEEE Conference on Robotics, Automation and Mechatronics. Presented at the 2008 IEEE Conference on Robotics, Automation and Mechatronics (RAM), IEEE, Chengdu, China, pp. 1130–1135. https://doi.org/10.1109/RAMECH.2008.4681459

Micaëlli, J.-P., Forest, J., Bonjour, E., Loise, D., 2016. Frugal innovation or frugal renovation: how can western designers adopt frugal engineering? J. Innov. Econ. Manag. 21, 39.

Michael, S., Kin Wai, Lin, W., Yin, 2018. Practice and Progress in Social Design and Sustainability. IGI Global.

Miglani, G.S., 2010. Developmental Genetics. I. K. International Pvt Ltd.

Miller, J.H., Page, S.E., 2009. Complex Adaptive Systems: An Introduction to Computational Models of Social Life. Princeton University Press.

Milner, T., Volas, M., Sanders, A., 2013. Systems Engineering Methodology for Linking Requirements to Design Complexity and Manufacturing Trade Space Constraints. Procedia Comput. Sci. 16, 947–956. https://doi.org/10.1016/j.procs.2013.01.099

Minelli, A., 2018. Plant Evolutionary Developmental Biology: The Evolvability of the Phenotype. Cambridge University Press.

Mitchell, I., 2015. Scrum Framework.

Mitchell, M., 2006. Complex systems: Network thinking. Artif. Intell., Special Review Issue 170, 1194–1212. https://doi.org/10.1016/j.artint.2006.10.002

Mittal, S., Martín, J.L.R., 2018. Netcentric System of Systems Engineering with DEVS Unified Process. CRC Press.

Moffett, M., Fazio, M.W., Wodehouse, L., 2004. A World History of Architecture. McGraw-Hill.

Monostori, L., Váncza, J., Kumara, S.R.T., 2006. Agent-Based Systems for Manufacturing. CIRP Ann. 55, 697–720. https://doi.org/10.1016/j.cirp.2006.10.004

Montecchi, T., Russo, D., 2015. Knowledge based Approach for Formulating TRIZ Contradictions. Procedia Eng. 131, 451–463. https://doi.org/10.1016/j.proeng.2015.12.440

Morris, D., Evans, D., Green, P., Theaker, C., 2012. Object Oriented Computer Systems Engineering. Springer Science & Business Media.

Muller, G., Klever, J.D., Bjørnsen, H., Pennotti, M., 2011. Researching the application of Pugh Matrix in the sub-sea equipment industry.

Müller, I.L., 2017. Your Private Sky: R. Buckminster Fuller: The Art of Design Science. Lars Müller.

Nagarajan, R., Ruckenstein, E., 1991. Theory of surfactant self-assembly: a predictive molecular thermodynamic approach. Langmuir 7, 2934–2969. https://doi.org/10.1021/la00060a012

Nair, V.N., Abraham, B., MacKay, J., Box, G., Kacker, R.N., Lorenzen, T.J., Lucas, J.M., Myers, R.H., Vining, G.G., Nelder, J.A., Phadke, M.S., Sacks, J., Welch, W.J., Shoemaker, A.C., Tsui, K.L., Taguchi, S., Jeff Wu, C.F., 1992. Taguchi's Parameter Design: A Panel Discussion. Technometrics 34, 127–161. https://doi.org/10.1080/00401706.1992.10484904

Nakatsu, R., 2010. Diagrammatic reasoning in AI. Wiley, Hoboken, N.J.

Narayanan, V.K., O'Connor, G.C., 2010. Encyclopedia of Technology and Innovation Management. John Wiley & Sons.

NASA, 2016. Final Report of the NASA Technology Readiness Assessment (TRA) Study Team 63.

NASA, 2015. NASA's Exploration Systems Architecture Studio.

NASA, 2014. NASA TRL Meter.

NASA, 2007. NASA Systems Engineering Handbook. NASA 297.

Natali, C., 2013. Aristotle: His Life and School. Princeton University Press.

Nedjah, N., Mourelle, L. de M., 2005a. Fuzzy Systems Engineering: Theory and Practice. Springer Science & Business Media.

Nedjah, N., Mourelle, L. de M., 2005b. Evolutionary Machine Design: Methodology & Applications. Nova Publishers.

Nei, M., 2013. Mutation-Driven Evolution. OUP Oxford.

Nelson, P., 2020. New Treehouses of the World. Abrams.

Neri, F., Cotta, C., Moscato, P., 2011. Handbook of Memetic Algorithms. Springer Science & Business Media.

Neufert, E., Neufert, P., 2000. Architects' Data. Wiley.

Nicholas, J.M., 2004. Project Management for Business and Engineering: Principles and Practice. Elsevier.

Nolfi, S., Floreano, D., Floreano, D.D., 2000. Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines. MIT Press.

Nomura, T., Asai, Y., 2010. Harnessing Biological Complexity: An Introduction to Computational Physiology. Springer Science & Business Media.

Norberg, J., Cumming, G., 2008. Complexity Theory for a Sustainable Future. Columbia University Press.

Norman, D.A., Draper, S.W., 2018. User Centered System Design: New Perspectives on Human-Computer Interaction. CRC Press LLC.

Nyqvist, 2016. Fidgeting for Creativity. Lund University.

Ochoa, D.A., Vonau, W., Ewert, M.K., 2009. A Comparison between One- and Two-Loop ATCS Architectures Proposed for CEV. SAE Int. J. Aerosp. 4, 344–350. https://doi.org/10.4271/2009-01-2458

Ockerman, S., Reindl, S., 2019. Mastering Professional Scrum: A Practitioners Guide to Overcoming Challenges and Maximizing the Benefits of Agility. Addison-Wesley Professional.

Ohare, B., 2015. A signal flow graph for the negative feedback amplifier. Patterned after D'Amico et al.

Ohnishi, K., Okano, J., Koeppen, M., 2017. Building a Simulation Model for Distributed Human-Based Evolutionary Computation, in: Tan, Y., Takagi, H., Shi, Y. (Eds.), Advances in Swarm Intelligence, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 40–49. https://doi.org/10.1007/978-3-319-61824-1_5

OMG, 2019. OMG Meta Object Facility (MOF) Core Specification.

OMG, 2015. XML Metadata Interchange (XMI) Specification.

O'Neill, M., Ryan, C., 2012. Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Springer Science & Business Media.

Osborn, A.F., 1993. Applied Imagination: Principles and Procedures of Creative Problem-solving. Creative Education Foundation.

Oxford University Press, 2020. maturation noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com [WWW Document]. URL https://www.oxfordlearnersdictionaries.com/definition/english/maturation (accessed 12.31.20).

Oxford University Press, 2003. Shorter Oxford English Dictionary, 2003rd ed. Oxford University Press.

Paetz, P., 2014. Disruption by Design: How to Create Products that Disrupt and then Dominate Markets. Apress.

Pahl, G., Beitz, W., Feldhusen, J., Grote, K.H., 2007. Engineering Design: A Systematic Approach, 3rd edition. ed. Springer, London.

Pande, P.S., Holpp, L., 2001. What Is Six Sigma? McGraw Hill Professional.

Pande, P.S., Neuman, R.P., Cavanagh, R.R., 2000. The Six Sigma way: how GE, Motorola, and other top companies are honing their performance. McGraw-Hill, New York.

Parcell, S., 2012. Four Historical Definitions of Architecture. McGill-Queen's Press - MQUP.

Park, G.-J., 2007. Analytic Methods for Design Practice. Springer Science & Business Media.

Parker, G.G., Alstyne, M.W.V., Choudary, S.P., 2016. Platform Revolution: How Networked Markets Are Transforming the Economy and How to Make Them Work for You. W. W. Norton & Company.

Parnell, G.S., Driscoll, P.J., Henderson, D.L., 2011. Decision Making in Systems Engineering and Management. John Wiley & Sons.

Pasquier, H., Cruzen, C.A., Schmidhuber, M., Lee, Y.H., 2019. Space Operations: Inspiring Humankind's Future. Springer.

Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V., 1993. Capability Maturity Model for Software (Version 1.1) 82.

Pearson, D.W., Steele, N.C., Albrecht, R.F., 2012. Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Alès, France, 1995. Springer Science & Business Media.

Pedrycz, W., Gomide, F., 2007. Fuzzy Systems Engineering: Toward Human-Centric Computing. John Wiley & Sons.

Persse, J.R., 2001. Implementing the Capability Maturity Model. Wiley.

Petro, G., 2017. The Need For Speed: Time-To-Market Acceleration Is The Ultimate Retail Differentiator [WWW Document]. Forbes. URL https://www.forbes.com/sites/gregpetro/2017/10/06/the-need-for-speed-time-to-market-acceleration-is-the-ultimate-retail-differentiator/ (accessed 10.30.20).

Philippe, B., 2020. The Age of Low Tech: Towards a Technologically Sustainable Civilization. Policy Press.

Phillips, D., 2004. The Software Project Manager's Handbook: Principles That Work at Work. John Wiley & Sons.

Pianka, E.R., 2011. Evolutionary Ecology. Eric R. Pianka.

Pierce, B.C., Pennsylvania), B.C. (Professor P., University of, 2002. Types and Programming Languages. MIT Press.

Plattner, H., Meinel, C., Leifer, L. (Eds.), 2010. Design Thinking: Understand – Improve – Apply, 2011 edition. ed. Springer.

Plowman, J., 2019. Unreal Engine Virtual Reality Quick Start Guide: Design and Develop immersive virtual reality experiences with Unreal Engine 4. Packt Publishing Ltd.

Polit Casillas, R., Howe, S.A., 2013. Virtual Construction of Space Habitats: Connecting Building Information Models (BIM) and SysML, in: AIAA SPACE 2013 Conference and Exposition. Presented at the AIAA SPACE 2013 Conference and Exposition, American Institute of Aeronautics and Astronautics, San Diego, CA. https://doi.org/10.2514/6.2013-5508

Polit-Casillas, R., 2008. Sustainable High School - PFC.

Pollio, V., 2018. Vitruvius, the Ten Books on Architecture. FRANKLIN CLASSICS TRADE Press.

Popov, G., Lyon, B.K., Hollcroft, B., 2016. Risk Assessment: A Practical Guide to Assessing Operational Risks. John Wiley & Sons.

Poppendieck, M., Poppendieck, T., 2003. Lean Software Development: An Agile Toolkit. Addison-Wesley.

Potter, M.A., Jong, K.A.D., 2000. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. Evol. Comput. 8, 1–29. https://doi.org/10.1162/106365600568086

Prasad, B., 1995. Sequential versus Concurrent Engineering—An Analogy. Concurr. Eng. 3, 250–255. https://doi.org/10.1177/1063293X9500300401

Prince, G.M., 2012. The practice of creativity: a manual for dynamic group problem-solving. Echo Point Books & Media, s.l.

Proulx, S., Promislow, D., Phillips, P., 2005. Network thinking in ecology and evolution. Trends Ecol. Evol. 20, 345–353. https://doi.org/10.1016/j.tree.2005.04.004

Pugh, S., 1986. Design activity models: worldwide emergence and convergence. https://doi.org/10.1016/0142-694X(86)90054-2

PWC, 2020. Workforce of the future 42.

Radjou, N., Prabhu, J.C., 2014. Frugal innovation: how to do more with less, First edition. ed. PublicAffairs, New York.

Rainey, L.B., Tolk, A., 2015. Modeling and Simulation Support for System of Systems Engineering Applications. John Wiley & Sons.

Ramnath, S., Dathan, B., 2010. Object-Oriented Analysis and Design. Springer Science & Business Media.

Ramos, A.L., Ferreira, J.V., Barcelo, J., 2012. Model-Based Systems Engineering: An Emerging Approach for Modern Systems. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 42, 101–111. https://doi.org/10.1109/TSMCC.2011.2106495

Rampersad, G., 2020. Robot will take your job: Innovation for an era of artificial intelligence. J. Bus. Res. 116, 68–74. https://doi.org/10.1016/j.jbusres.2020.05.019

Rao, P.N., 2004. CAD/CAM: Principles and Applications. Tata McGraw-Hill Education.

Rawlinson, J.G., 2017. Creative Thinking and Brainstorming. Routledge.

Ray, T.S., 1994. Evolution, complexity, entropy and artificial reality. Phys. Nonlinear Phenom. 75, 239–263. https://doi.org/10.1016/0167-2789(94)90286-0

Rayna, T., Striukova, L., 2016. From rapid prototyping to home fabrication: How 3D printing is changing business model innovation. Technol. Forecast. Soc. Change 102, 214–224. https://doi.org/10.1016/j.techfore.2015.07.023

Rechenberg, I., 1989. Evolution Strategy: Nature's Way of Optimization, in: Bergmann, H.W. (Ed.), Optimization: Methods and Applications, Possibilities and Limitations, Lecture Notes in Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 106–126. https://doi.org/10.1007/978-3-642-83814-9_6

Retamosa, A., 2020. Etymology of Architect.

Reynolds, R.G., 2018. Culture on the Edge of Chaos: Cultural Algorithms and the Foundations of Social Intelligence. Springer.

Richards, D., Dunn, N., Amos, M., 2012. An evo-devo approach to architectural design, in: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference - GECCO '12. Presented at the the fourteenth international conference, ACM Press, Philadelphia, Pennsylvania, USA, p. 569. https://doi.org/10.1145/2330163.2330244

Richardson, P., 2009. XS Extreme: Big Ideas, Small Buildings. Thames & Hudson.

Richardson, P., 2007. XS: Small Structures, Green Architecture. Universe, New York, NY.

Ritchey, T., 2002. General Morphological Analysis 10.

Rivkin, L.R., Santangelo, J.S., Alberti, M., Aronson, M.F.J., de Keyzer, C.W., Diamond, S.E., Fortin, M., Frazee, L.J., Gorton, A.J., Hendry, A.P., Liu, Y., Losos, J.B., MacIvor, J.S., Martin, R.A., McDonnell, M.J., Miles, L.S., Munshi-South, J., Ness, R.W., Newman, A.E.M., Stothart, M.R., Theodorou, P., Thompson, K.A., Verrelli, B.C., Whitehead, A., Winchell, K.M., Johnson, M.T.J., 2019. A roadmap for urban evolutionary ecology. Evol. Appl. 12, 384–398. https://doi.org/10.1111/eva.12734

Robert, C.A., Curedale, R., 2013. Design Research Methods: 150 Ways to Inform Design. Design Community College Incorporated.

Robertson, J.O., Chilingar, G.V., Sorokhtin, O.G., Sorokhtin, N.O., Long, W., 2018. The Evolution of Earth's Climate, 1 edition. ed. Wiley-Scrivener, Hoboken, New Jersey : Salem, Massachusetts.

Rodenacker, W.G., 2013. Methodisches Konstruieren: Grundlagen, Methodik, praktische Beispiele. Springer-Verlag.

Rodrigues, E., Amaral, A.R., Gaspar, A.R., Gomes, Á., 2015. An Approach to Urban Quarter Design Using Building Generative Design and Thermal Performance Optimization. Energy Procedia 78, 2899–2904. https://doi.org/10.1016/j.egypro.2015.11.662

Roke, R., 2017. Mobitecture: Architecture on the Move. Phaidon Press.

Roke, R., 2016. Nanotecture: Tiny Built Things. Phaidon Press.

Rosenbrock, H.H. (Ed.), 1989. Designing human-centred technology: a cross-disciplinary project in computer-aided manufacturing, The Springer series on artificial intelligence and society. Springer-Verlag, London ; New York.

Roser, M., 2013. Future Population Growth. Our World Data.

Roth, L.M., 1994. Understanding Architecture: Its Elements, History and Meaning. Herbert Press.

Rowan, D., 2019. Non-Bullshit Innovation: Radical Ideas from the World's Smartest Minds. Random House.

Roy, R.K., 1990. A Primer on the Taguchi Method. Society of Manufacturing Engineers.

Rozvany, G.I.N., Lewinski, T., 2013. Topology Optimization in Structural and Continuum Mechanics. Springer Science & Business Media.

Rubin, J., Chisnell, D., Dawsonera (Servicio en línea), 2008. Handbook of usability testing how to plan, design, and conduct effective tests. Wiley Pub., Indianapolis, IN.

Rufe, P.D., 2013. Fundamentals of Manufacturing, Third Edition. Society of Manufacturing Engineers.

Safavi, E., 2016. Collaborative Multidisciplinary Design Optimization for Conceptual Design of Complex Products. Linköping University Electronic Press.

Saha, P. (Ed.), 2014. A Systemic Perspective to Managing Complexity with Enterprise Architecture:, Advances in Business Information Systems and Analytics. IGI Global. https://doi.org/10.4018/978-1-4666-4518-9

Salomone, T.A., 2019. What Every Engineer Should Know about Concurrent Engineering. Routledge.

Salustri, F.A., 2014. Reformulating CK Theory with an Action Logic, in: Gero, J.S. (Ed.), Design Computing and Cognition '12. Springer Netherlands, Dordrecht, pp. 433–450. https://doi.org/10.1007/978-94-017-9112-0_24

Samuel, A., Weir, J., 1999. Introduction to Engineering Design. Elsevier.

Sass, S.L., 2011. The Substance of Civilization: Materials and Human History from the Stone Age to the Age of Silicon. Skyhorse Publishing Inc.

Satzinger, J.W., Jackson, R.B., Burd, S.D., 2008. Systems Analysis and Design in a Changing World. Cengage Learning EMEA.

Sauser, B., Boardman, J., Verma, D., 2010. Systomics: Toward a Biology of System of Systems. IEEE Trans. Syst. Man Cybern. Part A 40, 803–814. https://doi.org/10.1109/TSMCA.2010.2048024

Schneider, J., Kirkpatrick, S., 2007. Stochastic Optimization. Springer Science & Business Media.

Schöfer, M., Maranzana, N., Aoussat, A., Gazo, C., Bersano, G., 2015. The Value of TRIZ and its Derivatives for Interdisciplinary Group Problem Solving. Procedia Eng. 131, 672–681. https://doi.org/10.1016/j.proeng.2015.12.353

Schwarzbach, M., Wlach, S., Laiacker, M., 2015. Modifying a scientific flight control system for balloon launched UAV missions, in: 2015 IEEE Aerospace Conference. Presented at the 2015 IEEE Aerospace Conference, IEEE, Big Sky, MT, pp. 1–10. https://doi.org/10.1109/AERO.2015.7119055

Schweisguth, F., Corson, F., 2019. Self-Organization in Pattern Formation. Dev. Cell 49, 659–677. https://doi.org/10.1016/j.devcel.2019.05.019

Schweitzer, F., 1997. Self-Organization of Complex Structures: From Individual to Collective Dynamics. CRC Press.

Seider, W.D., Lewin, D.R., Seader, J.D., Widagdo, S., Gani, R., Ng, K.M., 2016. Product and Process Design Principles: Synthesis, Analysis and Evaluation, 4th Edition. Wiley.

Sekhar, T.G.T.S.C., 2005. Communication Theory. Tata McGraw-Hill Education.

Sendler, U., Wawer, V., 2008. CAD and PDM: Optimizing Processes by Integrating Them. Hanser.

Senge, P.M., 2010. The Fifth Discipline: The Art & Practice of The Learning Organization. Crown.

Shea, K., Aish, R., Gourtovaia, M., 2005. Towards integrated performance-driven generative design tools. Autom. Constr., Education and Research in Computer Aided Architectural Design in Europe (eCAADe 2003), Digital Design 14, 253–264. https://doi.org/10.1016/j.autcon.2004.07.002

Shelly, G., Rosenblatt, H.J., 2009. Systems Analysis and Design. Cengage Learning.

Shen, W., 2019. Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing. CRC Press.

Sheng, I.L.S., Kok-Soo, T., 2010. Eco-Efficient Product Design Using Theory of Inventive Problem Solving (TRIZ) Principles. https://doi.org/10.3844/ajassp.2010.852.858

Shieh, M.-D., Li, Y., Yang, C.-C., 2018. Comparison of multi-objective evolutionary algorithms in hybrid Kansei engineering system for product form design. Adv. Eng. Inform. 36, 31–42. https://doi.org/10.1016/j.aei.2018.02.002

Shore, James, Warden, S., Shore, Jim, 2008. The Art of Agile Development. O'Reilly Media, Inc.

Shukla, M., Krishnan, S., 2016. Concepts in Engineering Design. Notion Press.

Siegal, J., 2002. Mobile: The Art of Portable Architecture. Princeton Architectural Press.

Siegmund, P., Abermann, J., Cazenave, A., Derksen, C., Garreau, A., Howel, S., Isensee, K., Kennedy, J., Mottram, R., Nitu, R., Ramasamy, S., Sparrow, M., Tarasova, O., Trewin, B., Ziese, M., 2019. The Global Climate in 2015–2019.

Silva, F., Correia, L., Christensen, A., 2016. Evolutionary Robotics. Scholarpedia 11, 33333. https://doi.org/10.4249/scholarpedia.33333

Singer, D., Doerry, N., BUCKLEY, M., 2009. What Is Set-Based Design? Nav. Eng. J. 121, 31–43. https://doi.org/10.1111/j.1559-3584.2009.00226.x

Singhania, P., Gangopadhya, N., Tharak, A.G., Bhat, R., Vasilieff, S., 2019. 2020 Global Marketing Trends.

Sivanandam, S.N., Deepa, S.N., 2007. Introduction to Genetic Algorithms. Springer Science & Business Media.

Sloman, J., Guest, J., Garratt, D., 2018. Economics, 10th New edition edition. ed. Pearson Education UK, New York.

Smil, V., 2018. Energy and Civilization: A History, Reprint edition. ed. The MIT Press, Cambridge, Massachusetts.

Smith, D.K., Tardif, M., 2009. Building Information Modeling: A Strategic Implementation Guide for Architects, Engineers, Constructors, and Real Estate Asset Managers. John Wiley & Sons.

Smith, E.A., 2017. Evolutionary Ecology and Human Behavior. Routledge.

Smith, P.F., 2006. Architecture in a Climate of Change, 2 edition. ed. Routledge.

Smith, S., 2018. Two billion homes needed over next 80 years, studies show [WWW Document]. The Independent. URL http://www.independent.co.uk/life-style/design/housing-crisis-global-population-increase-two-billion-new-homes-80-years-end-of-century-a8245906.html (accessed 6.13.20).

Snee, R.D., Hoerl, R.W., 2003. Leading Six Sigma: a step-by-step guide based on experience with GE and other Six Sigma companies, Financial Times Prentice Hall books. Financial Times Prentice Hall, Upper Saddle River, NJ.

Sobek, D., Ward, A.C., Liker, J., 1999. Toyota's Principles of Set-Based Concurrent Engineering. Sloan Manage. Rev. 40.

Soenen, R., Olling, G.J., 2016. Advanced CAD/CAM Systems: State-of-the-Art and Future Trends in Feature Technology. Springer.

Solée, R.V., Valverde, S., Casals, M.R., Kauffman, S.A., Farmer, D., Eldredge, N., 2013. The evolutionary ecology of technological innovations. Complexity 18, 15–27. https://doi.org/10.1002/cplx.21436

Soro, A., Brereton, M., Roe, P. (Eds.), 2019. Social Internet of Things, Internet of Things. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-94659-7

Squires, A., Pyster, A., Sauser, B., Olwell, D., Enck, S., Gelosh, D., Anthony, J., 2010. Applying Systems Thinking via Systemigrams(TM) for Defining the Body of Knowledge and Curriculum to Advance Systems Engineering (BKCASE) Project: Defense Technical Information Center, Fort Belvoir, VA. https://doi.org/10.21236/ADA522654

Srinivas, M., Patnaik, L.M., 1994. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybern. 24, 656–667. https://doi.org/10.1109/21.286385

Steenhuis, H.-J., Bruijn, E.J., 2006. High technology revisited: definition and position. https://doi.org/10.1109/ICMIT.2006.262389

Stjepandić, J., Wognum, N., Verhagen, W.J.C., 2015. Concurrent Engineering in the 21st Century: Foundations, Developments and Challenges. Springer.

Stober, T., Hansmann, U., 2009. Agile Software Development: Best Practices for Large Software Development Projects. Springer Science & Business Media.

Suireg, A., 1981. Methodology of mechanical systems design.

Takagi, H., 2001. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. Proc. IEEE 89, 1275–1296. https://doi.org/10.1109/5.949485

Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (Eds.), 2011. Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-19893-9

Tamke, M., Nicholas, P., Zwierzycki, M., 2018. Machine learning for architectural design: Practices and infrastructure. Int. J. Archit. Comput. 16, 123–143. https://doi.org/10.1177/1478077118778580

Tan, W., Ramirez-Marquez, J., Sauser, B., 2011. A probabilistic approach to system maturity assessment. Syst. Eng. 14, 279–293. https://doi.org/10.1002/sys.20179

Tan, Y., Takagi, H., Shi, Y., 2017. Advances in Swarm Intelligence: 8th International Conference, ICSI 2017, Fukuoka, Japan, July 27 – August 1, 2017, Proceedings, Part I. Springer.

Tao, F., Zhang, M., Nee, A.Y.C., 2019. Digital Twin Driven Smart Manufacturing. Academic Press.

Thompson, D., 2020. Ace the Trading Systems Engineer Interview (C++ Edition): Insider's Guide to Top Tech Jobs in Finance. Dennis Thompson.

Thompson, J.N., 2009. The Coevolutionary Process. University of Chicago Press.

Thompson, J.N., 2005. The Geographic Mosaic of Coevolution. University of Chicago Press.

Tiller, M., 2012. Introduction to Physical Modeling with Modelica. Springer Science & Business Media.

Tolio, T., Ceglarek, D., ElMaraghy, H.A., Fischer, A., Hu, S.J., Laperrière, L., Newman, S.T., Váncza, J., 2010. SPECIES—Co-evolution of products, processes and production systems. CIRP Ann. 59, 672–693. https://doi.org/10.1016/j.cirp.2010.05.008

Tomaschek, K., Olechowski, A., Eppinger, S., Joglekar, N., 2016. A Survey of Technology Readiness Level Users. INCOSE Int. Symp. 26, 2101–2117. https://doi.org/10.1002/j.2334-5837.2016.00283.x

Townsend, A.M., 2020. Ghost Road: Beyond the Driverless Car. W. W. Norton & Company.

Turton, R., Bailie, R.C., Whiting, W.B., Shaeiwitz, J.A., 2008. Analysis, Synthesis and Design of Chemical Processes. Pearson Education.

Tyulin, A., Chursin, A., 2020. The New Economy of the Product Life Cycle: Innovation and Design in the Digital Era. Springer Nature.

Ullman, D., 2009. The Mechanical Design Process, 4 edition. ed. McGraw-Hill Education, Boston.

Ullman, D., Wood, S., Craig, D., 1990. The importance of drawing in the mechanical design process. Comput. Graph. 14, 263–274. https://doi.org/10.1016/0097-8493(90)90037-X

Ullman, D.G., 2010. The mechanical design process, 4th ed. ed, McGraw-Hill series in mechanical engineering. McGraw-Hill Higher Education, Boston.

UN, 2019. Affordable Housing and Social Protection Systems for All to Address Homelessness.

United Nations, Department of Economic and Social Affairs, Population Division, 2019. World population prospects Highlights, 2019 revision Highlights, 2019 revision.

Valacich, J.S., George, J.F., Valacich, J.S., 2017. Modern systems analysis and design, Eighth edition. ed. Pearson, Boston.

Valencia, V., Colombi, J., Thal, A., Sitzabee, W., 2011. Asset Management: A Systems Perspective. 61st Annu. IIE Conf. Expo Proc.

Valerdi, R., Boehm, B.W., Reifer, D.J., 2003. 3.1.1 COSYSMO: A Constructive Systems Engineering Cost Model Coming of Age. INCOSE Int. Symp. 13, 70–82. https://doi.org/10.1002/j.2334-5837.2003.tb02601.x

Valerdi, R., Wheaton, M., 2015. ANSI/EIA 632 As a Standardized WBS for COSYSMO, in: AIAA 5th ATIO And16th Lighter-Than-Air Sys Tech. and Balloon Systems Conferences. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2005-7373

Vargas, P.A., Paolo, E.A.D., Harvey, I., Husbands, P., 2014. The Horizons of Evolutionary Robotics. MIT Press.

Venturi, R., 1990. Complexity and Contradiction in Architecture. Museum of Modern Art.

Vermaas, P.E., Dorst, K., 2007. On the conceptual framework of John Gero's FBS-model and the prescriptive aims of design methodology. Des. Stud. 28, 133–157. https://doi.org/10.1016/j.destud.2006.11.001

Vijver, G., Salthe, S.N., Delpos, M., 2013a. Evolutionary Systems: Biological and Epistemological Perspectives on Selection and Self-Organization. Springer Science & Business Media.

Vijver, G., Salthe, S.N., Delpos, M., 2013b. Evolutionary Systems: Biological and Epistemological Perspectives on Selection and Self-Organization. Springer Science & Business Media.

Viollet-le-Duc, E.-E., Hearn, M.F., 1990. The architectural theory of Viollet-le-Duc: readings and commentary. MIT Press, Cambridge, Mass.

Vitruvius, 2012. The Ten Books on Architecture. Courier Corporation.

Vollini, E., Bohdal-Spiegelho, U., Brown, D., Burger, P., William, G., Gretczko, M., Hatfield, S., Maitra, S., Mazor, A., Moir, J., Sharma, D., Stephan, M., Tito, P., Van Durme, Y., 2020. The social enterprise at work: Paradox as a path forward.

Vose, M.D., 1999. The Simple Genetic Algorithm: Foundations and Theory. MIT Press.

Vossen, C., Kleppe, R., Hjørungnes, S., 2013. Ship Design and System Integration.

Vredenburg, K., Mao, J.-Y., Smith, P.W., Carey, T., 2002. A Survey of User-Centered Design Practice 8.

Wagner, A., 2013. Robustness and Evolvability in Living Systems. Princeton University Press.

Wake, W.K., 2000. Design Paradigms: A Sourcebook for Creative Visualization. John Wiley & Sons.

Waldeck, N.E., 2014. Advanced Manufacturing Technologies and Workforce Development. Routledge.

Walloth, C., Gurr, J.M., Schmidt, J.A., 2013. Understanding Complex Urban Systems: Multidisciplinary Approaches to Modeling. Springer.

Wang, K., Wang, Y., Strandhagen, J.O., Yu, T., 2016. Advanced Manufacturing and Automation V. WIT Press.

Wang, L., Nee, A.Y.C., 2009. Collaborative Design and Planning for Digital Manufacturing. Springer Science & Business Media.

Wang, T.C., 2002. Pencil sketching, 2nd ed. ed. Wiley, New York.

Wasson, C.S., 2005. System Analysis, Design, and Development: Concepts, Principles, and Practices. John Wiley & Sons.

Watanabe, M.S., 2002. Induction Design: A Method for Evolutionary Design. Springer Science & Business Media.

Watkins, S.M., Dunne, L., 2015. Functional Clothing Design: From Sportswear to Spacesuits. Bloomsbury Publishing USA.

Weilkiens, T., Lamm, J.G., Roth, S., Walker, M., 2015. Model-Based System Architecture. John Wiley & Sons.

Weiss, G., 1999. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press.

Werf, J. van der, Graser, H.-U., Frankham, R., Gondro, C., 2008. Adaptation and Fitness in Animal Populations: Evolutionary and Breeding Perspectives on Genetic Resource Management. Springer Science & Business Media.

Wessen, R.R., Borden, C., Ziemer, J., Kwok, J., 2013. Space mission concept development using concept maturity levels.

West, D.M., 2018. The Future of Work: Robots, AI, and Automation. Brookings Institution Press.

Whitesides, G., Mathias, J., Seto, C., 1991. Molecular self-assembly and nanochemistry: a chemical strategy for the synthesis of nanostructures. Science 254, 1312–1319. https://doi.org/10.1126/science.1962191

Wilensky, U., Rand, W., 2015. An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo. MIT Press.

Willemin, V., 2004. maisons mobiles, ALTERNATIVES edition. ed. Editions Alternatives, Paris.

Williams, K., March, L., Wassell, S.R., 2010. The Mathematical Works of Leon Battista Alberti. Springer Science & Business Media.

Williams, K., Ostwald, M.J., 2015. Architecture and Mathematics from Antiquity to the Future: Volume II: The 1500s to the Future. Birkhäuser.

Williams, L., 2010. Disrupt: Think the Unthinkable to Spark Transformation in Your Business. FT Press.

Wilson, C., 2013. Brainstorming and Beyond: A User-Centered Design Method. Newnes.

Windle, D.R., Abreo, L.R., 2003. Software Requirements Using the Unified Process: A Practical Approach. Prentice Hall Professional.

Winfree, E., 2006. Algorithmic Self-Assembly of DNA, in: 2006 International Conference on Microtechnologies in Medicine and Biology. Presented at the 2006 International Conference on Microtechnologies in Medicine and Biology, IEEE, Okinawa, pp. 4–4. https://doi.org/10.1109/MMB.2006.251471

Winner, H., 2013. Challenges of Automotive Systems Engineering for Industry and Academia, in: Maurer, M., Winner, H. (Eds.), Automotive Systems Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 3–15. https://doi.org/10.1007/978-3-642-36455-6_1

Witt, U., 1997. Self-organization and economics—what is new? Struct. Change Econ. Dyn. 8, 489–507. https://doi.org/10.1016/S0954-349X(97)00022-2

WMO, 2020. WMO statement on the state of the global climate in 2019.

Wolf, M., Mcquitty, S., 2011. Understanding the do-it-yourself consumer: DIY motivations and outcomes. AMS Rev. 1. https://doi.org/10.1007/s13162-011-0021-2

Wolfram, S., Illinois), S. (Wolfram R.I.W., 1999. The MATHEMATICA ® Book, Version 4. Cambridge University Press.

Wood, D., 2002. RVs & Campers: 1900-2000. Enthusiast Books.

Woodbury, R.F., 2010. Elements of Parametric Design. Routledge.

Wu, J., Qian, X., Wang, M.Y., 2019. Advances in generative design. Comput.-Aided Des. 116, 102733. https://doi.org/10.1016/j.cad.2019.102733

Wujec, T., 2017. The Future of Making. Melcher Media.

Wykis, 2007. Antibiotic resistance.

Wynn, D.C., Clarkson, P.J., 2018. Process models in design and development. Res. Eng. Des. 29, 161–202. https://doi.org/10.1007/s00163-017-0262-7

Wysocki, R.K., 2010. Effective Software Project Management. John Wiley & Sons.

Xin, D., Ma, L., Song, S., Parameswaran, A., 2018. How Developers Iterate on Machine Learning Workflows -- A Survey of the Applied Machine Learning Literature. ArXiv180310311 Cs Stat.

Yadin, A., 2016. Computer Systems Architecture. CRC Press.

Yan, X., Zhang, H., Wu, J., Li, W., Zhang, Y., 2011. Design the Evolutionary Algorithms Kernel in Adaptive Systems. Procedia Eng. 15, 2937–2942. https://doi.org/10.1016/j.proeng.2011.08.553

Yates, F.E., 2012. Self-Organizing Systems: The Emergence of Order. Springer Science & Business Media.

Yi, Y., Yan, Y., Liu, X., Ni, Z., Feng, J., Liu, J., 2020. Digital twin-based smart assembly process design and application framework for complex products and its case study. J. Manuf. Syst. S0278612520300571. https://doi.org/10.1016/j.jmsy.2020.04.013

Yin, S., Ang, Y., 2008. Applications of Complex Adaptive Systems. Idea Group Inc (IGI).

Yu, Y., Bandara, A., Honiden, S., Hu, Z., Tamai, T., Muller, H., Mylopoulos, J., Nuseibeh, B., 2019. Engineering Adaptive Software Systems: Communications of NII Shonan Meetings. Springer.

Zachman, J.A., 1987. A Framework for Information Systems Architecture.

Zannier, C., Erdogmus, H., Lindstrom, L., 2004. Extreme Programming and Agile Methods - XP/Agile Universe 2004: 4th Conference on Extreme Programming and Agile Methods, Calgary, Canada, August 15-18, 2004, Proceedings. Springer Science & Business Media.

Zeiger, M., 2009. Tiny Houses. Random House Incorporated.

Zeigler, D., 2014. Evolution: Components and Mechanisms. Academic Press.

Zhang, B.-T., Park, C.-H., 2008. Self-Assembling Hypernetworks for Cognitive Learning of Linguistic Memory 3, 5.

Zhang, J., Sanderson, A.C., 2009. Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization. Springer Science & Business Media.

Zhang, W., 2015. Selforganizology: The Science Of Self-organization. World Scientific.

Ziman, J., 2003. Technological Innovation as an Evolutionary Process. Cambridge University Press.

Zöllner, F., Nathan, J., Williams, K., 2003. Leonardo Da Vinci, 1452-1519: The Complete Paintings and Drawings. Taschen.

Zouaoua, D., Crubleau, P., Choulier, D., Richir, S., 2015. Application of Evolution Laws. Procedia Eng. 131, 922–932. https://doi.org/10.1016/j.proeng.2015.12.404

Zuech, A., 2002. Simplification Zachman Enterprise Framework.

Zyl, H., Du Preez, N., Schutte, C., 2007. Utilizing formal innovation models to support and guide industry innovation projects. South Afr. J. Ind. Eng. 18. https://doi.org/10.7166/18-2-127

# Published Papers

Publishing restrictions apply to a large percentage of the activity conducted at NASA which is not released.


1. R. Polit-Casillas; J. Elliott; MW Smith; A Austin; T Colaprete; T Fong; S Magnus; P Metzger; A Parness; HH Schmitt; B Sherwood; M Sims; MW Smith; G Voecks; K Zacny (2019). Architectural design considerations for a robotic power infrastructure on the Moon. International Astronautical Congress IAC2019, 21-25 October 2019, Washington DC, USA.
2. AS Howe; R Polit-Casillas; MW Smith; A Austin; T Colaprete; J Elliott; T Fong; S Magnus; P Metzger; A Parness; HH Schmitt; B Sherwood; M Sims; MW Smith; G Voecks; K Zacny (2019). Planetary Autonomous Construction System (P@X). International Astronautical Congress IAC2019, 21-25 October 2019, Washington DC, USA.
3. B Sherwood; A Austin; T Colaprete; J Elliott; T Fong; AS Howe; S Magnus; P Metzger; A Parness; R Polit-Casillas; HH Schmitt; M Sims; MW Smith; M Vaquero; G Voecks; K Zacny (2019). Robotic Lunar Surface Operations 2. Solar System Exploration Research Virtual Institute (SERVI), NASA Exploration Science Forum (NESF 2019), 23-25 July 2019, NASA Ames Research Center.
4. B Sherwood; A Austin; T Colaprete; J Elliott; T Fong; AS Howe; S Magnus; P Metzger; A Parness; R Polit-Casillas; HH Schmitt; M Sims; MW Smith; M Vaquero; G Voecks; K Zacny (2019). Robotic Lunar Surface Operations 2. Space Resources Roundtable (SRR), Planetary & Terrestrial Mining Science Symposium (PTMSS), 11-14 June 2019, Golden, Colorado, USA.
5. [6] J Elliott; A Austin; T Colaprete; T Fong; AS Howe; S Magnus; P Metzger; A Parness; R Polit-Casillas; HH Schmitt; B Sherwood; M Sims; MW Smith; G Voecks; K Zacny (2019). ISRU in Support of an Architecture for a Self-Sustained Lunar Base. International Astronautical Congress IAC2019, 21-25 October 2019, Washington DC, USA.
6. [7] MW Smith; A Austin; T Colaprete; J Elliott; T Fong; AS Howe; S Magnus; P Metzger; A Parness; R Polit-Casillas; HH Schmitt; B Sherwood; M Sims; MW Smith; G Voecks; K Zacny (2019). An Integrated Model for Performance Analysis of ISRU Lunar Base Concepts. International Astronautical Congress IAC2019, 21-25 October 2019, Washington DC, USA.
7. Polit-Casillas, R., Tuman, D., Ravich, J., Martinez, R., Karapetian, A., Soloway, L., McEnerney, B., Smith, C., “4D Architectures”, Publication Pending, 2019
8. Polit-Casillas, R., “A stitch in time and space”, pp 52-53 and cover, 2016. In Hadaegh, F., Zmuidzinas, J., ‘JPL Technology Highlights, 2015-2016’, , Pasadena, 2016, available online: https://scienceandtechnology.jpl.nasa.gov/sites/default/files/documents/JPL_2015-2016_Technology_Highlights.pdf
9. Polit-Casillas, R., “A stitch in time and space”, pp 14-15, 2017. In Hadaegh, F., Zmuidzinas, J., ‘JPL Technology Highlights, 2017-2018’, PP 14-15, Pasadena, 2017, available online: https://scienceandtechnology.jpl.nasa.gov/sites/default/files/documents/JPL_2016-2017_Technology_Highlights.pdf
10. Paper, A. A. Shapiro, J. P. Borgonia, Q. N. Chen, R. P. Dillon, B. McEnerney, R. Polit-Casillas, L. Soloway, “Additive Manufacturing for Aerospace Flight Applications”, AIAA, Journey of Spacecraft and Rockets, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, DOI: 10.2514/1.A33544, 2016
11. Howe, A.S., Wilcox B., McQuin, C., Mittman, D., Townsend J., Polit-Casillas, R., Litwin, T., “Modular Additive Construction Using Native Materials”, Jet Propulsion Laboratory, ASCE’s Aerospace Division, 14th Earth and Space Conference, St. Louis, October 2014
12. Article and cover, Polit-Casillas, R., Borgonia, J.P., “A Stich in time and space” (3-D printed multifunctional space fabrics”, Jet Propulsion Laboratory Technology Highlights 2015-2016, National Aeronautics and Space Administration, Jet Propulsion Lab, CL#16-2090, Page 62-63 and cover, 2016

13. Paper, Sherwood, B., Lunine, J., Elliott, J., Imken, T., Cable, M., Kempf, S., Southworth, B., Tucker, S., Waite, J.H., Frick, A., McCoy, K., Karapetian, A., Polit-Casillas, R., Oh, D., Hand, K.P., "Sylph: Life Detection Probe for a Europa Plume", 67th international astronautical congress (IAC), Mexico, September 2016, International Astronautical Federation (IAF), 2016
14. Article, Polit-Casillas, R., Shapiro A., Borgonia, J.P., Mcenerney, B., "Multifunction 3D-Printed Fabrics", NASA Tech Briefs, NPO-49925, www.techbriefs.com/tsp (materials and coating), September 2015
15. Interview and article, NASA Spinoff, "Mars Rover Work Spawns PDF Collaboration Software", NASA Technology Transfer Program, [Cited February 2016], Available at https://spinoff.nasa.gov/Spinoff2016/it_3.html , 2016
16. Paper, Zafirian, P., Imken, T., Matousek, S.E., Moeller C.R., Bennett M.W., Norton, C.D., Rosenberg L., Alibay, F., Spangelo, S., Banazadeh, P., Polit-Casillas, R., Kawate, J., "Team Xc: JPL's Collaborative Design Team for Exploring CubeSat, NanoSat, and SmallSat-based Mission Concepts", IEEE Big Sky Conference, Published, 2015
17. Paper, Polit-Casillas, R., Howe, A.S., "Virtual Construction of Space Habitats and hardware: Connecting Building Information Models (BIM) and SysML", AIAA Space 2013 Conference
18. Contribution, Price, H., Manning R., Evgeniy Sklyanskiy, Robert Braun4 "A High-Heritage Blunt-Body Entry, Descent, and Landing Concept for Human Mars Exploration", AIAA
19. Paper, Polit-Casillas, R., "Self-Deployable Lunar Habitat, Part-1: Overall Architectural Design and Deployment", PI, AIAA ICES Conference, San Diego 2012, AIAA 2012-3556, 2012
20. Paper, Polit-Casillas, R., "Logistics-2-Shielding Mapper: Automated Space Construction System Using CTBs", AIAA 2012-3556, ICES 2012 (patent pending)
21. Paper, Shull, S., Howe, A.S., Polit-Casillas, R., "NASA AES: Concepts for Logistics to Living", AIAA Space Conference 2012, Published AIAA, 2012
22. Publication, Polit-Casillas, R "Building Technological Habitats International Exhibition", AIAA Aerospace America 2012 volume, Space Architecture, pp 61, December 2012.
23. Paper, Polit-Casillas, R., "Applied Space Architecture (AIAA 2010-6108)" 40thth ICES, AIAA, 2010, Published AIAA, 2010

# Outreach and Media Publications

24. Documentary, NASA, 'Hispan@s de la NASA (Hispanics of NASA)', 2019. https://www.youtube.com/watch?v=o1b0H5AfKZw
25. Google Sci-Foo 2018 attendant and presenter, "4D Multidisciplinary Architectures, doing more with less", Palo Alto, CA, 2018
26. Amazon MARS 2018 attendant and expo presenter, "4D Multidisciplinary Architectures for Space, doing more with less through mass repurposing", Palm Spring, CA, 2018
27. TEDx Manhattan Beach Speaker, "Impossible Architectures: 4D Spacecraft", https://www.youtube.com/watch?v=bPaQaoGHerI , CA, 2017
28. TIME Magazine, Zorthian, J., "NASA's New 'Space Fabric'", TIME, http://time.com/4775439/nasas-new-space-fabric/, online, May 2017
29. WIRED Magazine, Stinson, E., 'NASA's Wild Fabric Is Basically Chain Mail From the Future', Wired, https://www.wired.com/story/nasa-fabric-chain-mail-from-the-future/ , online, June 2017
30. NASA-JPL,' 'Space Fabric' Links Fashion and Engineering', https://www.jpl.nasa.gov/news/news.php?release=2017-110 , online, April, 2017

31. CNN Video, 'Space Fabric', CNN, http://time.com/4775439/nasas-new-space-fabric/ , online, 2017
32. Keynote Speaker, Polit-Casillas, R "Designing for additive manufacturing in the space industry", Solidthinking Converge Conference, Los Angeles, September 2016
33. Lecture, "The Hidden Dimensions of Architecture and Engineering Design", Frank O. Ghery, Robert Manning, Raul Polit-Casillas, JPL, Pasadena July 2016
34. Keynote Speaker, Polit-Casillas, R., Howe, S., "Virtual Space Construction for Space Habitats and Systems – VSCH-BIM", Consult Australia Technology Symposium, Sydney, Online Attendance, 2013
35. Lecture, Polit-Casillas, R, "Applied Space Architecture: Advance constructive design for habitats in Space and on Earth" seminar at Caltech - Solar Decathlon Project - Caltech/SciArc - Guest lecturer (Pasadena, CA, 2012)
36. Lecture and talk, "Applied Space Architecture", JPL NASA, Caltech, Pasadena, CA, 2012
37. Lecture and student project review, "Implementing Space Habitats" Technical Seminar, Cal Poly Pomona - Dept. of Environmental Design, Space Architecture Studio (M. Fox), Pomona, CA, 2012
38. Seminar, Space Architecture and Sustainability International Seminar – UCV, Director (2010-11)
39. Symposium, Aliter: International Symposiums in Space Architecture: Approaching Space Architecture Otherwise - Seminar, workshop and public exhibition, Founder, director (Spain, 2010)
40. Lecture, Polit-Casillas, R., "Space Architecture: from Dolmens to Space Stations" Conference, Ateneo Mercantil, Valencia, 2010

# Legal Notes, Copyright, and Restriction Disclosure

## Work at JPL

This work was done as a private venture and not in the author's capacity as an employee of the Jet Propulsion Laboratory, California Institute of Technology. This is a fundamental research activity presenting only theoretical results.

The topic of this dissertation has a purely fundamental research approach, and it is not related to any real space hardware, mission, or practice. Tools, processes, and methods here are generic and based on public and commercially available resources or publications. The office of export control at JPL was informed and updated of the scope and purpose of this activity since before the activity started. This research is not connected to any specific sector or industrial application since it only introduces personal theories, conclusions, and general concepts developed solely by the author as part of this fundamental and theoretical research.

## Copyright

*Inscription au-delà de la 3ème année de doctorat*

*Soutenance du Doctorat prévue avant le 31 décembre de l'année universitaire*

Direction de la recherche et de la valorisation
Collège doctoral
Université de Strasbourg

# Attestation sur l'honneur.

Inscription au-delà de la 3ème année de doctorat. Soutenance du doctorat prévue **avant** le 31 décembre de l'année universitaire

**Demande déposée par :**

☐ Madame ☒ Monsieur

Nom : Polit Casillas ..... Nom d'usage : .....

Prénom Raul ..... Né(e) le *(jj/mm/aaaa)* : 09/05/1979 .....

Ville de naissance Jaca ..... Pays de Naissance : Espagne .....

- → **Titre de la Thèse :** CONCEPTION D'ARCHITECTURES EVOLUTIVES DE SYSTEMES .....
- → **Directeur de thèse :** Emmanuel Caillaud .....
- → **Unité de recherche :** iCUBE .....
- → **Ecole doctorale :** Mathématiques, Sciences de l'Information et de l'Ingénieur (MSII, 269) .....
- → **Thèse en cotutelle internationale :** ☐ **Oui** ☒ **Non.**

**Date prévisionnelle de la soutenance :** 17 march 2021 .....

**Je soussigné(e)** Raul Polit Casillas **, m'engage à soutenir ma thèse avant le 31/12 de l'année universitaire en cours.**

**Si cet engagement n'est pas respecté, je m'engage à payer les droits CVEC et mes frais d'inscription au titre de l'année** 2021 .....

Fait à : Los Angeles ..... le : January 18th, 2021 .....

Signature du/de la doctorant·e:

« L'état d'avancement des travaux, de la rédaction de la thèse sont suffisamment avancés pour que la soutenance du doctorat puisse se tenir avant le 31 décembre de l'année universitaire. »

Signature du Directeur de thèse :

Signature du Directeur de l'unité de recherche :

Pièces à Joindre

- → **Proposition du jury de soutenance**
- → **Résumé de la thèse**

Direction de la **recherche**
et de la **valorisation**
Collège **doctoral**
Université de Strasbourg

# Déclaration sur l'honneur
# *Declaration of Honour*

J'affirme être informé que le plagiat est une faute grave susceptible de mener à des sanctions administratives et disciplinaires pouvant aller jusqu'au renvoi de l'Université de Strasbourg et passible de poursuites devant les tribunaux de la République Française.

Je suis conscient(e) que l'absence de citation claire et transparente d'une source empruntée à un tiers (texte, idée, raisonnement ou autre création) est constitutive de plagiat.

Au vu de ce qui précède, **j'atteste sur l'honneur que le travail décrit dans mon manuscrit de thèse est un travail original et que je n'ai pas eu recours au plagiat ou à toute autre forme de fraude.**

*I affirm that I am aware that plagiarism is a serious misconduct that may lead to administrative and disciplinary sanctions up to dismissal from the University of Strasbourg and liable to prosecution in the courts of the French Republic.*

*I am aware that the absence of a clear and transparent citation of a source borrowed from a third party (text, idea, reasoning or other creation) is constitutive of plagiarism.*

*In view of the foregoing,* ***I hereby certify that the work described in my thesis manuscript is original work and that I have not resorted to plagiarism or any other form of fraud.***

**Nom : Prénom :** Raul Polit Casillas

**Ecole doctorale :** Collège doctoral - Université de Strasbourg

**Laboratoire :** iCUBE

**Date :** April 13th, 2021

**Signature :** 

Raul Polit-Casillas

# Conception d'architectures évolutives de systèmes : Pour des systèmes complexes adaptables, régénératifs et intelligents à l'ère du changement climatique.

## *Evolutive System Architecture Design: Enabling Adaptable, Regenerative, and Smart-Driven Complex Systems in the Era of Climate Scarcity.*


## Résumé

Cette thèse explore les nouveaux besoins de conception d'ingénierie des systèmes pour les architectures de système évolutives (eSAR), qui sont un sous-ensemble d'une nouvelle génération de systèmes matériels complexes, dans un contexte défini par des facteurs de stress de conception globaux tels que la rareté des ressources et la complexité. Ces systèmes évolutifs sont hautement adaptables, visant la régénération des ressources et présentant une base de référence très intelligente. Sur la base d'une vaste revue de la littérature mettant en évidence les principales lacunes dans les techniques d'ingénierie de conception et d'ingénierie système de pointe, une méthodologie de développement évolutif à cycle complet (eSARD) est présentée, inspirée des mécanismes d'évolution naturelle tout en abordant le patrimoine et de meilleures performances du système. La méthode holistique eSARD aborde la conception, la mise en œuvre, les opérations système et l'optimisation globale du système d'un eSAR.


Mots-clés : Ingénierie système, conception, soutenabilité


## Résumé en anglais

This thesis explores new systems engineering design needs for evolutive system architectures (eSAR), which are a subset of a new generation of complex hardware-based systems, within a context defined by global design stressors such as resource scarcity, and complexity. These evolutive system are highly adaptable, aiming towards resource regeneration, and presenting a highly intelligent baseline. Based upon an extensive literature review highlighting key gaps on state-of-the-art design engineering and system engineering techniques, a full cycle evolutive development methodology (eSARD) is presented inspired by natural evolution mechanisms while addressing heritage, and better system performances. The holistic eSARD method tackles design, implementation, system operations, and overall system optimization of an eSAR.


Keywords: systems engineering, design, sustainability