Agility as a tool for the Management of Systems Engineering Projects

Diego Armando Diaz Vargas

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Institut National des Sciences Appliquées de Toulouse

Présentée et soutenue par

Diego Armando DIAZ VARGAS

Le 15 février 2019

L'agilité comme outil pour la gestion de projets d'ingénierie des systèmes.

École doctorale : **SYSTEMES**

Spécialité : **Informatique**

Unité de recherche :
LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes

Thèse dirigée par
Claude BARON et Philippe ESTEBAN

Jury

M. Emmanuel CAILLAUD, Rapporteur
M. Christophe MERLO, Rapporteur
M. Marc ZOLGHADRI, Examinateur
M. Rob VINGERHOEDS, Examinateur
Mme Claude BARON, Directeur de thèse
M. Philippe ESTEBAN, Co-directeur de thèse
Mme. Citlalih Y.A. GUTIERREZ ESTRADA, Invité

Acknowledgements

I would like to start by sharing the meaning of doing a PhD thesis. As a foreign student, a PhD is a challenge and a wonderful life learning, I dare to describe it as a hike to the mountain. At the beginning of the hike, you only see to the top of the mountain, you know there is a way to go, but you don't know the details and surprises you will find on the way, as you progress, the mountain challenges you, and tests your physical and mental abilities, sometimes the challenges to face, make you doubt to reach the top, but when you pause, to observe what is around you, you realize that there is always a possibility to reach the top of the mountain. Once you get there, you realize that it has been a unique learning, you improve some of your skills and learning new, however, one of the most important lessons, when you reach to the top of the mountain, is the humility you acquire with yourself, with your environment and with life.

I want to thank my country, thanks to the financing granted through CONACyT, I have been able to experience and learn many things from different countries of mine. I also want to thank all the people in France (INSA, LAAS-CNRS and EDSYS) who supported me to comply with the procedures required during my PhD.

I would like to tank my superviors, Porfessor BARON Claude, and Professor ESTEBAN Philippe, for allowing me to come to LAAS-CNRS laboratory, and the time given to my thesis work. Thanks to Professor Citlalih GUTIERREZ ESTRADA, and Professor Rob VINGERHOEDS for the advice given to my work. I would like to thank all my colleagues of the ISI team, Min, Karla, Guillaume, Xin, Cathy and Vatsal, in LAAS-CNRS, for the moments we shared together, I feel fortunate to have shared unforgettable moments with all of you!

Thanks to my Mexican friends in France, it has been a pleasure to see how you have become an example of life for me. I would also like to thank Chantal, and Marshall, for make me laugh, when the days were difficult, you are the best roomates. Thanks to Gertrude and Serge for allowing me to enter their home, and remember my childhood. Thanks to Isabel for being with me all this time, for being the light when my life was darker, for listening and understandig me, this work also belogs to you. Thanks to Baptiste for being part of my life, for showing me to be brave, for the shared adventures, and for undestanding me at this stage of my life. All of you are my family in France, and I will always love you.

Last but not least, I want to thank my family. My mother and my father have always supported me in achieving my professional goals, they are the best example of love, happiness and patience in my life. My beloved sisters, nephew, nieces, and brother-in law, I love you so much and this work is for you!

Abstract

Project performance is considered as an important factor to ensure the success of a project. Companies are interested in the use of efficient practices through efficient methods and tools to design and deliver innovative products and services and decrease the time to market. Project duration, costs, and performance are aspects that normally face changes during the project development. These changes should be treated by using adapted and optimized processes in order to better control, coordinate, manage, and improve projects. Agile methods seem to be efficient for the management of successful projects, however they are mainly use in companies where the business domain is software. Agile methods recently received a growing interest from industry and now are well accepted and deployed in software engineering. This thesis thus tackles the point of transferring the agile methods from software to systems engineering, and issues that are induced.

The report first introduces the notion of agility and the birth of the agile movement as well as the principles and values of agile software development. It also presents the main agile methods, as well as other philosophies that share a number of similarities with Agile. Project attributes can be defined, from the literature, to help contextualizing agile projects; we describe and use these project attributes to compare different agile methods and identify the differences between them. Agile and Lean are compared to determine why Lean is used in software development, and how it differs from other agile methods. Finally, we identify several issues to transfer agile methods in the context of systems engineering.

We then focus on the understanding of agility in systems engineering. Two meanings of "agile" are found in literature. Considering that agility is focused in the rapid change of convincing, designing, and implementing processes of products and systems in an easy way, we explore the question of introducing agility in systems engineering. A first analysis is led to identify any notion of agility in systems engineering standards. The results of this analysis help us to highlight the issues and challenges of transferring agility into systems engineering. Focusing on the issues, we then present a four steps research methodology. The first step aims to define a contextual model for systems

engineering development. The contextual model contains the organizational factors and the project attributes for engineering projects. This contextualization leads us to identify if and which agile method could be used for the management of engineering projects (step two). The step three justifies our selection of the Scrum Framework, between agile methods, for the management of engineering projects. Scrum Practices are defined and evaluated in the project attributes for engineering projects. However, several difficulties are identified and listed while using the Scrum Practices in engineering projects. Finally, the step four proposes some solutions to solve a set of difficulties.

This work finally proposes the use of Scrum Practices in two engineering projects. An educational project is analyzed first. This project aims to develop a connected robot. By starting from the contextual model for systems engineering development (cf. section III.4.2.d), we characterize the project to identify what type of project is, then we propose the use of the graphical view of the Scrum Framework to plan the development of the robot. Following the same schema, a second industrial project is analyzed. The second project aims to develop an automotive application for engine management.

Résumé

La performance d'un projet est considérée comme un facteur important pour en assurer le succès. Les entreprises s'intéressent à l'utilisation de pratiques efficaces au moyen de méthodes et d'outils efficaces pour concevoir et offrir des produits et des services novateurs et réduire le temps de mise sur le marché. La durée, les coûts et le rendement du projet sont des aspects qui font normalement face à des changements au cours de l'élaboration du projet. Ces changements doivent être traités en utilisant des processus adaptés et optimisés afin de mieux contrôler, coordonner, gérer et améliorer les projets. Les méthodes agiles semblent être efficaces pour la gestion de projets réussis, mais elles sont surtout utilisées dans les entreprises où le domaine d'activité est le logiciel. Les méthodes agiles ont récemment suscité un intérêt croissant de la part de l'industrie et sont maintenant bien acceptées et déployées en génie logiciel. Cette thèse aborde donc l'intérêt de transférer les méthodes agiles du logiciel à l'ingénierie des systèmes, et les enjeux qui y sont induits.

Le travail de thèse introduit d'abord la notion d'agilité et la naissance du mouvement agile ainsi que les principes et les valeurs du développement logiciel agile. Il présente également les principales méthodes agiles, ainsi que d'autres philosophies qui partagent un certain nombre de similitudes avec l'agile. Il y a des attributs de projet, dans la littérature, qui aident à caractériser les projets agiles, ces attributs de projet sont décrits et utilisés pour comparer différentes méthodes agiles pour identifier les différences entre elles. Agile et Lean sont comparés pour déterminer pourquoi Lean est utilisé dans le développement logiciel, et en quoi il diffère des autres méthodes agiles. Enfin, nous identifions plusieurs problèmes de transfert de méthodes agiles dans le contexte de l'ingénierie des systèmes.

Nous nous concentrons ensuite sur la compréhension de l'agilité en ingénierie des systèmes. Deux sens d’agile se retrouvent dans la littérature. Considérant que l'agilité est centrée sur le changement rapide de processus de produits et de systèmes convaincants, conçus et mis en œuvre de manière simple, nous explorons la question de l'introduction de l'agilité en ingénierie des systèmes. Une première analyse est menée pour identifier toute notion d'agilité dans les normes d'ingénierie des systèmes. Les résultats de cette analyse nous aident à mettre en évidence les enjeux et les défis

du transfert de l'agilité dans l'ingénierie des systèmes. En nous concentrant sur les enjeux, nous présentons ensuite une méthodologie de recherche en quatre étapes. La première étape vise à définir un modèle contextuel pour le développement de l'ingénierie des systèmes. Le modèle contextuel contient les facteurs organisationnels et les attributs des projets d'ingénierie. Ensuite, la sélection d'une méthode agile qui pourrait être utilisée pour la gestion de projets d'ingénierie est proposée à l'étape deux. La troisième étape introduit l'utilisation de Scrum. Les pratiques Scrum sont définies et évaluées dans les attributs de projet pour les projets d'ingénierie. Les difficultés sont identifiées et répertoriées lors de l'utilisation des pratiques Scrum dans les projets d'ingénierie. Enfin, la quatrième étape propose des alternatives pour résoudre un ensemble de difficultés.

Ce travail propose enfin l'utilisation des pratiques Scrum dans deux projets d'ingénierie. Un projet éducatif est d'abord analysé. Ce projet vise à développer un robot connecté. En partant du modèle contextuel pour le développement de l'ingénierie des systèmes, nous caractérisons le projet pour identifier le type de projet, puis nous proposons l'utilisation de la vue graphique de Scrum pour planifier le développement du robot. Suivant le même schéma, un deuxième projet industriel est analysé. Le second projet vise à développer une application automobile pour la gestion du moteur.

Content

List of Figures

List of Tables.

I. Introduction

This section introduces the context of the research work. All the research work that has been done during this thesis was carried out with the Systems Engineering and Integration team (ISI) in the Laboratory for Analysis and Architecture of Systems (LAAS-CNRS), in Toulouse, France. ISI team covers the design of complex systems in the fields of aeronautics, automotive, railway industry, and microsystems. The ISI team contributions aim to propose solutions to effectively implement and manage engineering processes, and to improve the process and methods for designing complex systems. The work presented in this report involves three bodies of knowledge: Systems Engineering (SE), Project Management (PM), and Agile Management. This thesis was funded by a Mexican scholarship from the National Council for Science and Technology (CONACyT).

I.1. Context of the research and objectives

Companies continuously need to improve their practices and performance, thus using efficient methods and tools to design and deliver innovative products and services and to decrease the time to market. In project and program environments, complexity is a characteristic that is difficult to manage due to the human behavior, system behavior, and ambiguity (Rebentisch, 2017). In today's highly competitive economic environment, companies are concerned by improving project performance and applying efficient practices to manage their engineering projects; their goal is to simplify and speed up the implementation of projects in order to better control, coordinate and manage them. Nowadays, most of companies use agile methods, such as Scrum Framework, Extreme Programming, Crystal, Dynamic System Development, etc., that have proven to be very efficient to lead successful software engineering projects, but we noted that they are scarcely used in systems engineering projects. Indeed, agile methods are nowadays widely spread in software industry, but fields like systems engineering are still contemplating these methods in order to see if they can lead projects in complex systems development this way. Even if introducing agility in systems engineering makes sense, companies have not deployed such methods yet.

Why? Is it due to a lack of appropriated methods? Are the most popular agile methods used in software engineering not well adapted to systems engineering applications? Is it due to a question of compliance to international standards? These standards for instance indeed recommend having a full list of precise requirements before beginning the design, while agile methods recommend being in constant interaction with the customer to iteratively define the requirements (Meyer, 2014).

This thesis thus tackles the research objective of using agility to manage and improve the performance of systems engineering projects. That includes several questions: Can the transfer of agile methods from software engineering to systems engineering be immediate? If not, what are the difficulties? Does the agility refer to the product, the processes or the project? Do systems engineering standards already implicitly consider a kind of agility or could they be compliant with agility?

The scientific approach, used in this work, starts from an analysis of the use of agile methods in software engineering projects. The identification of the principles and best agile practices is our first focus to further integrate agility in engineering projects. We then study the most used agile methods and compare them to identify the key factors that could be useful in systems engineering projects. Before going further, we analyze the current international systems engineering standards to determine if ever a kind of agility could be found in some of them. We also identify the potential problem(s) in integrating an agile method in engineering projects in order to try and consider options to solve some of them. We finally elaborate a methodological guide that allows the integration of agility as a tool managing systems engineering projects and validate the proposal with a case study.

I.2. Organization of the report

The thesis report is organized in five chapters. Figure 1 illustrates the structure of each chapter. Chapter I introduces the context and objectives of the research work. This chapter also states some questions to highlight the issues to be considered during the research.

Chapter II presents a literature review on agility. It starts from the characterization of agility, then describes the evolution of agile practices. It also introduces and describes the main currently used agile methods; this analysis leads us to identify the domains and levels of deployment of agile methods in industry and to define the essence of agility. How agility was extended to operations, Lean thinking and the Scaled Agile Framework (SAFe) overview is also part of this chapter. Then an analysis of the most known agile methods is given to formulate a new possible agile approach for the management of engineering projects Chapter II concludes pointing out the interest of agility and identifying the difficulties to overcome and to transfer agile practices to systems engineering management.

Chapter III introduces the presence of agility in systems engineering projects. It starts by discussing the differences between agile-systems engineering and agile systems-engineering. Then, an exploration, to introduce agility in systems engineering, is described. It also analyses the ISO/IEC 15288 standard in order to try and found any trace of agility, the result of this analysis shows that agility is present in systems engineering. Finally, a research methodology is proposed. The research methodology led us to define a contextual model for systems engineering development, and to deploy Scrum Framework in engineering projects. Some alternatives are listed in order to solve the difficulties while using the Scrum Practices in engineering projects.

The last stage of this research is presented in Chapter IV; it includes the validation of the research methodology through a case study. Two projects are analyzed to identify how they can be managed using the Scrum Practices.

Chapter V concludes on this work and give some perspectives for future work.

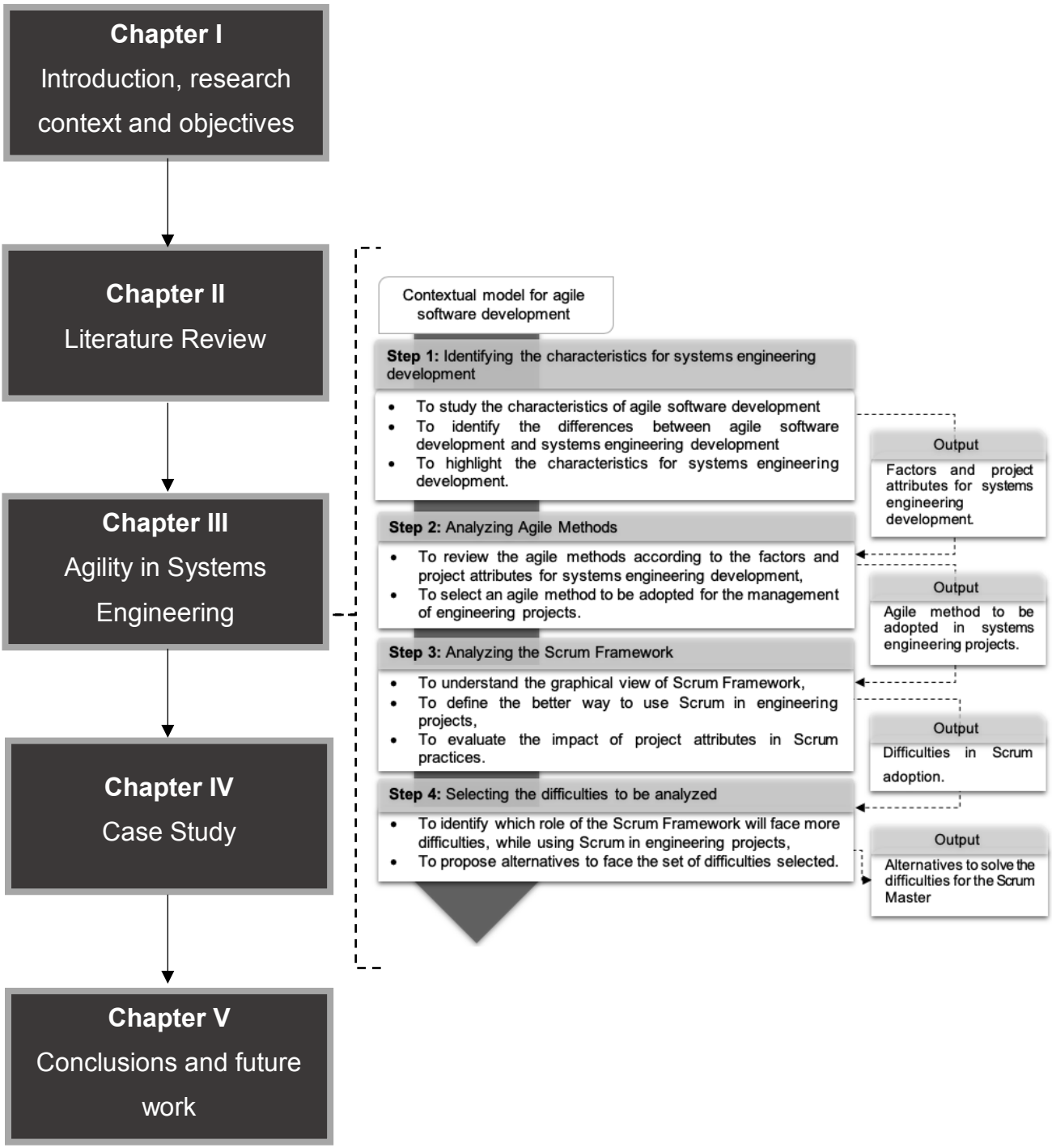

Figure 1. Organization of the thesis report

II. Literature Review

This chapter introduces the literature related to the subject of this thesis. Section II.1 defines agility, indicates when and how the agile movement was born and presents the agile methods that can be found in literature. Section II.2 introduces Lean philosophy. Section II.3 presents the Scaled Agile Framework (SAFe). Section II.4 compares the agile methods with Lean philosophy and SAFe. Section II.5 presents the conclusions of this chapter.

II.1. Agile Overview

This section introduces the "agile" and "agility" terminology in the software development context, then describes the emergence of the agile movement and where the agile ideas date back of.

II.1.1 Agile Term.

Agility is defined by the Cambridge dictionary as the "*ability to move quickly and easily"* (Cambridge English Dictionary, 2017). The term 'Agile' appeared with a movement which was born in the early 1990s. In 1992, the Agile Manufacturing Enterprise Forum was founded by Texas Instruments and General Motors to identify the nature of agile solutions by organizing collaborative workshop groups (Hoda & Murugesan, 2016). In the definition of the Cambridge dictionary agile is an adjective and the noun is agility, in all cases, authors take the adjective to imply that their methods use agility meaning.

Several definitions can be found in literature, such as the following, to only name a few:

- Augustine et al. characterize the *agile approach* as an "overall humanistic problem solving approach", which assumes that all members are skilled and valuable stakeholders relying on the collective ability of autonomous

teams as the basic problem-solving mechanism and minimizing up-front planning (Augustine, Payne, Sencindiver, & Woodcock, 2005).

- Bertrand Meyer states that the word *'Agile'* denotes a compendium of ideas which a number of full-fledged methods are applied in various subsets and combinations (Meyer, 2014).

- The Massachusetts Institute of Technology (MIT) defines *Agility* as a Team's Competence that will contribute to performance regardless of the product development context or business sector (Conforto, Rebentisch, & Amaral, 2014).

- According to the SE Handbook of INCOSE, *Agility* is a capability exhibited by systems processes that enables them to sustain effective operation under conditions of unpredictability, uncertainty, and change (Walden et al., 2015).

- The Agile Practice Guide considers "agile" as a term used to describe a mindset of values and principles as set forth in the Agile Manifesto (PMI, 2017a).

- The PMBoK (Project Management Body of Knowledge) introduces the term "agile" as a characteristic of the Adaptive Life Cycles (ALC). ALC is a project life cycle that is iterative and incremental, that means that the deliverables of a project are developed over multiples iterations where detailed scope is defined and approved for each iteration when it begins (PMI, 2017b).

- The Agile Alliance introduces "agile" as the ability to create and respond to change in order to succeed in an uncertain and turbulent environment (Agile Alliance, 2018)

Agile methods promote the engagement of the team members as local domain experts in integration management. The notion of agility is now widely spread in software

engineering. However, it is not the case in systems engineering. Some references to agility in systems engineering can be found in literature, but a particular attention must be given to the terminology. According to the MIT executive report (Conforto et al., 2014) agility or being agile is not only an adjective, method or practice, agility should be considered as a team's competence in the project environment, to achieve this competence properly organizations and decision-makers have to consider several aspects and elements, some of those elements are listed in Figure 2.

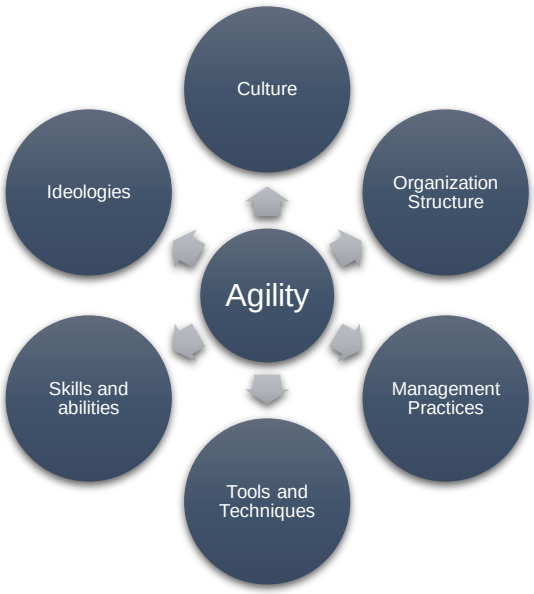

Figure 2. Aspects and Elements of Agility (Conforto et al., 2014).

The way organizations see agility should move from a mindset focused on the collection of tools and practices to that of an indicator of the project team's competence (Conforto et al., 2014). The definition of the MIT executive report seems to be one of the most complete; the elements listed in Figure 2 integrate ideas from the definitions of other authors, and they can be applied in different combinations under uncertainty and change conditions. This definition will be use as the base concept of agility in this thesis work.

II.1.2 Agile Movement

Different events took place until 1984, where the criticism of the "waterfall" sequential approach started, and formulations of alternative incremental approaches were become more pointed, the specific reason was that complete and stable specifications were not available (Agile Alliance, 2018). Before the existence of the Manifesto for

Agile Software Development (in the following sections we will use the term “Agile Manifesto” to refer to), different events led to the use of agile practices. The Agile Alliance trace the history and evolution of Agile; this organization states that the agile roots began in 1986, with the Conway’s law. This law emphasizes that any organization that designs a system will inevitably produce a design whose structure is a copy of the organization’s communication.

In 1990s, some techniques were developed to formalize the Agile Manifesto, and several methodologies began to gain increasing public attention, each having a different combination of old and new ideas. That brings the agile movement, the agile ideas date back to the development of Extreme Programming (XP) by Kent Beck (Poppendieck & Poppendieck, 2010). The terms ‘agile’ and ‘agility’ can be traced back to the manufacturing industry in 1991 when lean development emerged in manufacturing with the aim of eliminating waste, amplifying learning, delivering as fast as possible and empowering teams. This movement allows, in 1992, the Agile Manufacturing Enterprise Forum; it was founded by Texas Instruments and General Motors, to identify the nature of agile solutions by organizing collaborative workshop groups (Hoda & Murugesan, 2016). In 2000, a number of articles described a variety of “light” or “lightweight process” and “light methodologies”; the “agile” term had not been used in a formal way at that time (Beck et al., 2001). Agile ideas reached fame with the appearance of the Agile Manifesto in 2001, the specific reason behind its definition was that software experts want to figure out why so many software projects were failing, sharing their experiences they gave birth to the Agile Manifesto (Beck et al., 2001).

Concepts, as continuous delivery (Beck et al., 2001), were used in process and methodologies before its definition. Agile methodologies emphasized close collaboration between the development team and business stakeholders; frequent delivery of business value, tight, self-organizing teams; and smart ways to craft, confirm, and deliver code. We can find in literature that the principal methods that inspired the Agile Manifesto were (Agile Alliance, 2018):

- *Crystal,* it focusses in people, not process or artifacts. The use of osmotic communication is a strong characteristic of this methodology (Meyer, 2014),

- *Refactoring,* an aid in designing application frameworks and evolving object-oriented systems. This methodology responds to change over following a plan (Agile Alliance, 2018).
- *Dynamic System Development Methods (DSDM),* it focusses on get together the best part of control, quality (traditional approach) and good communication, business involvement transparency (Agile Business Consortium, 2015),
- *Scrum,* a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value (Schwaber & Sutherland, 2017),and
- *Extreme Programming (XP),* it is a disciplined approach to delivering high quality software, quickly and continuously. The notion of Increment, then simplify, is a principal characteristic of this method (Meyer, 2014).

Section II.2.4 will describe in more detail the agile methodologies. Each methodology presents techniques where the interaction between individuals is an important characteristic over the development cycle of the project or product. Another common feature in agile methodologies is the collaboration of client throughout the cycle of project or product development. Change during the development cycle of a project (or product) are very common, how actors of the project or product react to change should follow a plan in agile environments. All these characteristics allowed to define a base of values and principles for software development.

The following section introduces the set of values and principles that were defined as part of the Agile Manifesto.

II.1.3 Agile Manifesto

This section introduces the Agile Manifesto and what is the global idea of using agility in software development. The Agile Manifesto is the source of the states, values and principles of the agile movement. It formalized techniques which had been developed in the 1990s. It relies on 4 values (Beck et al., 2001):

1. ***Individuals and interactions*** *over processes and tools.*
2. ***Working software*** *over comprehensive documentation.*
3. ***Customer collaboration*** *over contract negotiation.*
4. ***Responding to change*** *over following a plan.*

These values focused on a direct and oral communication with the customer; the values state that we should value individuals and interactions, working software, customer collaboration and responding to change more than process and tools, comprehensive documentation, contract negotiation and following a plan, that does not mean that processes, contract, etc., are not important for the project development. In addition to the values of the Manifesto, there are 12 principles that support the values (Beck et al., 2001):

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Principles are very general, and they give the ability to make a good decision in a particular situation during software development. In synthesis, the Agile Manifesto focuses on teamwork closely involving the customer; agility relies in an iterative and incremental development, resulting in frequent and fast releases (Uskov, Krishnaiah, Kondamudi, & Singh, 2016).

II.1.4 Analysis of agile methods

This section introduces the most widely used agile methods. They are currently practiced in the industry and some of them are derived from theories that existed before the definition of the Agile Manifesto (Stark & ClydeBank Business, 2016). Figure 3 shows an example of guidance detail and life coverage of the agile methods; the black squares are focused in team method and the white squares in a scaled approach. It can be noticed, that some agile methods are more full-featured than others (guidance detail axe); on the other hand some agile methods are formalized for common use (life coverage axe), that means that they are designed for a specific use by a single organization, within a single context or in a variety of contexts (PMI, 2017a).

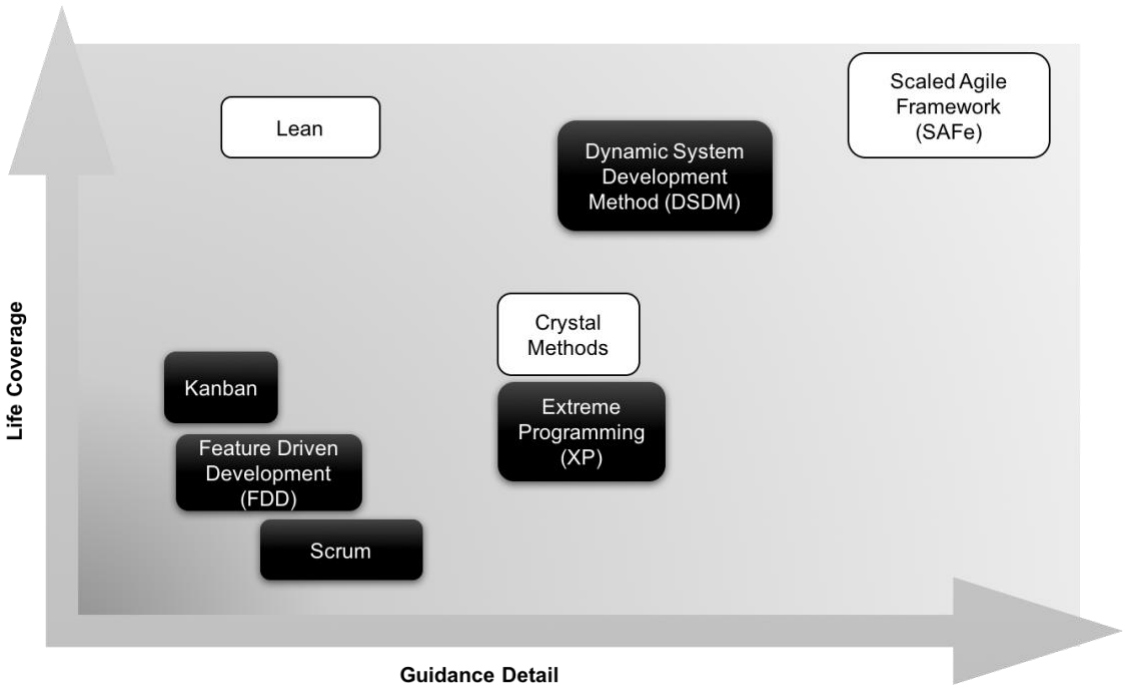

Figure 3. Agile Methods Plotted by Life Coverage and Guidance Detail (PMI, 2017a)

Agile methods imply a set of values and principles. The principles and values are included in the Agile Manifesto and they are the result of the analysis of methods as,

Crystal, Extreme Programming (XP), Dynamic System Development Method (DSDM), Feature Driven Development (FDD), Scrum, etc. An Agile Method is a particular combination of principles, practices, roles and artifacts, not just an arbitrary mix, but a reasoned construction with its own distinctive view of software development (Meyer, 2014). The following of this section presents the most popular agile methods and their main characteristics.

II.1.4.a Scrum

Scrum was first proposed by Jeff Sutherland and Ken Schwaber in 1993 (Cervone, 2011), (Chandra Misra, Kumar, & Kumar, 2010). Scrum is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions based on what is known (Schwaber & Sutherland, 2017). Scrum is therefore defined as a framework which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. This framework consists of scrum teams and their associated roles, events, artifacts and values, each component serves a specific purpose and is essential to Scrum's success and usage. Figure 4 shows the global distribution of scrum framework. Below a description of the different Scrum aspects is given (Schwaber & Sutherland, 2017):

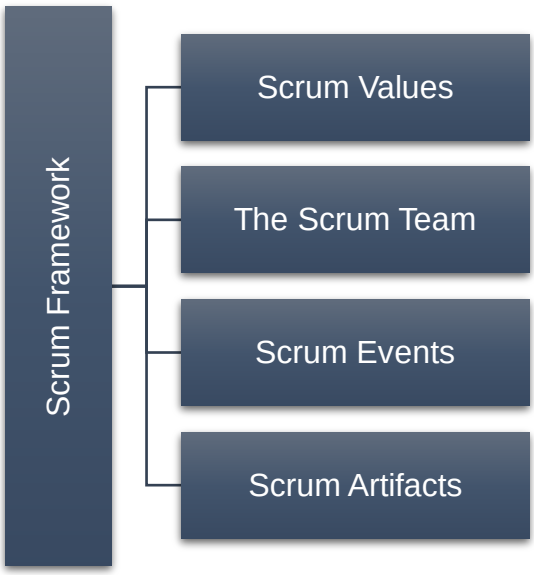

Figure 4. Global organization of Scrum Framework.

Scrum Values

The Scrum values are followed by everyone in the project. It enumerates five values:

V_1 - Courage, Scrum Team members have courage to do the right thing and work on tough problems,

V_2 - Focus, everyone focuses on the work of the Sprint and the goals of the scrum team,

V_3 - Commitment, people personally commit to achieving the goals of the Scrum Team,

V_4 - Respect, Scrum Team members respect each other to be capable, independent people, and

V_5 - Openness, the Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work.

Scrum Events

Scrum Events are specifically designed to enable critical transparency and inspection. Each event is a formal opportunity to inspect and adapt something. All events are time-boxed events and they may ensure an appropriate amount of time without allowing waste in the process. Scrum enumerates five events:

- Sprint – a time-box of one month or less, during which a usable and potentially releasable product increment is created; an increment is the sum of all the items completed during a sprint and the value of the increments of all previous Sprints. The Sprint is the heart of Scrum and a new Sprint starts immediately after the conclusion of the previous Sprint.
- Sprint Planning – a time-box to a maximum of eight hours for a one-month Sprint. Sprint planning plans the work to be performed in each Sprint. This plan is created and maintained by the collaborative work of all members of the project. Sprint planning also defines the sprint goal, an objective set for the Sprint, that can be met through the implementation of the Product Backlog.
- Daily Scrum – a 15-minute time-boxed event, held every day of the Sprint. Daily Scrum is use to inspect progress toward the Sprint Goal and to inspect how progress is trending toward completing the work in the Sprint Backlog.

- Sprint Review – the inspection of the increment. During these inspection stakeholders review the results of a Sprint. A Sprint Review is held at the end of the Sprint and adapts the Product Backlog if needed.
- Sprint Retrospective – an opportunity for the team to review itself and create a plan for improvements to be done during the next sprint. Sprint Retrospective occurs after the Sprint Review and prior the next Sprint Planning.

The Scrum Team

Scrum Teams have as a characteristic "self-organizing and cross-functional". Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies needed to accomplish the work without depending on others. Furthermore, the Scrum Team delivers products iteratively and incrementally, maximizing opportunities for feedback.

A Scrum Team comprises three entities:

1. a Product Owner: one person only and its decisions have to lead the entire team and the organization,
2. a Development Team: professionals who do the work of delivering a potentially releasable increment of product at the end of each sprint, and
3. a Scrum Master: it is responsible for promoting and supporting scrum framework, he does this by helping everyone understand practices, rules, values and scrum theory.

Scrum Artifacts

Scrum Artifacts represent work or value to provide transparency and opportunities for inspection and adaptation; they are designed to maximize transparency of key information. Product Backlog is considered as an ordered list of everything that is known to be needed in the product; the Product Owner is responsible for the Product Backlog. Sprint Backlog is the set of Product Backlog items selected for the Sprint, a plan for delivering the product increment and realizing the Sprint Goal; it makes the work that the Development Team identifies as necessary to meet the sprint goal visible. The Increment indicates the sum of all the Product Backlog items finalized during a

Sprint, as well as the ones finished during all previous sprints. Product Backlog, Sprint Backlog and Increment are considered as artifacts.

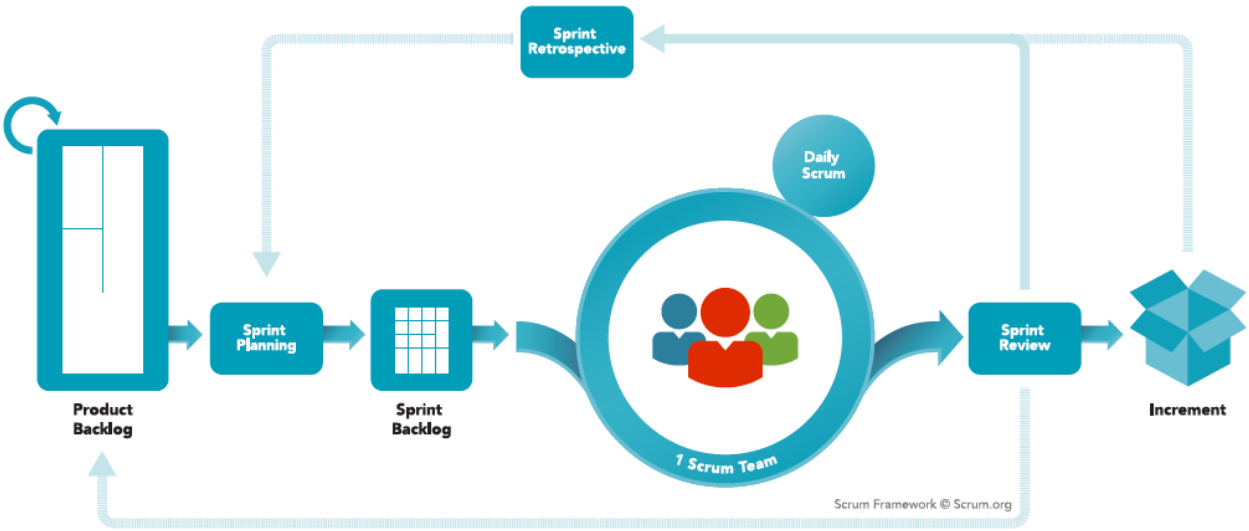

Figure 5. Global Scrum Framework (Schwaber & Sutherland, 2017).

Figure 5 shows the complete Scrum Framework and how each entity is distributed. The framework begins with the Product Backlog; the Product Owner defines all the features that the Development Team has to do and then to prioritize. The next step is the Sprint Planning; here the Product Owner and the Development Team plan which features of the Product Backlog have to be done and the time of each Sprint, which results in the Sprint Backlog. The Development Team has to do each Sprint defined in the Sprint Backlog. They have to hold daily meetings to organize the work that has to be done. Sprint Review allows the Development Team to identify issues that occurred during the sprint development, the issues encountered are evaluated during the sprint retrospective, and the cycle begins again.

Scrum Framework gives a good guidance to develop products, its adaptation to problems, during the product development cycle, is a strong characteristic. However, stakeholders experience plays a very important role in this method. Even if the method proposes the Sprint Retrospective, the issues found in the Sprint Review can impact the whole project because the definition of the features were not well defined.

II.1.4.b Extreme Programming (XP)

In the mid-90s, The Chrysler Comprehensive Compensation Program (C3) was started, and then switched to an Extreme Programming Project performed by Kent Beck. This project helped to define the XP framework. Kent Beck goal was to improve the performance of the system. Extreme Programming (XP) is an agile software development framework and was created in response to problem domains whose requirements change. The name is based on the philosophy of distilling a given best practice to its purest, simplest form, and applying that practice continuously throughout the project (Agile Alliance, 2018; Cockburn, 2017; PMI, 2017a). Kent Beck introduces XP as a basic cycle where repeat until the team and the customer are happy, when it works, look for any damage, and apply techniques to solve. The mindset of that cycle is “increment then simplify” (Meyer, 2014). Figure 6 shows the global distribution of XP framework. Below a description of the different XP aspects is given (Agile Alliance, 2018; Cockburn, 2017):

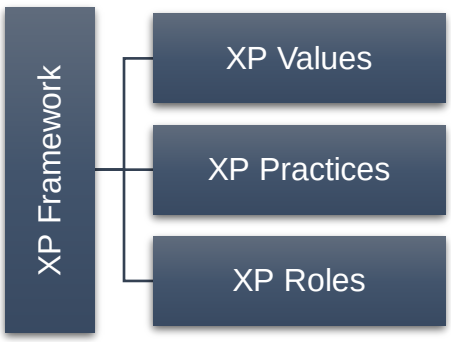

Figure 6. Global distribution of Extreme Programming Framework

XP Values

The core of values of XP framework is based as follows:

1. V_1 - *Communication*, Team communicate face to face daily and work together on everything from the requirements to code, knowledge is transfer from one team member to everyone else on the team,
2. V_2 - *Simplicity*, do what is needed and ask for what is needed, but no more, are the characteristics of simplicity. Avoid waste and do only necessary things to

keep the design of the system as simple as possible, do not try to predict the future,

3. *V_3 - Feedback,* through constant feedback about previous efforts, teams can identify areas for improvement, make any changes needed, and ten adjust the product going forward,
4. *V_4 - Courage*, it is defined as effective action in the face of fear. Do not document excuses for failure, accept and act on feedback are examples of courage,
5. *V_5 - Respect*, everyone gives and feels the respect they deserve as valued team member, provide and accept feedback that honors relationship's team members, and to work together to identify simple designs and solutions.

XP Practices

Interconnected set of software development practices are the core of XP. The method was first formalized as a set of twelve primary practices (planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-hour week, on-site customer and coding standard), but then gradually evolved to adopt several other practices(Agile Alliance, 2018)(Jeffries, 1998). Figure 7 shows the basic eXtreme Programming practices, this figure is distributed in three colors. The red practices (external circle) indicate every contributor in the project (customer, team) and how their interactions are established (planning game and small releases). The green practices (middle circle) are the basis that the team should follow throughout the project, and the blue practices (center circle) are technical aspects while teams are programming software.

The distribution of XP practices, in Figure 7, proposes three stages to integrate XP practices, first the external circle, is dedicated to the planning of the activities during the project, in this stage the customer presents the desire features and plans, with the whole team, how the features will be release and test, the middle circle enclosed the practices dedicated to the quality of the project, and how features have to be done. In this stage the use of common standard, a common vision, and system fully integrate, are the main characteristics to be considered. Finally, XP proposed the center circle, these practices consider more technical aspects, and they are focused on code production. Figure 7 reminds the XP mindset "increment then simplify", that means that

work in short cycles, and test can simplify the features proposed by the customer to accomplish the project.

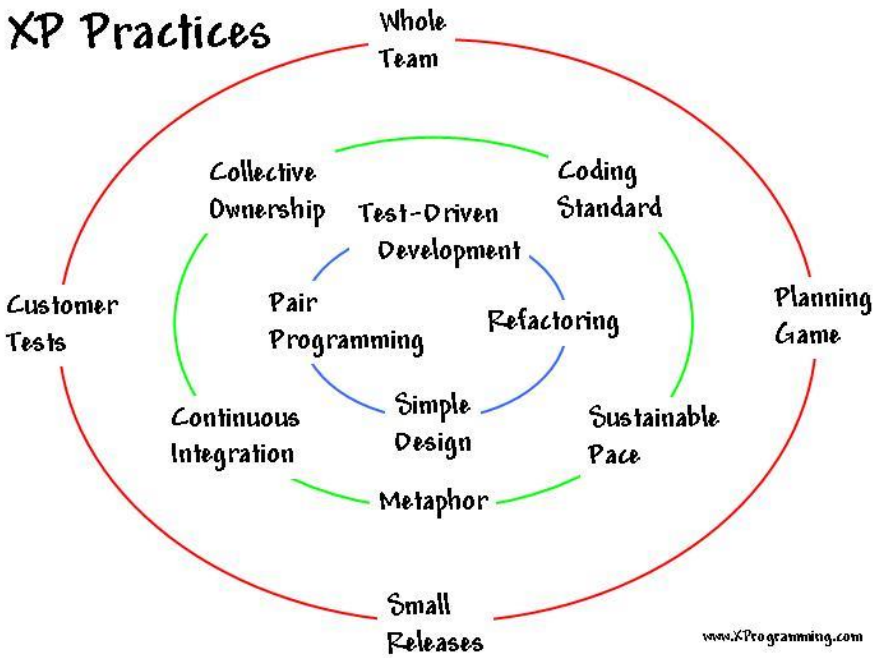

Figure 7. Basic Extreme Programming Practices (Jeffries, 1998).

Whole Team, XP teams use a simple form of planning and tracking to decide what should be done next, and predict when the project will be done.

Planning Game, it includes two planning steps, first the *release planning:* the costumer presents the desire features to the developers, and developers estimate their difficulty. It helps to estimate the project cost and to know the importance of the project features. Initial release plans are needed but imprecise because neither the priorities nor the estimates are truly solid. The second step is the *iteration planning:* during this step, the costumer presents the features desires, and the team build and deliver running useful software in two-week, that is called "iteration". These planning steps provide very good information and excellent steering control in the hands of the customer.

Small Releases, XP teams practice small releases in two ways. First, in every iteration, the team releases running, tested software and business value. This keeps everything open and tangible for the customer. Second, XP teams release to their end users frequently as well.

Customer Test, the XP customer defines one or more automated acceptance test to show that the feature is working. The teams have to build the customer tests and uses them to improve to themselves.

Collective Code Ownership, giving many people's attention to all code increase code quality and reduces defect. Any pair of developers can improve any code at any time.

Coding Standard, a common coding standard is followed by XP teams, all the code in the system looks as if it was written by a single individual.
Sustainable Pace, XP teams are in it for the long term. They work hard, and at a pace that can be sustained indefinitely.
Metaphor, a metaphor is a common vision, that XP teams develop, of how the program works.
Continuous integration, XP teams keep the system fully integrated at all times.
Test-Driven Development, it means that XP teams should work in very short cycles of adding a test, then making it work. Teams can produce code with nearly 100% test coverage.
Refactoring, it is focused on removal of duplication, and on increasing the cohesion of the code. Continuous design improvement and test evolve refactoring.
Simple Design, XP teams build software to a simple but always adequate design.
Pair Programming, software is built by two programmers, sitting side by side, at the same machine. This practice ensures that all production code is reviewed by at least one other developer, and results in better design, better testing, and better code

Table 1. Basic eXtreme Programming practices description (Jeffries, 1998).

Table 2 shows the evolution of the basic XP practices. This table is distributed in four categories (organizational, technical, planning, and integration), each category lists the originally primary practices and the evolution of this practices (secondary practices) (PMI, 2017a):

Practice Area	Primary	Secondary
Organizational	• Sit Together • Whole team • Informative workspace	• Real Costumer Involvement • Team Continuity • Sustainable pace
Technical	• Pair Programming • Test-First Programming • Incremental Design	• Shared code/ collective ownership • Documentation from code and test • Refactoring
Planning	• User stories • Weekly cycle • Quarterly cycle • Stack	• Root cause analysis • Shrinking teams • Pay per use • Negotiated scope contract • Daily standups
Integration	• 10-minute build • Continuous integration • Test-first	• Single code base • Incremental deployment • Daily deployment

Table 2. Practices of eXtreme Programming (PMI, 2017a).

XP Roles

A guide for the definition of roles in XP framework is not well defined. The roles typically found in projects behave on XP projects are four:

1. *The customer*, a single person who is responsible for making all the business decision regarding what should the system do, how to know when the system is done, what is the available funding and in what order features of the system are delivered. The XP Customer is expected to be actively engaged on the project and ideally becomes part of the team,
2. *The Developer,* it is responsible for realizing the stories identified by the customer, the stories are sentences written by the customer terminology without techno-syntax,
3. *The Tracker,* this is often one developer who spends part of its time tracking relevant metrics that the team feels necessary to track its progress. This role also identifies areas for improvement and is not a required role for the team. If the team determines a need for keeping track their metrics, the tracker role should be defined.
4. *The Coach,* this is usually an outside consultant or someone from elsewhere in the organization who has used XP. It is included in the team as a mentor of all team members. It helps the team to maintain discipline during the project and to avoid mistakes that newest teams make.

XP framework is based primarily on the use of several practices, these practices allow teams to develop small and medium projects software. Each practice gives an overview of the factors to be considered, but the framework does not provide a detailed guide on how integrate them or how to deploy them. Based on Table 1, and each practice description, the framework can be started in the external circle (red one, Figure 7), following the customer input, the definition of the planning game can be done, and continue with the definition of the small releases that the whole team have to do. Once the planning is defined (how the features should be done), the middle circle is activated and so on.

Another aspect identified in this framework is that roles are not well related to the practices, even if they are well defined. That means, for example, the tracker spends

part of its time tracking relevant metrics that the team feels necessary to track its progress, but it is not mentioned what practices are more useful to achieve that. Another example, feedback is defined as a value where teams can identify areas for improvement, make any changes needed, and then adjust the product going forward, but this framework does not establish a room or a step to discuss about changes or issues found during the product development.

II.1.4.c Kanban

Kanban was proposed by Taiicho Ohno and applied at the main Toyota manufacturing facility in 1953. The word 'Kanban' can be translated from Japanese as '*visual sign'* or '*card'*. Kanban method is a means to design, manage, and improve flow systems for knowledge work. This method differs from the others agile methods because it does not recommend the use of time-boxed iterations. Kanban method is normally used in situations where work arrives in an unpredictable fashion and/or when a team or organization is in need of flexibility, focus on continuous delivery and increased productivity and quality, increased efficiency, team member focus, variability in the workload and reduction of waste conditions (Agile Alliance, 2018; PMI, 2017a). Figure 8 shows the global distribution of Kanban method. Below a description of the different Kanban aspects is given (Agile Alliance, 2018; PMI, 2017a):

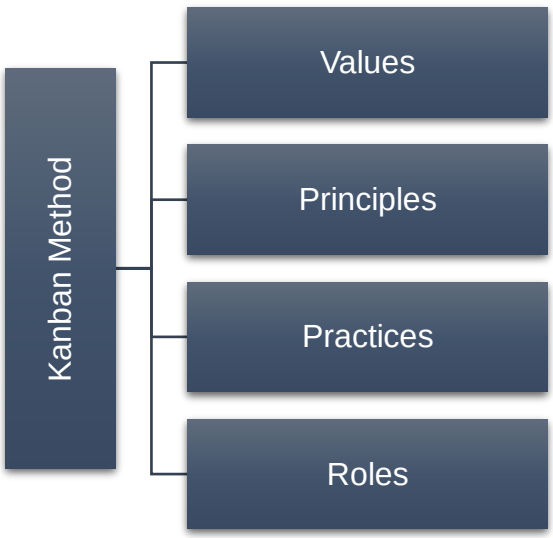

Figure 8. Global distribution of Kanban Method

Kanban Values

Kanban values are used by teams that want to improve the services they deliver; the core values are nine:

V_1 - Transparency, openly sharing information and the use of clear language improve the transparency in the slow of business value,

V_2 - Balance, in order to achieve effectiveness, different aspects, viewpoints and capabilities must be balanced,

V_3 - Collaboration, people should work together,

V_4 - Customer focus, optimized flow of value to customers is important,

V_5 - Flow, work is a continuous or episodic flow of value,

V_6 - Leadership, it is needed in order to realize continuous improvement and deliver value. It is considered as the ability to inspire others to act via example, words, and reflection,

V_7 - Understanding, to move forward and improve, the starting point must include individual and organizational self-knowledge,

V_8 - Agreement, everyone involved in the project are committed to improvement and agree to move toward goals, respecting differences of opinion and approach,

V_9 - Respect, value, understand, and show consideration for people.

Kanban Principles

Kanban principles are structured in the change management principles: to address the human tendency to resist change, and the service delivery principles: organizations are recognized as a collection of independent services, and to place the focus on the work, not the people doing work. Table 3 shows the distribution of Kanban principles.

Change Management Principles	**Service Delivery Principles**
1. Start with what do you know 2. Agree to pursue improvement through evolutionary change 3. Encourage acts of leadership at every level	4. Understand and focus on customer needs and expectations 5. Manage the work; let people self-organize around it 6. Evolve policies to improve customer and business outcomes

Table 3. Kanban principes distribution *(Agile Alliance, 2018)*.

Kanban Practices

Kanban method includes six essential practices; they are described as follows:

1. *Visualize*, in order for the visualization, Kanban systems should show the commitment point, the delivery point and the polices that determine what work should exist in a particular stage. In the commitment point the team agrees to work in a specific item of the process, the delivery point is when the team delivers the work item to a customer,
2. *Limit work in progress,* limits are periods of time that the Kanban teams set according to the amount of work in progress, those limits help to start new items,
3. *Manage flow,* Kanban teams use empirical control through transparency, inspection and adaption in order to balance a potentially conflicting goal,
4. *Make polices explicit,* explicit polices help explain a process beyond the list of different stages in the workflow. They should be sparse, simple, well-defined, visible, always applied, and readily changeable,
5. *Implement feedback loops,* feedback loops look to provide evolutionary change, they are an essential element in any system.
6. *Improve collaboratively, evolve experimentally,* instead of trying to reach a predefined finish goal, Kanban method proposes to start with the processes as it currently exists, then applies continuous and incremental improvement.

Kanban Roles

During the practice of Kanban method, two roles have emerged to serve particular purposes:

1. *a Service Request Manager*, this role understands customer needs and expectations, it also facilitates, select, and order work items,
2. *a Service Delivery Manager*, this role is responsible to deliver select items to customers, it is also responsible for the flow of work of select items.

Completing the work is more important than start in a new work, teams work together to implement and adhere to the work in progress and add value is derived from completed work; that reflects Kanban mindset.

II.1.4.d Crystal

Crystal is not one specific methodology, is more a family of methodologies use for software development. It was defined by Alistair Cockburn. Cockburn approach was focused on interviewing as many projects as possible. He wrote down team's opinion about what was important to their success or failure. Crystal is based in project characterized along two dimensions: *criticality (considered as the potential for the system to cause damage), and size (number of people involved in the project)*. These methodologies were characterized according to the number of people involved in the project, and labeled with a color (Cockburn, 2017; Meyer, 2014; PMI, 2017a; Stark & ClydeBank Business, 2016).

Figure 9 shows the distribution of Crystal methodologies and illustrates how to determine which Crystal methodology to use for a project. As the project size increases (moves to the right in the figure), the harder the project, and hence a more comprehensive (darker color) of Crystal is necessitated. As the criticality of a project increases (moves from the bottom to the top of the figure), aspects of the methodology need to be put in place to accommodate the additional requirements, including the artifacts generated by the team; however, criticality does not affect the color of Crystal used. One characteristic of Crystal is its intentional scaling to projects, based on the size and critically. The larger a project gets the darker the color (Coffin & Lane, 2016).

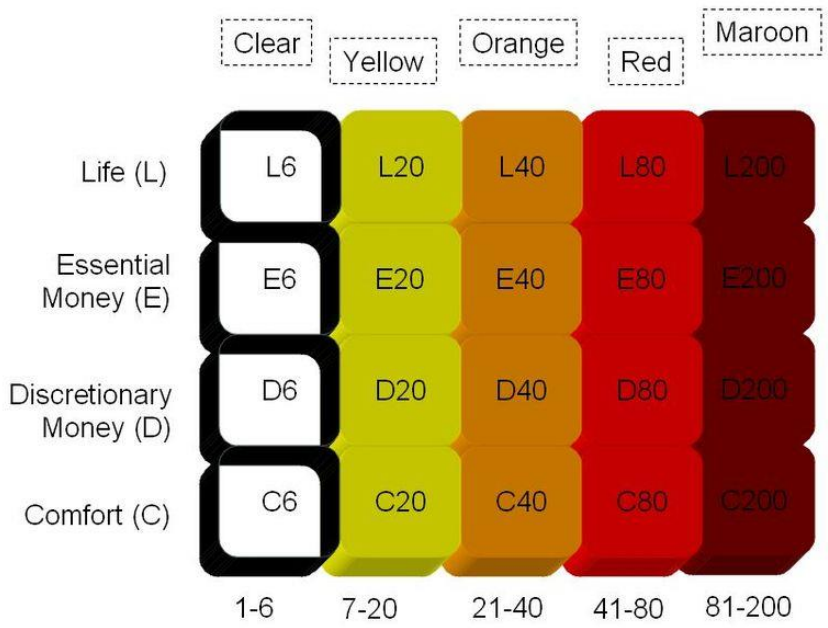

Figure 9. The crystal family of methodologies (Coffin & Lane, 2016).

Crystal “clear” covers small projects (1 to 6 people involved), between 7 and 20 people involved, Crystal “yellow” is used, Crystal “orange” was the first developed to address larger projects, it covers 21 to 40 people, and Crystal “red” involves 41 to 80 people. Crystal mindset is focus in “osmotic communication” or “core communication”, that means questions and answers flow naturally and with surprisingly little disturbance among the team (Meyer, 2014).

Crystal Values

Crystal methods are focused in maximizing the potential of each person on the project team (Stark & ClydeBank Business, 2016). The core values are (PMI, 2017a):

1. V_1 - People,
2. V_2 - Interaction,
3. V_3 - Community,
4. V_4 - Skills,
5. V_5 - Talents,
6. V_6 - Communications

Crystal Principles

Crystal defines seven principles as follows (“Crystal Methods - Wikiversity,” 2017; Meyer, 2014; PMI, 2017a):

1. *Frequent delivery,* it means constantly release of the software program, the release times depends on the project size. It is considered as an important property in the project,

2. *Reflective improvement,* it involves the team members, they take a break to do a retrospective of how things are working in the project. Feedback helps to find ways to better improve their process in each project iteration,

3. *Close or osmotic communication,* it promotes a constant and free flow of information between team members. Questions can be rapidly answered by using this type of communication. The team members can know what is going out in the project,

4. *Personal Safety,* team members should be free to speak up, without fear of reprisal. Members team must be able to trust each other and feel free to speak up about issues or whatever arises,

5. *Focus,* it defines the conditions under which developers can perform their jobs, first, it refers on an individual task in the project, and then, it refers to the direction of the project,

6. *Easy access to expert users,* developers requires a realistic guarantee of access to knowledgeable user representatives. There should be a minimum of at once a week, two-hour meeting with the expert user, and the ability to make phone calls to the expert user too,
7. *Technical environment with automated test, configuration management, and frequent integration.*

Crystal is a concentrate of software development wisdom; it means that Crystal do not propose a step by step guide to follow.

II.1.4.e Dynamic System Development Method (DSDM)

DSDM is considered as an agile project delivery framework. Created in 1994, it emphases on constraint-driven delivery and was developed as a non-commercial collaboration among industry leaders. DSDM is effective on small solutions or large complex corporate projects, that includes non-Information Technology (IT)-solutions and non-software projects. It can be used in any business, in any technical environment for any project. The philosophy and principles of DSDM help shape the Agile Manifesto. Projects aligned to clear business goals, frequent delivery and collaboration of motivated and empowered people are the mindset of DSDM. DSDM tactics includes: MoSCoW prioritization, it helps to identify priorities in the project by using different concepts (Must, Should, Could and Won't) and time-boxing, it establishes the desired quality, expense, and time frame of the project at its initiation (Agile Business Consortium, 2015; PMI, 2017a; Stark & ClydeBank Business, 2016).

DSDM principles

DSDM propose eight principles, these principles are supported by products, process and practices. DSDM principles help direct and shape the attitude of a DSDM team, they are defined as follows (Agile Business Consortium, 2015):

1. *Focus on the business need,* DSDM teams should understand business priorities, establish a valid case, ensure continuous business sponsorship and commitment,

2. *Deliver on time,* delivering a solution on time is quite often the single most important success factor. To deliver on time DSDM teams need to time-box the work, focus on business priorities, always hit deadlines and build confidence through predictable delivery,
3. *Collaborate,* collaboration encourages increased understanding, speed and shared ownership. DSDM teams should involve the right stakeholders throughout the project, encourage pro-active involvement from the business representatives, ensure that all members of the team are empowered to take decisions and build a one-team culture in order to fulfil this principle,
4. *Never compromise quality,* all work should be aimed at achieving the level of quality defined at the start of the project. DSDM teams need to agree the level of quality before development starts, ensure that quality does not become a variable, test in all appropriate levels, do constant review and design and document appropriately,
5. *Build incrementally from firm foundations,* the establishment of firm foundations for the project, before committing to significant development, is a key factor to consider while using DSDM framework. To create strong foundations DSDM teams need to carry-out appropriate analysis and enough design up front and formally re-assess priorities and informally re-assess ongoing project viability with each delivered increment,
6. *Develop iteratively,* the concept of iteration is at the heart of everything developed as part of DSDM framework. In order to fulfill this principle, DSDM teams *need to build* business feedback into each iteration, recognize that most detail should emerge later rather than sooner, embrace change and use iterative development to encourage creativity, experimentation and learning,
7. *Communicate continuously and clearly,* it involves effective communication between teams and individuals in the project. Encourage informal, face-to-face communication at all levels, run daily team sessions, always aim for honesty and transparency in all communication and Demonstrate the Evolving Solution early and often are some points that DSDM teams need in order to fulfill this principle,
8. *Demonstrate control,* it is essential to be in control of a project at all times, to achieve that, DSDM teams should make plans and progress visible, measure progress through focus on delivery of products, manage proactively, evaluate

continuing project viability and use an appropriate level of formality for tracking and reporting. The roles involved in this principle are the Project Management and Team Leader

DSDM Roles

DSDM roles are distributed in two dimensions: *interests and categories*. In DSDM projects, interest is represented: roles representing the business view (focus in business interests), roles representing the solution/technical view (focus in solution/technical interests), roles representing the management/leadership view (focus in management interests) and roles representing the process view (focus in process interests). DSDM groups roles in three categories: Project roles, Solution Development Team roles and Supporting roles. Figure 10 shows three categories for the role distribution, first, the project-level roles (Business Sponsor, Business Visionary, Technical Coordinator, Project Manager and Business Analyst) are the directors, managers and coordinators of the work for the project, and they are also responsible for the governance of the project. Second, the solution development team-level roles (Team Leader, Solution Developer, Solution Tester, Business Ambassador and Business Analyst), in this level, Business Analyst role is intentionally positioned as well as part of the project-level.

This allows help the business to formulate the Business Case, and also to be involved in assisting the business in defining their requirements during feasibility and foundations, sometimes before the full Solution Development Team. is assigned. The role then continues in supporting the Solution Development Team alongside the project-level roles, as the more detailed requirements emerge. Finally, the supporting-level roles, (Business Advisors, Technical Advisors, Workshop Facilitator and DSDM Coach) are defined. The roles help and guidance to the project, as noticed in the Figure 10, Technical Advisor role and Business Advisor role, are shared with the solution development team-level.

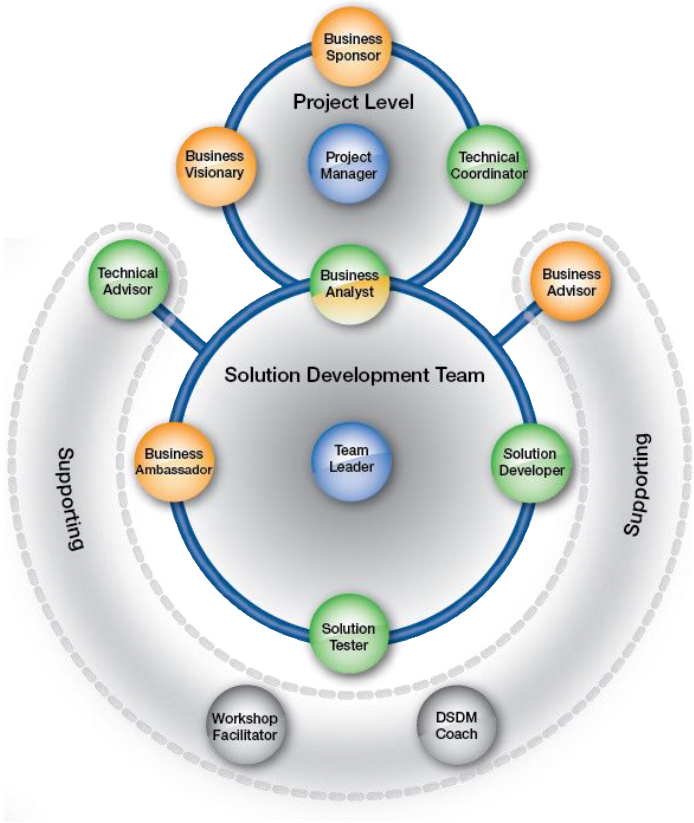

Figure 10. Roles of the DSDM framework (Agile Business Consortium, 2015).

Business Interests | Solution/technical interests | Management Interests | Process Interest

The mix of colors is a role that straddles two separate areas of interest.

Dynamic System Development Method allows the management of different size projects, non-software projects are included. The framework will set cost, quality, and time at the outset, and then use formalized prioritization of scope.

II.1.4.f Feature Driven Development (FDD)

FDD was developed to meet the specific needs of a large software project. It was first introduced in 1999 by Jeff DeLuca and Peter Coad. Features are the most important aspect of FDD, a feature is considered as a small client -valued function expressed in the form: action, result, object. FDD is proposed in five main activities, performed iteratively, a brief description is given bellow (Ambler, 2014; PMI, 2017a).

1. *Develop an overall model,* at the start of a project, one goal is to identify and understand the fundamentals of the domain that the system is addressing, that reflects what the team will build. It helps to define a high-level object model and notes,
2. *Build a feature list,* it is a group of features list grouped into sets and subject areas,
3. Plan by feature, it is a development plan class owner,
4. Design by feature,
5. Build by feature.

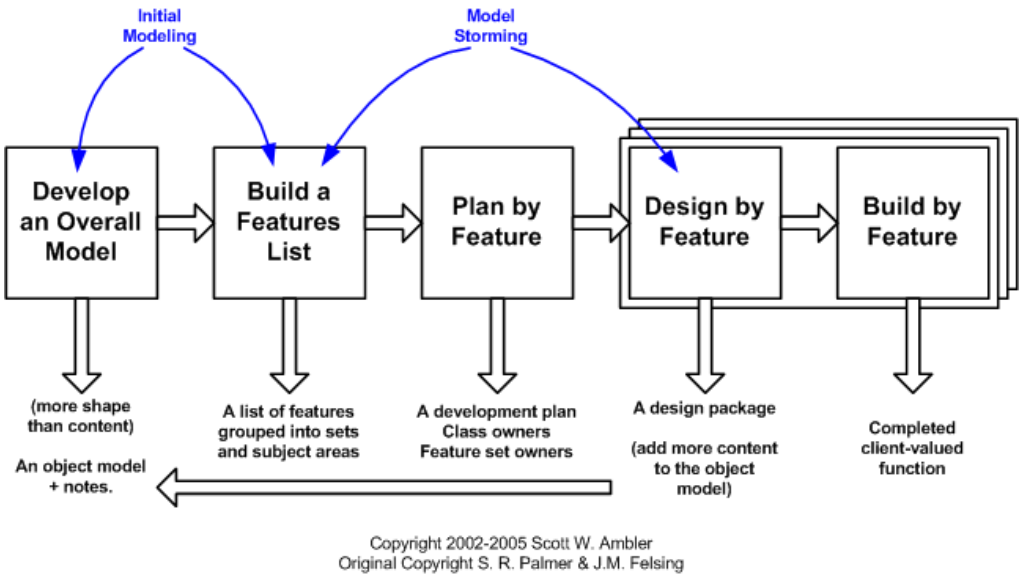

Figure 11. FDD life Cycle (Ambler, 2014).

Figure 11 shows the life cycle of FDD. As can be noticed, there are five main activities that are performed iteratively. The FDD cycle starts developing an overall model, the result of this activity is an object model and notes, that means that is most important to understand and to identify the fundamentals of the domain that the product is addressing to create a shape. In that case the content is not a priority. The second activity is to build a features list, the output of this activity is a list of features grouped into sets and subject areas. These two first steps map to the initial modeling effort. Next activity is focused to stablish a plan by feature, the identification of the class owners and feature set owners is the result. FDD project starts by performing the first three activities to identify the scope of the effort, the initial architecture, and the initial high-level plan. The majority of the effort on an FDD project, is comprised in the fourth and fifth activities (Design By Feature and Build By Feature). These two activities

include tasks such as detailed modeling, programming, testing, and packaging of the product. If it is necessary to add more content to the object model, before to complete a client-valued function, FDD cycle proposes to go back to the first activity. Construction efforts traditionally occur in two-week iterations, with the team iteratively working through all five steps as needed.

FDD roles

There are six primary roles on an FDD project (Ambler, 2014):

1. Project Manager,
2. Chief Architect,
3. Development Manager,
4. Chief Programmer,
5. Class Owner, and
6. Domain Expert.

FDD practices

FDD activities are supported by a core set of software engineering best practices, these practices are listed as follows (PMI, 2017a):

1. Domain object modelling,
2. Developing by feature,
3. Individual class ownership,
4. Feature teams,
5. Inspections,
6. Configuration management,
7. Regular builds, and
8. Visibility of progress and results.

II.1.5 Domains and Levels of Deployment of Agility in enterprise

Companies live in an era where products and services are evolving rapidly, that makes them face radical changes in project and program management. The ability to adapt to these radical changes is considered to be very important for every organization in order to stay competitive and profitable in an economic environment. The search for constant innovation has transformed “change” from an exception into a rule and the

terms ‘Enterprise Agility’, ‘Product Development Agility’ and ‘Project Management Agility’ will become a common goal for project and program managers, strategist and executives (Conforto et al., 2014).

Changes are often difficult to manage; large changes can be even more painful for a company. There are aspects to be considered for the adoption of an agile approach in systems engineering. These aspects can be: managing interoperability between equipment that may need to work together in undefined future scenarios, strong variability, evolution capacities of needs, conditions of the environment in which the system is designed, build and maintained (Alain ROUSSEL, 2016).

The use of an agile approach as a tool should answer to the challenges that systems engineering faces in the development of a system, the integration of systems engineers into the project organization, the specific delivery of a system (for example, operational capacity). These are some points that we should consider in order to adopt an agile approach in systems engineering. As a result of first analysis and different meetings with people that works with agile methods, we identify that the challenges of this research are distributed in three different levels: the organizational level, the project level and the product level (Kruchten, 2013).

II.1.5.a Organizational level

In this level we can identify different aspects as:

a. Cultural aspect: national and corporate culture may affect development process. Experts, in this domain, identify that one of the biggest problems using an agile approach in systems engineering is related with cultural aspects in the organization. Other aspects that Alain ROUSSEL point out are the structure of the organization (human’s resources, distribution of departments, etc.), the tools (the organization should use tools that allow them to share their experiences in all levels of an agile environment) (Alain ROUSSEL, 2016),

b. The Business model: companies have traditional business model; agile approaches have a fixed-duration and a fixed-cost project plan and contract that means that agility should be present in contract definition.
c. The Number of instances: normally systems are developed for different instances, agility should help to manage the interactions between instances.

II.1.5.b Project level

In this level the aspects to be considered are:

d. The Size: the overall size of the systems is the main factor; it influences team size.
e. The Team distribution: definition of the specific roles and responsibilities are important in systems engineering. Agile roles are different from the traditional roles. The team works in a collaborative way, there is no a strong hierarchy.
f. Stable structure: Agile methods are used for stable architectures. This characteristic is not often present in the development of a complex system (Kruchten, 2013).

II.1.5.c Product level

Seeing the system as a product, there are technical processes that impact the development of the system, and the idea of added value in the system have to be defined, requirements is another example or the product level, for the system development we have stakeholders needs and requirements definition process, stakeholders needs and requirements could change, the technical team should respond to change more than requirements definition.

This work will explore and analyze the project level. Software projects present the same challenges. The Standish Group has as a mission to change the world in the way software projects are managed, and every year, since 1994, they publish the

CHAOS report. In 2015, they analyzed 50 000 software projects around the world and they compare project outcomes between agile and traditional waterfall projects, the CHAOS database was segmented by the agile process and waterfall method, the total number of software projects were over 10 000, the result is shown in Figure 12 (Hastie & Wojewoda, 2015). Globally, in all size projects, projects developed in an agile way were more successful than traditional waterfall projects. It also shows that all size projects are less challenged using agile methods.

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

Figure 12. CHAOS resolution by agile versus waterfall (Hastie & Wojewoda, 2015)

The report includes an enhanced definition of factors of a successful project: executive support, emotional maturity user involvement, optimization, skilled staff, agile proficiency, modest execution, project management expertise, clear business objectives (Hastie & Wojewoda, 2015). These factors should be considered to have successful projects in terms of time and budget with a satisfactory result. The point of view of the Standish Group is focus on software projects, and this view shows that the percentage of the successful projects developed using a waterfall method is still an issue to be consider, in another hand, projects developed with an agile method seem to be a good option to manage projects in software engineering, the percentage of success projects using an agile method are higher than successful projects using a waterfall method and less changed. This analysis could help us as an input to consider agility as a tool to perform projects in other contexts like systems engineering.

II.1.6 Agility Today

Agile approaches became very popular in the last 15 years, as presented above, their iterative and incremental software development techniques allow a better percentage of success projects. Advancements in agile culture and methods over the last decade exposed the need for a more holistic approach to the end-to-end software delivery lifecycle (Ehle, 2018).

II.1.6.a The agility essence in development process

What's agility? A new fashion? A buzz? A list of practices? The invention of new techniques, new concepts? In fact, agility brings together the good principles and good practices that produce good results, in general, everything that has worked well for 40 years into a coherent and robust whole.

Agility is an approach, a state of mind. The principles and best practices, present in this approach, promote the creation of maximum value for clients, better and faster adaptation to change, the assurance of having initiated and maintained a continuous improvement process, and the assurance of making the most of the collective intelligence of an organization. The agile approach is strongly based on all that is related to the human behavior; it aims to make a way of functioning more natural. Its primary objective is not to go fast, but rather to produce a maximum of business value, in high quality and industrialized increments, and thus reduce the tunnel effect.

Agile mindset involves six key concepts:

- Make “small”; it proposes to set small, clearly identified objectives that we know how to achieve, and on which a commitment will be possible (and will make sense!),
- Make “complete”; it offers to make complete application tips, 100%, to save time while having the right to error and accepting change. This implies that the quality is the best possible and that the expected business behavior is constantly

checked. Functionality is preferred to functionality completion (for example an operational deliverable even if a little degraded on one or two points),

- Short iterations; teams switch from a predictive mode to an adaptive mode (multiple iterations, from 2 to 4 weeks, linking specification-design-integration and test),
- Collaboration and mutual assistance; Clear roles to face together,
- Sharing the same vision; the alliance around the same vision, with the same objectives, the same language, the same constraints (budget, deadline, organization, etc.). A vision carried collectively, led by a representative of the client within the team for the functional aspects,
- Continuous improvement; learning by walking. Knowledge comes first from experience. At the end of each project, take stock of what went well (the Bests) to reproduce it and the points for improvement (the Concerns).

In the literature we find many preconceived ideas about agility, however they can easily be addressed and answered:

- Agility is not creating at a lower cost, it is creating more value and delivering it sooner,
- Agility is not a magic wand; agility does not solve problems; it allows them to be identified as soon as possible.
- Agility is common sense, it implies thinking differently and special support,
- Agility is the end of management; in agility there is no longer a leader that centralize the hole project, but there is always a manager, a leader or a facilitator at the service of the teams,
- Agility does not mean the end of documentation; on the contrary the product evolves regularly, so, it is necessary to maintain a necessary and sufficient documentation and to capitalize on this documentation will be built by the team and for the team,
- Agility does not mean do whatever you want! agility is a set of practices, ceremonies, formalized roles. Agility is a real discipline, both in terms of organization and engineering. This discipline, well carried out, allows with the wire of iteration a real control of the project and the quality of the product; on

the other hand, lowering the level of requirements on these practices is a risk of failure.

The essence of agility is now spread into a technical community, but agile has not completely been able to fulfil operational aspects. A new mindset, called DevOps, appears as an alternative to face rapid and continues adaptation doing tests.

II.1.6.b Agility extended to operations: DevOps

DevOps is about using coordination and automation to deliver higher quality software, faster. It requires being instantiated to specific organizational contexts, instead of applying a specific set of technologies. DevOps integrates Lean philosophy, in order to face operational challenges in software development. Lean thinking is focused on the creation of the value without waste, an overview of Lean philosophy is given in section II.2. Waste time, space and effort in any process impact the performance of development. As almost of agile methodologies, DevOps contains core values, patterns and practices that allow developers to test their product more efficiently (Agile Alliance, 2018).

DevOps values

1. *Empowered individuals,* empower people to do the right things,
2. *Accountable,* everyone is responsible for quality. Quality is everywhere in the project, build it and runt it,
3. *Teamwork,* coordinate across functions at all times and respect skill sets.
4. *Trust,* trust in each other and in management, trust requires everyone to be working in the same values and objectives, trust but verify,
5. *Transparency,* clear knowledge of long-term plans, teams have to have a visibility into objectives and priorities. Shares access to metrics and source code,
6. *Continuous learning,* improvement, this requires introspection, identifying root cause problems, sprint and incident retrospectives are great source to continuous improvement,
7. *Feedback loops,* this considers structured ways to solicit feedback from all stakeholders,

8. *Data-Driven decisions,* upfront data capture and analysis capability drives good decisions,
9. *Standardization,* prioritize repeatable actions through automation, use of standards,
10. *Customer-centric,* focus on value to the customer and creates empathy for the customer.

These values are covered by specific techniques used as part of implementing the above concepts and processes.

II.2. Lean Overview

Lean was first implemented in manufacturing process, and it is considered as a philosophy that uses practices to create value without waste in product development (Oehmen et al., 2012). Lean success (making industrial production more efficient by not building any unneeded part or product, reducing waste at every step and minimizing unnecessary communication) allowed authors to apply it in software, product, and systems development (Meyer, 2014). Lean thinking adopts a number of practices and it is organized around six concepts. These concepts can be distributed in a house structure (Knaster & Leffingwell, 2017).

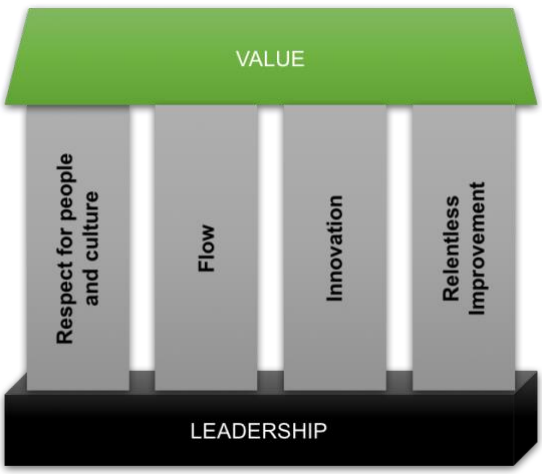

Figure 13. The six concepts in the house of lean thinking (Knaster & Leffingwell, 2017).

Figure 13 presents the house of lean; the roof represents the goal of delivering value, the pillars support this goal through the respect for people and culture, flow, innovation

and relentless improvement (Knaster & Leffingwell, 2017). The main concepts that comprise the lean house are: value, waste, and the process of the creating value without waste. These concepts are present in the Lean principles.

Lean Value.

Value is defined by the costumer, and the goal is to deliver the maximum value in the shortest sustainable lead time. It represents the customer needs during a system life cycle (Knaster & Leffingwell, 2017; Oehmen et al., 2012).

Waste in Lean

It is the ability to identify and eliminate waste. Womack et al, states that all the activities should follow three categories (Oehmen et al., 2012):

1. *Value-added activities,* these activities should transform information or material, or reduce uncertainty. The customer must be willing for it, and activities are done right the first time.
2. Required (also called necessary) non-value-added activities, these activities are required by law, contract, company, mandate, current technology, or other similar reason.
3. Non-value-added activities, these activities consume resources and create no value. They are pure waste

Lean principles

Lean list six principles, the processes of creating value without waste is captured into these principles (Oehmen et al., 2012):

1. *Value,* capture the value defined by the customer stakeholders. High moral, emotional a physical safety, and customer delight are further goals with economic benefits,
2. *Value Stream,* map all end-to-end linked task, control/decision nodes, and interconnecting flows necessary to realize customer value. Map the value stream (plan the program) and eliminate waste. In this principle the information is considered as a knowledge, and it is created by different task, which flows to other task for subsequent value adding,
3. *Flow,* flow the work through planned and streamlined value-adding steps and processes. Continuous flow enables faster value delivery and constant delivery,

4. *Pull,* let customer stakeholder pull value. This principle varies according to the context in which it is applied. In manufacturing, the JIT delivery (Just in Time) covered this principle,
5. *Perfection,* pursue perfection in all processes. The use of continuous improvement help customer to become a learning organization through continuous reflection and adaptation. Making imperfection visible is a motivation to apply continuous improvement in real time,
6. *Respect people*, respect the people in your program. A Lean enterprise recognizes its people as the most important resource. This principle should extend to relationship with suppliers, partners customer, and the broader community.

Lean thinking was introduced in this section; it can be use in deferent domains. Lean started with Toyota development processes. Toyota defined a way to become a lasting learning organization in which problems are constantly surfaced and team associates are equipped with the tools to eliminate waste. Eliminate waste, creating an add value, is the principal characteristics of Lean. This characteristic was impregnated in the six principles. Currently, Lean can be used in software, manufacturing, program, and product development that allow companies to adapt the lean principles according to their context.

SAFe is a framework that uses Lean philosophy, this framework also integrates agile practices in order to develop complex systems and software in a Lean-Agile way. The next section gives a general overview of this framework.

II.3. SAFe Overview

SAFe is a scaled agile framework, defined by Dean Leffingwell, it first appearance was in 2011, with the first version (V1.0). After having done the documentation of dozens of case study, in software and systems context, its authors defined the version 4.0 and 4.5. These versions are achieving substantial business benefits in terms of increased productivity, increased quality, faster time to market, and measurable increases in

employee engagement and job satisfaction. SAFe is based in three bodies of knowledge (Knaster & Leffingwell, 2017):

- *Agile development,* this body of knowledge is centralized by the Agile Manifesto, it unleased a dramatic and entirely new way of thinking and working for software developers and is a critical part of the SAFe Lean-Agile mindset,

- *Systems thinking*, it is a holistic approach to solution development, which views a system as an interrelated set of elements. Systems thinking considers four key aspects: i) optimizing a component does not optimize the whole system, ii) a system exhibits emergent behavior, which occurs only as a result of the interaction from all the systems elements, iii) a system operates in its own environment, the context in which it delivers its value, iv) the value of a system passes through its interconnections. SAFe understanding considers that individuals, teams, programs, and business units are all part of the product development system,

- *Lean product development*, it is a hybrid of Lean thinking and product development flow, based on the Toyota Production System. Its essence is the continuous evaluation of existing processes whit the goal of eliminating waste and delays. In the last few decades, the science of Lean has been spread from manufacturing to product development, providing an extensive body of knowledge where process can be dramatically improved. The Lean practices that SAFe uses are: i) focusing on understanding the full value stream, the value stream is the sequence of steps needed to take a concept from idea to marker, ii) developing and managing a sustainable flow of value, iii) respecting people and culture, iv) accelerating innovation by releasing a minimum viable product to get fast feedback with the least amount of effort, v) embracing kaizen, that means continuous improvement, and, vi) empowering the leadership.

SAFe has evolved as a proven approach for developing complex systems and software in a Lean- Agile way, and it is based on the three bodies of knowledge described above. This framework proses and describe, in three different configurations, the roles,

responsibilities, artifacts, and activities necessary to implement Lean- Agile development mindset. It also provides a guidance for all the levels of the enterprise that are engaged in solution development, the levels are structured as follows:

- *Team level,* this level describes the structure and activities of the agile teams that build the solution, teams can deliver software, hardware, and any combination. Teams apply software quality practices and hardware quality. Software quality practices includes continuous integration, test-first, refactoring, pair work, and other agile practices, in the other hand, hardware quality is supported by exploratory early iterations, frequent-level integration, design verification, modeling, and set base-design. In this level, each team deliver value, tested and working systems, each team implements user stories, and teams have three or nine members, while using scrum.

- *Program level*, this level is considered as the heart of SAFe, and it is identified by the Agile Release Train (ART). The ART is made up of the agile teams, key stakeholders, and other resources. ART is a self-managing and organizing team of agile teams that plans, commits, and executes together. ARTs have a long-lived structure and mission.

- *Value stream level,* it helps enterprises to build large scale, multidisciplinary software, hardware and complex IT systems. This level includes, an economic framework that provides financial boundaries, solution intent (considered as a repository for current and future solution behaviors), solution context (describes how the system will interface and be packaged and deployed in its operating environment), systems engineering disciplines, capabilities and enablers that describe the large behaviors of the solution, and additional roles to support ARTs.

- *Portfolio level,* this level organizes the Lean-Agile enterprise around the flow of value. Portfolio level includes, strategic themes (business objectives), the enterprise value streams, the Program Portfolio Management (PPM) function (consists in the executive and management stakeholders), epics (initiatives that

require multiple value streams), epic owner (who is responsible for guiding epics), enterprise architects (who work with business stakeholders and solution/ system architects), a Lean-Agile budgeting mechanism that empower decision makers and accelerate value delivery, and finally objective metrics that supports Lean-Agile governance and continuous improvement.

All these levels are supported by the SAFe's foundations. SAFe foundations are the supporting principles, values, mindset, implementation guidance, and leadership roles needed to successfully deliver value.

Core values

SAFe defines four values, they define the belief system and are key to SAFe's effectiveness:

1. Alignment, people act as one unit or team, all pulling in the same direction,
2. Built-in quality, it helps ensure that each solution element, at every increment, achieves appropriate quality standards throughout development,
3. Transparency, it enables fast, decentralized decision-making and higher levels of employee empowerment and engagement. It is the key to building trust and improving performance,
4. Program execution, it requires the active support of Lean-Agile leaders, who combine internal leadership with their orientation toward systems thinking, objective measures, and customer outcomes.

Lean-Agile mindset.

It provides the thinking tools and belief system that leadership needs to guide a successful Lean-Agile enterprise.

SAFe principles.

SAFe practices are grounded in nine principles that include the three bodies of knowledge described above. These principles are described as follow:

1. Take an economic view
2. Apply systems thinking
3. Assume variability; preserve options
4. Build incrementally with fast, integrated learning cycles

5. Base milestones on objective evaluation of working systems
6. Visualize and limit WIP, reduce batch sizes, and manage queue lengths
7. Apply cadence, synchronize with cross-domain planning
8. Decentralize decision-making

Implementing.

SAFe provides an implementation road map that helps organizations in their transformation to become a Lean-Agile technology enterprise.

Lean-Agile leaders.

They are people responsible for the success of Lean-Agile adoption in the organization. SAFe identifies eight key behaviors leaders:

1. Exhibit the Lean-Agile mindset,
2. Lead the change,
3. Know the way and emphasize lifelong learning,
4. Develop people,
5. Inspire and align with mission. Minimize constraints,
6. Decentralize decision-making,
7. Unlock the intrinsic motivation of knowledge workers,
8. Evolve the role of the development manager.

SAFe's foundations are present in all the configurations of SAFe. The configurations are distributed as follows:

- *The Essential SAFe configuration*, it is the most basic configuration of the framework, and the basic building block for all other SAFe configurations. This configuration impacts the team and program level. Team and Program forms an organizational structure called the Agile Release Train (ART), where agile teams, key stakeholders, and other resources are dedicated to an important, ongoing solution mission.

  Figure 14 shows all the elements to be considered in each level of this configuration. In team level, roles are defined as the scrum framework proposition (Product Owner, Scrum Master, Development Team), and the

program increment planning is defined by using a mix of different agile methods (Scrum, XP, Kanban). According to the PMI definition of “program”, this configuration is mainly focus in a group of projects that are managed in a coordinate way to obtain benefits, these benefits can be obtained in an agile way using the essential SAFe configuration.

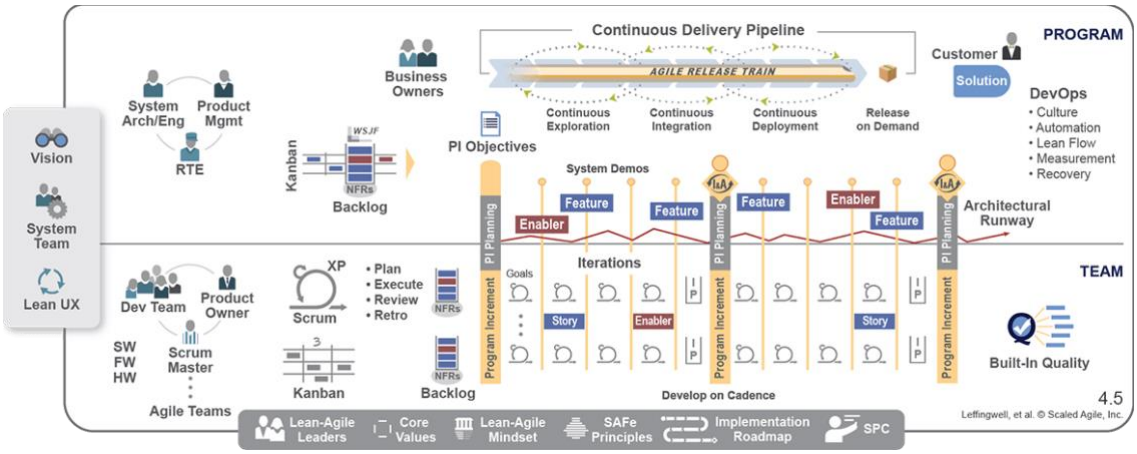

Figure 14. Essential SAFe configuration (Knaster & Leffingwell, 2017).

- *Three-level SAFe configuration*, this configuration is intended for solutions that implies a modest number of agile teams (perhaps 5-10). This configuration adds a portfolio level (see Figure 15), metrics and shared services are also included, Lean Management is present in this configuration, that means that three-level SAFe integrates the value stream by eliminating waste, and that mindset is share with the teams involved in this configuration.

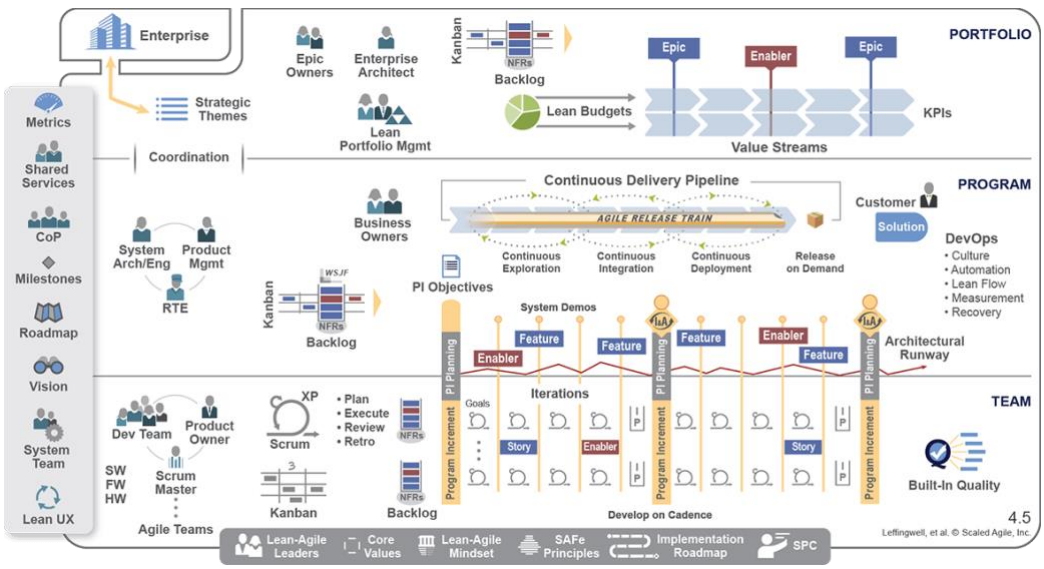

Figure 15.The three- level SAFe configuration (Knaster & Leffingwell, 2017).

- *The Four-level SAFe configuration*, it supports enterprises that build and maintain large integrated solutions, which require hundreds of people or more, and includes all levels of SAFe: team, program, large solution, and portfolio. It can be interpreted in Figure 16, that this configuration is dedicated to deploy Lean-Agile thinking in the entire organization. The coordination is done by using agile method practices, and economic aspects are considered (Economic Framework).

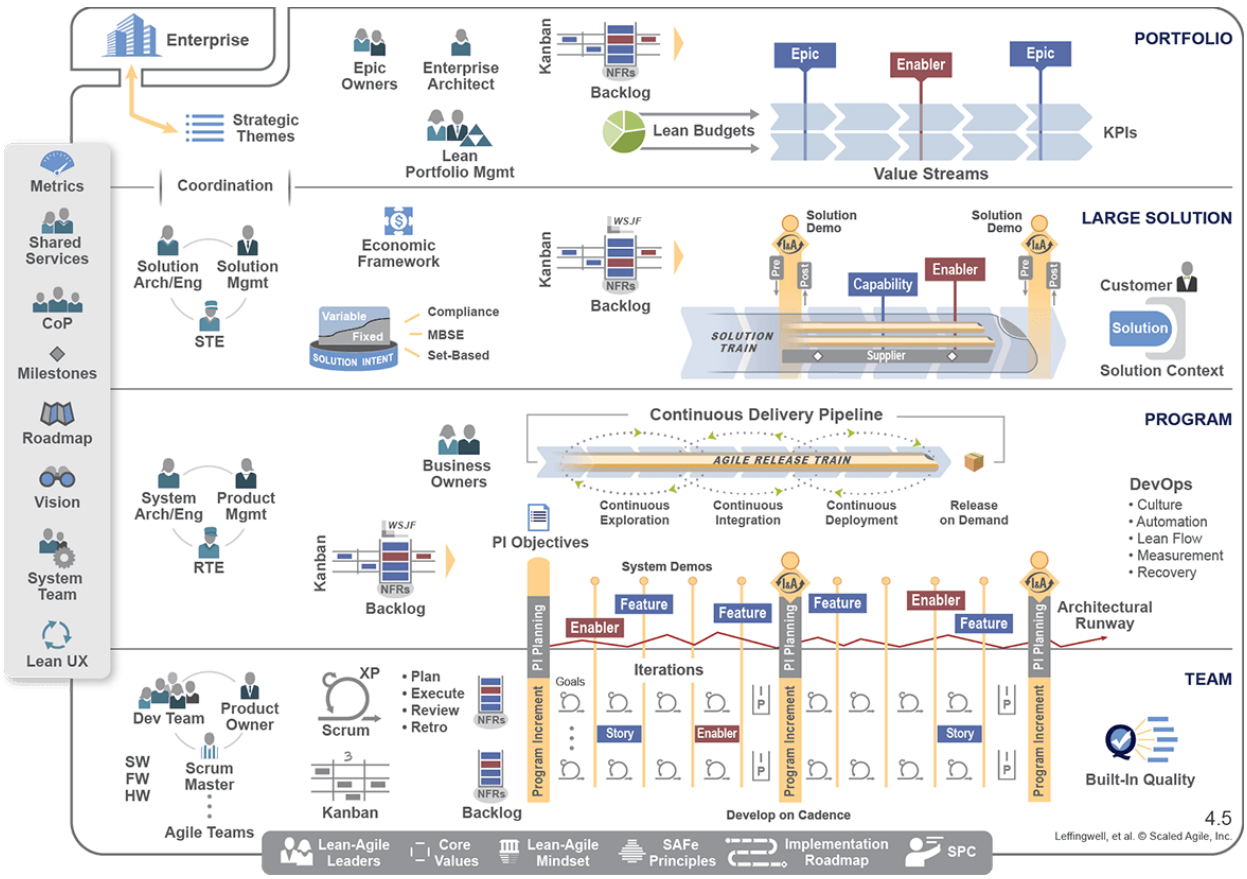

Figure 16.The four-level SAFe configuration (Knaster & Leffingwell, 2017).

In the last version (V4.5), the framework proposes four different configurations: essential SAFe, portfolio SAFe, large solution SAFe, and Full SAFe. This version also proposes a Lean-Agile mindset, and allows companies, to test ideas more quickly, deliver much faster and simplify governance and improve portfolio performance.

This framework considers software and hardware integration at the same time, that allows companies to develop their products in a Lean- Agile way. The different configurations allow companies to work with complex projects. SAFe integrates the

thinking lean, embraces agility, and systems thinking, the combination of these three disciplines can help companies to face challenges, and to achieve more effective large-scale solution development.

II.4. Comparison of Agile Methods

II.4.1 Understanding the differences between Agile Methods

This section introduces different comparisons of two bodies of knowledge presented in the literature review: agile and lean development. First it presents which approaches deal with projects rather than just the development and product delivery. This section also shows what kind of projects are managed for each agile approach.

	Project Level		**Product Level**	
	Software	**Engineering**	**Software**	**Engineering**
Agile Methods				
Scrum			x	
XP	x		x	
Kanban		x		
Crystal	x		x	
DSDM	x	x		
FDD			x	
Lean			x	x
SAFe	x	x	x	x

Table 4. Areas of activity of agile approaches

For example, DSDM is often used to manage the full project and Scrum is focused in product development process. Kanban and DSDM approaches are used to manage engineering projects. Lean approach is mainly used to develop engineering products; we can find in literature that author adopted the lean principles and applied them to develop software products, including the management of a project. SAFe appears as a mixture of Lean and Agile ideas, this framework integrates the lean thinking principles

and the agile principles defined by the Agile Manifesto. SAFe brings to the customers the opportunity to develop products and manage projects (both, software and systems) in a Lean-Agile philosophy.

In order to better identify the differences between the agile methods presented above, we proposed to identify similarities and differences between them. Agile approaches shared a core of characteristics, these characteristics are (Meyer, 2014):

- Values: general assumptions to understand the agile idea,
- Principles: core of rules to be follow in an agile environment,
- Roles: responsibilities and privileges of various actors in an agile environment,
- Practices: specific activities practiced by agile teams, and
- Artifacts: tools that support the practices.

Table 5 summarizes the presence of the core of characteristics in agile approaches.

Approach	Values	Principles	Roles	Practices	Artifacts
Scrum	5 default values	No defined	3 entities	5 called events	Defined
XP	5 default values	No defined	4 entities	12 primary practices	No defined
Kanban	9 default values	2 principles	2 entities	6 practices	No defined
Crystal	6 default values	7 principles	No defined	No defined	No defined
DSDM	No defined	8 principles	13 entities	No defined	No defined
FDD	No defined	No defined	6 entities	8 practices	No defined
Lean	One value	6 principles	No defined	No defined	No defined
SAFe	4 core values	8 values	8 entities	3 levels of practices	No defined

Table 5. Core characteristics of the agile approaches

This analysis shows that values were defined in four agile approaches of the six described above, Lean and SAFe also define values. Values are not similar, their definition depends of the vision of each method, however, they share some characteristic as continuous feedback, good team communication and collaboration. Principles are only defined in three agile approaches, these three approaches are globally agreeing on communicate continuously and clearly, focus in customer needs and frequent delivery. Roles were defined in almost all approaches, only Crystal and Lean do not propose a specific role definition. Practices were defined in four agile approaches; a common characteristic is incremental and iterative development. The meaning of Artifacts is only well explained in scrum approach.

Projects face conflicting demands, and the four most common demands are: time, cost, features and quality. In the traditional approach to managing a project, the features content of the solution is fixed while time and cost are subject to variation. Kruchten identified eight key factors in project level context (the octopus model) in order to contextualize agile software development, these factors are (Kruchten, 2013):

- *Size:* the overall size of a system is the main factor, it influences team size
- *Stable architecture:* can the architecture could be frozen at the beginning of the project? Most projects follow commonly accepted patterns in their respective domain
- *Business model:* this refers to the type of business model, it could be an internal system, commercial product, open source, etc.;
- *Team distribution:* Linked often to the size of the project, it refers to the number of teams involved in the project. Collocation of team members is important in a project,
- *Rate of change:* this refers to the requirements evolution. There are still projects with very stable requirement definitions but there are others that are not well defined,
- *Age of system:* this refers to the type of system that is being developed (new system or an evolution of old system),
- *Criticality:* this considers the risk of a system failure,

- *Governance:* this refers to the way the project is governed, it covers who manage the software project managers, who decides what happens when things go wrong, and, the definition of success or failure in the project.

Kruchten (Kruchten, 2013) also defines the organizational level (environmental factors) in order to contextualize agile software development. In these level five factors are listed, business domain, number of instances, maturity of organization, level of innovation, and culture. Organizational factors impact the octopus model.

Figure 17 presents the relation between organizational level and project level context. Business domain (see Figure 17) impacts four project factors (size of system, critically, rate of change, business model), that means that the domain of activity of the organization constrains these projects factors in order to identify if software is the primary business activity of the organization.

The number of instances impacts the governance of the project, in that case, the number of instances where the system is deployed will condition the way of how the project will be managed.

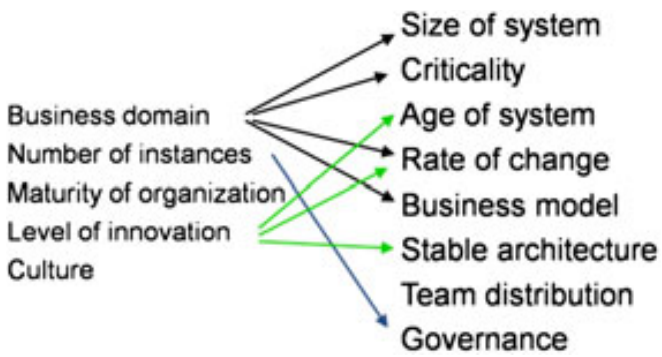

Figure 17. Relationship between organization level and project level in software projects (Kruchten, 2013)

Table 6 point out the presence of the factors, defined by Kruchten (Kruchten, 2013), in Agile Methods, Lean overview and SAFe overview presented in literature review.

	Factors in the Project Level for agile software development							
	Size	Stable Architecture	Business Model	Team distribution	Rate of change	Age of system	Criticality	Governance
Agile Methods								
Scrum	Small teams	yes	product	Well defined	Medium to high	New system	low	Well defined
XP	Small teams	yes	product	Well defined	Medium to high	New system	low	Well defined
Kanban	Small teams	yes	product	Well defined	Medium to high	New system	low	Well defined
Crystal	Small, medium and big teams	yes	project	unproposed	Medium to high	New system	low	Well defined
DSDM	Medium and big teams	no	project	Well defined	Medium to high	New system	high	Not defined
FDD	Small teams	yes	product	Well defined	Medium to high	New system	low	Well defined
Lean								
Lean	Medium and big teams	yes	Project and product	unproposed	Medium to high	New system	high	Well defined
SAFe (Lean-Agile mindset)								
SAFe	Medium and big teams	no	Project and product	Well defined	Medium to high	New system	high	Well defined

Table 6. Presence of Agile and Lean in Project level

The analysis presented in Table 6 centralizes how Agile methods, Lean and SAFe are involved in the project factors proposed by Kruchten. Agile methods are mainly used in small and medium size projects, Lean and SAFe in medium and large size projects. Agile, Lean and SAFe are used to develop an internal system or commercial product. Team distribution and governance are well defined in almost all methods.

II.4.2 Understanding the differences between Agile and Lean

There are a number of similarities that Agile and Lean share, the concept of added value, the definition of principles, more performing teams, flexibility, faster delivery of products, are some of these similarities. Beyond the definitions given in later sections,

it is important to understand the differences between these two concepts. It can be found in literature, that Lean is well used in software development, but it is more commonly used in environments of predictable demand, and high volume. On the other hand, Agile is most commonly used in unpredictable environments, that means that Agile helps stakeholders respond quickly to change.

Another important aspect is that talking about Agile means talking about a whole set of frameworks, practices and methods, that share a set of common values and principles (set out in the Agile Manifesto). On the contrary, talking about Lean means talking about having industrial production more efficient, where the most important is to build a system having the whole vision of it, that helps understand the system and optimize it. Lean and Agile are two different but complementary concepts (Oehmen et al., 2012). Table 7 presents a comparison between Agile and Lean approaches in order to see more clearly the differences between them.

	Lean	***Agile***	***Comment***
Values	• It represents the customer needs during a system life cycle.	• Individuals and interactions • Working software • Customer collaboration • Responding to change	The value is not quite identical, but it could be linked: Lean focalizes value in customer needs, and Agile integrates customer collaboration, that means that customer opinion and needs are considered while using Agile.
Principles	1. Capture the value defined by the customer. 2. *Value Stream,* map all end-to-end linked task, control/decision nodes, and interconnecting flows. 3. Flow the work through planned and streamlined value-adding steps and processes. 4. Let customer stakeholder pull value.	1. Highest priority is customer satisfaction. 2. Welcome changing requirements. 3. Frequent delivery software. 4. Business people and developers work together. 5. Build projects around motivated individuals. 6. Face-to-face conversation. 7. Progress measured by working software.	The number of principles varies from one approach to another, Lean proposes six principles and Agile Twelve. Principles could be also linked, people in the organization are important, this idea is well impregnated in value number 6 of Lean, and value number 5 of Agile. Even if the principles of the two approaches are focused in optimize and execute the process, Agile principles are about scope

	5. Pursue perfection in all processes. 6. Respect the people in your program.	8. Constant peace over stainable development. 9. Continuous attention to technical excellence. 10. Simplicity is essential. 11. Self-organizing teams. 12. Regular reflection and adaptation	(reflects in principle 9) and value discovery (reflected in principles 2,3,4,6), while Lean principles are about process improvement (reflected in principle 2, and 3), and quality (reflected in principle 5).

Table 7. Comparing Agile and Lean approaches.

Some interpretations could be made from Table 7: Lean principles are most useful while having the perspective of the whole, that means that the definition of customer needs and requirements should be well done from the beginning (contrary to Agile principles). Lean optimizes processes and organization to deliver value, Agile stresses responsiveness to changing customer requirements. Lean does not forbid changing customer requirements, and Agile does not absolve an organization that does not understand customer value properly (Oehmen et al., 2012).

II.5. Conclusions

II.5.1 Interest of agility and deployment challenges

This chapter presents the literature review around the bodies of knowledge that we covered in this thesis. First, we introduce the agile overview, starting with the definition of the agile term, we conclude that agility should be considered as a team's competence in the project environment, this competence has to consider different aspects as: culture, organization structure, management practices, business environment, etc.; the story of agility was introduced to know, when the first agile practices were established, and how the Agile Manifesto was defined. The Agile Manifesto covers better ways to develop software, it lists four values and twelve principles. The principles are guidelines to deliver high-quality software in an agile manner. This work only lists the most commonly used methods currently used, each method proposes principles, values, practices, roles and artifacts, and they capture the

notion of the Manifesto for Agile software development, these methods have become very popular in the last 15 years. The Chaos report (Hastie & Wojewoda, 2015) showed that projects are more successful while using agile methods, but this report only emphasize the management of software projects. Agile methods are deployed in software context, some to manage complex projects (DSDM, Crystal), and some to manage the product process (FDD, XP). Agility involves the ability to adapt to different changes, it is a state of mind, where the creation of value for clients is one of the most important goal. The need for more holistic methods, allowed the definition of agility in operations, this idea brought together, both development and operations (DevOps). DevOps sprang from applying Agile and Lean thinking to operations work. This method involves the collaboration between development and operations staff throughout all stages of development lifecycle (Mueller, 2017).

The second part, of this literature review, introduces Lean thinking. Lean was implemented in manufacturing process first, the creation of value without waste is the general objective of this approach. Lean defines a set of principles to be follow for create value. The term “waste” is well defined, and it covers value-add activities, necessary non-value-added activities and non-value-added activities. Currently, Lean is used in different contexts, not only manufacturing.

Another section of this research considers a scaled agile framework (SAFe). This framework brings together, agile development, systems thinking, and lean product development. SAFe allows the development of complex systems and software in a Lean-Agile way. Its core of values, and principles, covered Lean and Agile ideas. The implementation of SAFe can be done using three different configurations: essential configuration (the most basic configuration), three-level configuration (where the number of teams are between five and ten), and four-level configuration (considers large integrated solutions, and a lot of people are involved).

Based in the comparisons made in this chapter, we conclude that agile methods can be used to manage software projects, and product development process. Each method covers different contexts and it depends on their core of values and principles. We found that there are other methods, as Kanban, to manage engineering projects. Lean is mainly used to develop engineering products, but, its ability to adapt, allows its use

in software context. This analysis opens up different questions: Can agile methods be used in the word of systems projects? Is agility present in the engineering standards? In the engineering practices, is there a notion of agility?

This panel of questions will be discussed in the next chapter, in order to explore if agile ideas and methods can be used in systems engineering.

II.5.2 Problem Analysis

The number of projects designed and developed around the world is immense, and their success is one of the most important attributes. Project performance can be an important factor to ensure the quality and success of the project, in a highly competitive economic environment, companies are facing changes in the way they are managing their projects, they are interested to improve project duration, project costs, project performance and to apply efficient practices to manage their engineering projects. The challenge to achieve that is to simplify and speed up the implementation of the process in order to better control, coordinate and manage projects, it is also important to consider customer satisfaction and to help engineers and managers to better manage and lead the project. The adoption of more decentralized approaches, collaborative and self-managed project development, performance indicators of the project, good integration of all stakeholders, good team motivation in the project could ensure the success of the project. Currently there are methodologies that have proven to be a good tool for successful projects, but they are not used in systems engineering projects, agile methods are one of them, these methods are mainly used in companies whose business is software development. Fields as systems engineering are contemplating these methods in order to see if they can lead projects in complex systems development.

III. Agility in Systems Engineering Projects.

This chapter brings together Agility and Systems Engineering. First, section III.1. introduces the difference between Agile - Systems Engineering (A-SE) and Agile Systems – Engineering (AS-E). Section III.2 gives an overview of different works that consider the question of agility in systems engineering. Section III.2. looks for any mention of agility in systems engineering standards and analyzes their ability to be compliant with agility. Section III.4. introduces the research methodology we followed for deploying agility in systems engineering projects. A contextual model for systems engineering development and the deployment of Scrum in engineering projects will be proposed. The conclusion of this chapter is drawn in section III.5.

III.1. Agile – SYSTEMS ENGINEERING and AGILE SYSTEMS – Engineering

Two different mentions to the word ‘agile’ can be found in literature in systems engineering, both called ‘Agile Systems Engineering’. In the first case the system of interest is an engineering process, and in the second case the system of interest is what is produced by an engineering process (Dove & LaBarge, 2014). Haberfellner and de Weck state that Agile - Systems Engineering focuses on flexibility and speed in the upstream process of conceiving, designing, and implementing products and systems; while Agile Systems – Engineering puts the emphasis on embedding agility in the systems themselves, that means that the system is flexible and has the ability to change from one state or operating condition to another rapidly, without large switching costs or increases in systems complexity (Haberfellner & de Weck, 2005). The term *flexibility* appears in both cases. Flexibility is a property of a system that can be changed easily, while agility is a property of a system that can be changed rapidly (Fricke, Gebhard, Negele, & Igenbergs, 2000). Considering these two concepts, it is possible to conclude that:

- Agile - Systems Engineering (A-SE) focuses in the rapid change of convincing, designing, and implementing processes of products and systems in an easy way, and

- Agile Systems – Engineering (AS-E) focuses in the ability of the systems to rapidly change from one state (or operating condition) to another in an easy way.

Some characteristics to differentiate the two terms were established by Haberfellner and de Weck. Table 8 lists some characteristics of the agile product development process and agile systems. It can be interpreted, from this table, that, for A-SE, agility is deployed in the systems engineering process to develop a product/system, and for AS-E agility, is installed within the system, that means the ability of the system to change after its initial fielding.

Agile product development process (A-SE)	**Agile systems (AS-E)**
A generic agile product development process can be: • Adaptive and response to new (sometimes unexpected) information that appears during the product/system development, • Opposite of the traditional belief, in engineering design, where requirements and design solutions should be frozen as early as possible, • Nimble, dexterous and swift.	Agile systems can be: • Flexible in terms of functions and performance levels. Systems can be modified after initial deployment by addition of modules or modification of performance level, • Scalable in the sense of capacity. Systems that can rapidly adapt to actual market demand, • Reconfigurable and extensible.

Table 8. Characteristics of Agile - Systems Engineering and Agile Systems - Engineering (Haberfellner & de Weck, 2005)

Ambiguities in customer requirements, the viability of new technologies or the appropriateness of one manufacturing process over another, can cause situations where there are significant uncertainties during product/systems development and manufacturing. Usually there is the expectation that these uncertainties can be resolved before the product or systems is shipped (Haberfellner & de Weck, 2005).
For A-SE, agility is considered as the ability of a system to thrive in an uncertain and unpredictably evolving environment, deploying effective response to both opportunity and threat, within mission. Effective response has four metrics: timely (fast enough to deliver value), affordable (at a cost that can be repeated as often as necessary), predictable (it can be counted on to meet the need) and comprehensive (anything and

everything within the system mission boundary) (Alain ROUSSEL, 2016). For AS-E the system of interest is the process of engineering systems, in this context, system agility is generally warranted when the ability to predict future demand or functional requirements is severely compromised, and the focal point of agility in practice to date has been on process innovation rather than product innovation. Agile Systems are designed for proactive and reactive adaptation to evolving needs and opportunities when these are unpredictable, uncertain, and likely to change (Dove & LaBarge, 2014).

Mature industries that focus on process innovation rather than product innovation might be most amenable to A-SE (Haberfellner & de Weck, 2005). For this reason, this work will take particular attention to Agile - Systems Engineering (A-SE) concepts.

III.2. Exploring the question of introducing agility in Systems Engineering

The AFIS organization (Association Française d'Ingénierie Système) is representing the International Council of Systems Engineering (INCOSE) in France. This association was founded in 1998 by thirteen major industrial groups. Several AFIS members (from whom members of the team ISI of the LAAS-CNRS) contribute to the Agile Systems Engineering working group of AFIS. This group proceeded to an analysis of the volumes 17 (Composable capability for Agile SoS) and 18 (Approaches and Benefits for Adopting Agile Methods) of the INSIGHT INCOSE journal, dedicated to the issues associated with Agile Systems Engineering. The result of this analysis summarizes the contents of volumes 17 and 18, which include (Alain ROUSSEL, 2016):

- The criteria that lead agile method's adoption.
- The criteria that characterizes an agile architecture and evaluation criteria.
- The principles that allows the systems development in agile way.
- The resources that allows the measure of agility in system engineering context.

AFIS 's analysis proposes to paid attention in those key subjects (Alain ROUSSEL, 2016):

- *Added value:* means that agile approach must maximize the added value to the users of the system and control its deployment. This added value is taken into

from the notion of “valuable capability” [Volume 18, principles for agile development] and that concept is closer to the operational capacity on systems engineering.

- *Enjoyable (être ludique):* the concepts, tools and supports implemented in agile approach should be enjoyable, the main idea of this concept is to focus on team interaction ambiance.
- *Terminology:* the sprint term is rather replaced by the notion of iteration for the development of a system. Indeed, the usual time unit of a sprint (a few weeks) is generally not suited to the development of a system.

These key subjects are strongly related with the values and principles of Agile Manifesto. Terminology is very important when we want to develop systems using agile practices, values and principles. The terminology used in software development could not be transposable to Systems Engineering; for instance, some interlocutors cannot agree to replace the term ‘sprint’ term by the term ‘iterative’. AFIS analysis also discusses why it is important to implement an agile approach in systems engineering; they highlight some attributes for the adoption of an agile method in engineering projects. Cohn (Cohn, 2010) also highlights some challenges that organizations face by adopting agile methods. He identified certain attributes of transitioning a company process to an agile method that make it more difficult than most other changes. The attributes were listed as follows:

- Successful change is not entirely top-down or bottom-up structure,
- The end state is unpredictable,
- Agile methods are pervasive,
- Agile methods are dramatically different to the traditional systems development (waterfall model for example),
- Change is coming more quickly than ever before,
- Best practices are dangerous.

Successful change is not entirely Top-Down or Bottom-Up, it means that change in the organizational level cannot be fully top-down or bottom-up, the top-down change leaders share a vision of the future and the organization follows the leader towards that vision. In another hand, bottom-up change the team or some individuals decide

that a change is needed, and they set about making it happen. Mike Cohn states that the key to any successful adoption of an agile method will be combining elements of both bottom-up and top-down change (Cohn, 2010).

The End State is Unpredictable, having a chance to change or personalize a process to fit themselves, seems to be critical success factor for a team to adopt a process (Cockburn, 2017), it is not sufficient adopt an agile method thinking that it is the best for the organization, instead, we will need to tailor the process to more precisely fit the unique circumstances of an organization.

Agile methods are pervasive, being agile will have implications to the organization that reach far outside the software development department.

Agile methods are different, using an agile method involves asking people to work in ways that are unfamiliar and run counter to training and experience, change scares people and people are often hesitant or resistant to change (Cohn, 2010)

Other aspects (like the problems that companies face when they want to perform their engineering projects, using an agile method) are considered in the exploration of related works to this subject. To know these problems, a meeting was held with Mr. David Brocard, who is working as an agile consultant for French companies. He mentioned that agility should be considered as a continuous improvement of process, this vision is totally in accordance with the characteristics of A-SE (presented in section III.1.). Brocard proposes four agility pillars: value, adaptation, team and feedback.

Figure 18 shows the list of activities for each pillar. The activities of each pillar focus on the global business value and their success depends on the notion of renunciation. Most of the agile initiatives which end as a mixed result or a bad implementation have often not respected this principle of renunciation. In three words, to renounce means: “not to do”, it means that people simply have to give up certain attitudes or practices that they have been using for many years. That idea could be considered as one of the most difficult part when companies apply agility. In other words, renunciation assumes simplicity, concretely, on a daily basis through non-action.

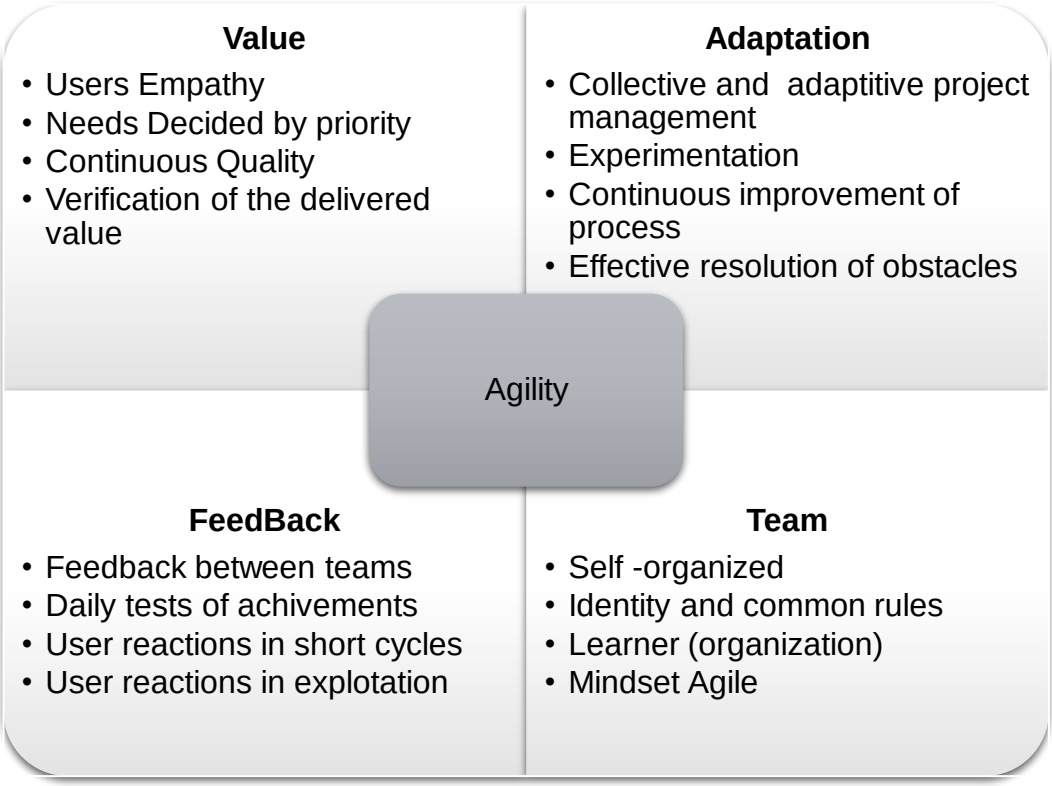

Figure 18. Pillars of agility (Brocard, 2017)

Being agile does not mean adding a new process layer to the existing one, the multiplication of deliverables and the work overload can then become counter-productive and generate demotivation and many doubts about the interest of the approach. Unfortunately, it is difficult to be able or willing to give up some of the usual practices that affect the benefits of implementing Agility (Brocard, 2017).

As an example of renunciation, Brocard proposes to negotiate with the quality department that the project benefits came from a deviation in the usual rules of process and deliverables, another example of renunciation is when companies want to deploy an agile tool (it could be a framework) in all products development, they should delete all current roles, meetings and deliverables that do not fit on this framework.

In conclusion to this overview, the different entities or consultants that were questioned by introducing agility in SE agree that value is an important characteristic of agility. Faster time-to-market, high productivity and quality, low costs, stakeholder satisfaction, improved employee engagement and job satisfaction, are reasons to

adopt an agile approach in systems engineering. So why agility is not deployed in systems engineering? Why systems engineering standards do not recommend using agility? The following sub-section analyses systems engineering standards to identify if we can find any notion of agility in them and if they could be compatible with agile practices.

III.3. Looking for agility in Systems Engineering Standards

Different Systems Engineering (SE) standards and guides have been defined in different fields of application, such as aeronautics military, automatic, and management. Organizations, such as ANSI/EIA 632, ISO/IEC, IEEE 1220, INCOSE, and SEBoK, have then proposed their own SE standards, each one providing a different implicit systems life cycle, level of detail, and scope. This sub-section first presents the major SE standards currently in use, then relies on previous works on SE management to select the standard where we could have the greatest chance to find explicit or underlying mention to a certain kind of agility. It finally analyses this standard to conclude on the presence of any kind of agility in it and on its ability to be compliant with agile practices.

III.3.1 Commonly used Systems Engineering standards and guides

Many SE standards and guides have been elaborated over the years. Figure 19 shows the timeline and relationships between SE standards and guides. MIL-STD-499 was the first SE standard developed in 1969, an update version 499A was established in 1974 and one-second version 499B was delivered in 1994. MIL-STD 499B standard split into EIA/IS-632 (developed by the Electronic Industries Alliance and the International Council on System Engineering (INCOSE)) and IEEE-1220 (developed by the Institute of Electrical and Electronic Engineers (IEEE) society). IEEE-1220 focuses in the development and management of systems engineering processes. As shown in Figure 19, two versions of IEEE-1220 standard were delivered between 1994 and 2005, in that case the IEEE-1220:2005 is the current version proposed by IEEE. On the other hand, EIA/IS-632 evolved into ANSI/EIA-632 (making this the current version) in 1998, and ISO/IEC-15288 in 2002. The International Organization

published ISO/IEC-15288 for Standardization (ISO); it focuses on processes for engineering a system through its full life cycle. The current version of ISO/IEC-15288 was published in 2015 (Xue, 2016).

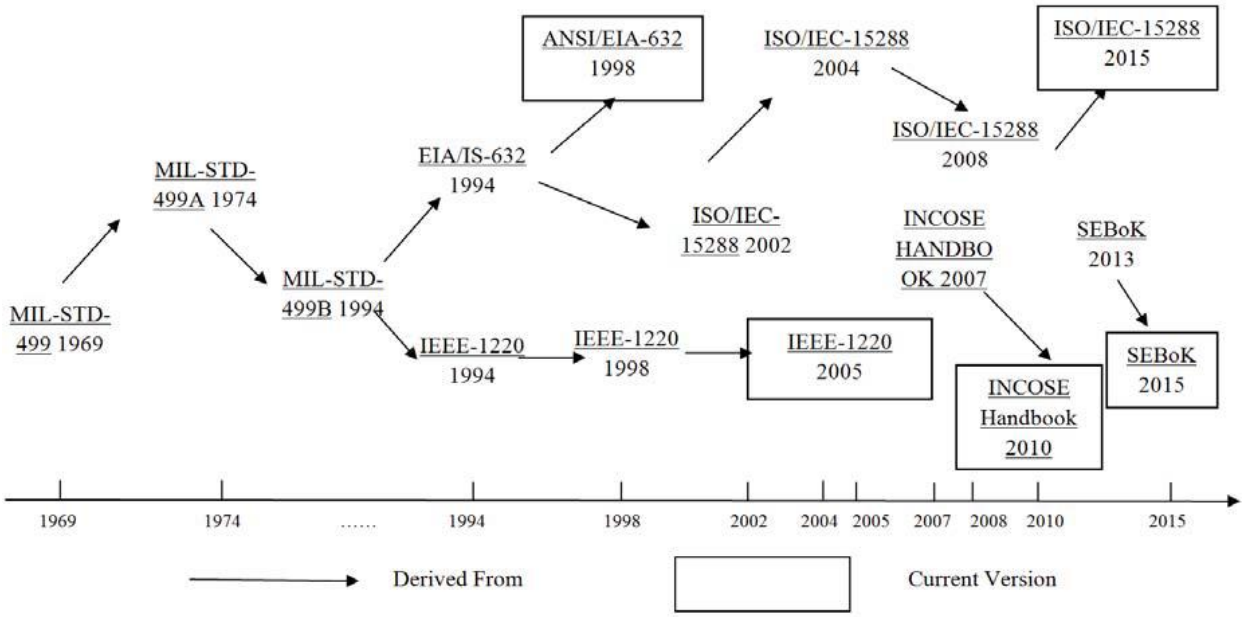

Figure 19. Timeline of SE standards and guides (Xue, 2016).

In 2007, the INCOSE published the SE Handbook; it covers the system's life cycle, processes and activities; the latest version dates back to 2010. In 2013 several experts wrote the SEBoK, which currently is the most detailed SE guide. There are different standards and guides for Systems Engineering, focusing on different aspects, the definition and the development of a system, or the management of its life cycle. Each standard or guide proposes a group of process to be followed in order to develop a system.

III.3.2 Comparison of Systems Engineering standards and guides

Sharon et al discussed similarities and differences among SE standards to help individuals and organizations understand them, in order to choose the one that is most adaptable to their needs (Sharon, de Weck, & Dori, 2011). The criteria used to proceed to a comparison were:

- Scope of standard,
- Level of abstraction,
- System life cycle, and,

- Systems Engineering Management plan guidance.

At that time, The INCOSE SE Handbook and SEBoK were not considered. Rui Xue proposed an extended analysis integrating the latest versions of ANSI/EIA 632, IEEE 12120, ISO/IEC 15288 (including the Handbook and the SEBoK). Nine criteria were defined to do the comparison.

- Content, it describes the number of process defined by the reference and how they are pooled together,
- Focus on system life cycle, it covers the stages of the system life cycle in each standard,
- Number of pages,
- Level of details, it refers how detailed the standard is,
- Context of application, it covers the environment where the standard is used (program, enterprise and external environment),
- Year of publication,
- Revision frequency,
- Number of management processes, it considers how many processes are included to the management,
- System Engineering Management (SEM) – process ratio, the proportion of SEM-Processes with respect to their number can be found in each reference.

Table 9 compares the most used SE standards and guides, based on the criteria defined by (Xue, 2016). According to them, the ISO/IEC 15288 appears to be the most interesting to analyze to look for agility:

- The number of process are reasonable in comparison to the IEEE 1220 standard; on another hand ISO/IEC 15288 competes with the HANDBOOK and the SEBoK, they propose the same number of process,
- It covers the entire system life cycle, in that case ISO/IEC 15288 only differs with ANSI/EIA,
- ISO/IEC 15288 is up to date,
- ISO/IEC 15288 has the most processes related to project management.

	ANSI/EIA 632	ISO/IEC 15288	IEEE 1220	INCOSE HANDBOOK	SEBoK
Content	13 processes 34 requirements	25 processes	8 processes	25 processes	26 processes
Focus on system life cycle	Conception and development	Systems' entire life cycle	Systems' entire life cycle	Systems' entire life cycle	Systems' entire life cycle
Number of pages	110	70	70	400	850
Level of details	++	++	++	++++	+++++
Context of application	Program and project environment	Enterprise environment	Program and project environment	Enterprise environment	External environment
Year of publication	1998	2015	2005	2010	2013
Revision frequency	++	+++++	++	+++	+
Number of SEMP	3	12	1	12	12
SEMP's proportion	3/13	12/25	1/14	12/25	12/26

Table 9. Comparison between the most use SE standards and guides (Xue, 2016)

As mentioned before A-SE puts emphasis on the SE process, and the ISO/IEC 15288 reference involves different processes to the development of a system and to the management of a project. The following sub-section will introduce agility in Systems Engineering, in order to identify, if the process of the ISO/IEC 15288 can be flexible in terms of rethinking and modifying solutions and concepts face to uncertainties during the systems development.

III.3.3 Is there any agility in the ISO/IEC 15288?

The ISO/IEC 15288 is a Systems Engineering international standard covering processes and life cycle stages. It establishes a common framework of process descriptions for describing the life cycle of man-made systems. It also provides processes that support the definition, control and improvement of the system life cycle processes used within an organization or a project (IEC/IEEE, 2015).

Figure 20 shows the systems life cycle processes proposed by the standard. The standard is broken down into four groups of processes: Agreement Processes, Organizational Project-Enabling Processes, Technical Management Processes and Technical Processes.

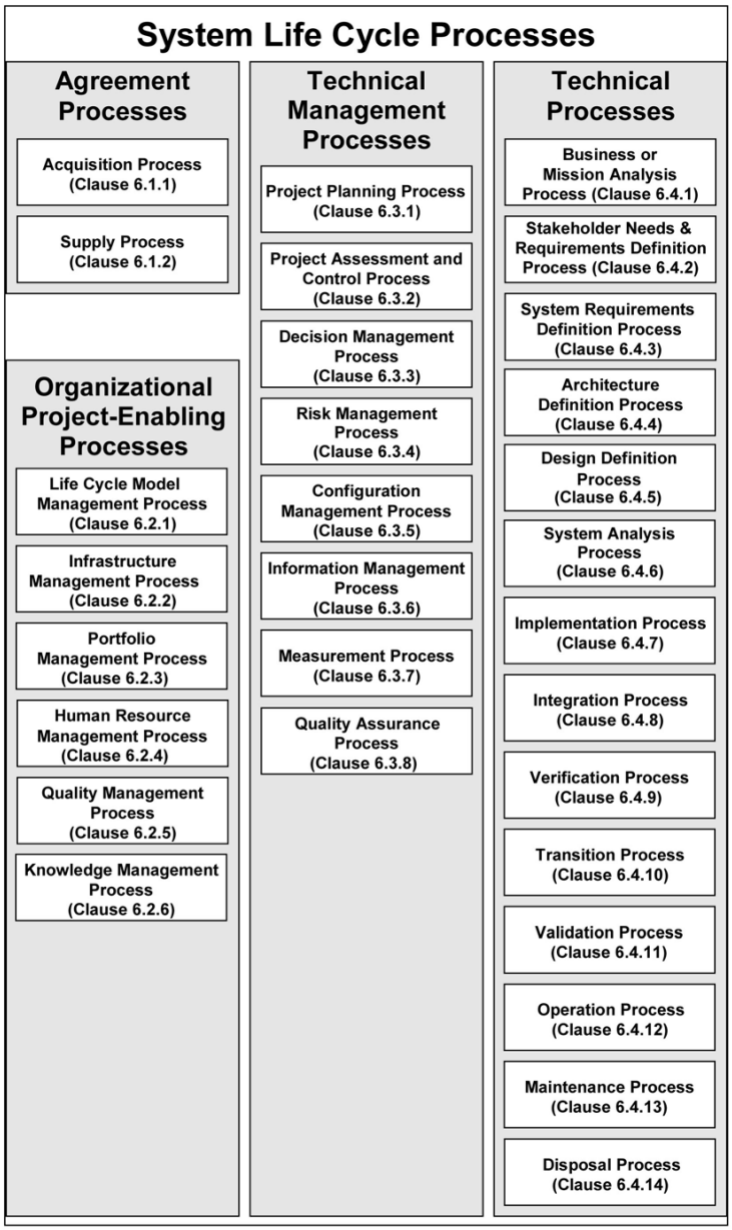

Figure 20. Systems life cycle processes (IEC/IEEE, 2015)

The group “Technical Management Processes” is concerned by managing the resources and assets allocated by the organization management; it relates to planning (cost, timescales, achievements), to the checking of actions and to the identification and selection of corrective actions if needed. Therefore, its scope covers part of project management scope, that also includes planning and controlling resources to achieve project goals (IEC/IEEE, 2015). Thus, if ever there was any notion of agility in the ISO/IEC 15288 standard it would be in the Technical Management Processes group. The analysis thus focuses on this process group.

III.3.3.a Analysis of ISO/IEC 15288 Technical Management Process.

This section analyzes the Technical Management Processes group of ISO/IEC 15288 to search any implicit or explicit mention of agility such as defined in section II.1.1. Technical Management Processes includes eight processes that are used to establish and perform the technical management of projects (IEC/IEEE, 2015). This group contains: Project Planning, Project Assessment and Control, Decision Management, Risk Management, Configuration Management, Information Management, Measurement Management, Quality Assurance. Processes are described with attributes: title, purpose (describes the goals of performing the process), outcomes (express the observable results expected from the successful performance of the process), activities (sets of cohesive tasks) and tasks (requirements, recommendations or actions intended to support the achievement of the outcomes) (IEC/IEEE, 2015).

To detect agility in the standard, the analysis must stand at the task level. The method thus consists in exhaustively analyzing the tasks related to the activities of the Technical Management Processes and to check with the 12 principles from the Agile Manifesto, presented in the literature review (section II.1.3), if any agility can be found.

Figure 21 shows the activities and their associated tasks of the project planning process. The Project planning process has 3 activities: Define the project, Plan project and Technical management, Activate the project. Each activity includes several tasks.

ACTIVITY 1 (A1). DEFINE THE PROJECT

- Task 1: Identify the project objectives and constraints.
- Task 2: Define the project scope as established in the agreement.
- Task 3: Define and maintain a life cycle model.
- Task 4: Establish a work breakdown structure.
- Task 5: Define and maintain the processes.

ACTIVITY 2 (A2). PLAN PROJECT AND TECHNICAL MANAGEMENT

- Task 1: Define and maintain a project schedule.
- Task 2: Define achievement criteria for the decision gates.
- Task 3: Define the costs and plan a budget.
- Task 4: Define roles, responsibilities, accountabilities, and authorities
- Task 5: Define the infrastructure and services required.
- Task 6: Plan the acquisition of materials and enabling system services
- Task 7: Generate and communicate a plan for project and technical management.

ACTIVITY 3 (A3). ACTIVATE THE PROJECT

- Task 1: Obtain authorization for the project.
- Task 2: Submit requests and obtain commitments for necessary resources.
- Task 3: Implement project plans.

Figure 21. Activities and tasks of the Project Planning Process (IEC/IEEE, 2015)

Figure 22 shows the results of the analysis for the project planning process. P_8-Promote sustainable development, and P_9-Technical excellence are the most referred principles among all the tasks from the project planning process; P_{11}-Self-organization teams is never referred in the project planning process.

ISO/IEC 15288 *Project Planning Process*		12 Principles of Agile Manifesto											
ACTIVITIES	**TASKS**	**P_1**	**P_2**	**P_3**	**P_4**	**P_5**	**P_6**	**P_7**	**P_8**	**P_9**	**P_{10}**	**P_{11}**	**P_{12}**
A1	*Task 1*	*			*				*	*			
	Task 2			*					*	*			
	Task 3	*								*			
	Task 4	*	*										
	Task 5	*			*				*				
A2	*Task 1*		*	*		*	*		*	*			*
	Task 2		*	*		*			*	*	*		*
	Task 3	*			*	*	*	*	*		*		*
	Task 4								*				
	Task 5					*				*	*		
	Task 6												
	Task 7			*			*						*
A3	*Task 1*	*											
	Task 2		*				*						*
	Task 3							*		*			

Figure 22. References to agile principles in the tasks related to the project planning processes

Following the same method, the analysis can be extended to the seven remaining Technical Management Processes. Figure 23 shows the results of the hole analysis of the Technical Management Processes. Agile principles are not present in Risk Management Process; P_4- Business teams and developers working together is one the most referred principle among all the process of technical management processes and P_{11} - Self-organization teams, is referenced only once. In proportion, the two main referred principles are P_4- Business teams and developers working together and P_6 - Face-to-face communication.

Process	Number of Activities	Number of Tasks	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Project Planning Process	3	15	40%	27%	27%	20%	27%	27%	13%	47%	47%	20%		33%
Project Assessment and Control	3	16	56%	38%	19%	75%	38%	38%		31%	75%	6%	13%	56%
Decision Management	3	10	20%	60%		60%	10%	70%			30%	10%		
Risk Management	5	16												
Configuration Management	6	21		38%	5%	57%	5%	48%	5%	14%	10%			14%
Information Management	2	10				70%	100%							
Measurement	2	11				36%		45%						
Quality Assurance	5	17		6%	6%			6%		6%				12%
	Total of presence.		3	5	4	6	5	6	2	4	4	3	1	4

Figure 23. References to agile principles in Technical Management Processes

The analysis proposes a method to evaluate if systems engineering standards such as the ISO/IEC 15288 could be compliant with any principles of agility such as defined by the Agile Manifesto. As mentioned before (in section III.1.) this thesis work is focused in the flexibility and speed that the engineering process can have while developing a product or system. The principles defined in the Agile Manifesto were the input to identify if there a notion of agility in SE process. The results of this analysis show that some of the technical management processes could be somehow aligned with the agile principles. This analysis also makes a first contribution to discuss agility in systems engineering. Indeed, introducing agility in systems engineering could help in reducing development cycles and ensuring control of the system.

However, many issues remain. The first one is to clearly precise What really could be the benefits from introducing agility in systems engineering? At what level? What really

are the issues for companies? Are they technical or organizational ones? Could agility be introduced in the development of software parts of a complex systems while other parts remain developed with a more traditional approach? These set of questions will be addressed in the following section. Where an approach is proposed to better deploy an agile method in systems engineering projects.

III.4. Integrating agility in systems engineering projects.

This section presents the research methodology we followed to consider how we could integrate agility in systems engineering development. Section III.4.1 first introduces the methodology, which consists in four steps. The following sections detail each of the steps. Step one is introduced in section III.4.2; it aims at defining the attributes that will contextualize systems engineering development. Section III.4.3 introduces the step two, that aims at analyzing the currently popular agile methods and identifying which one is the most adaptable for the attributes of systems engineering development. Step three (section III.4.4) studies how to deploy this method in systems engineering projects. Section III.4.5 develops the last step of the research methodology, that deals with the difficulties related to this deployment.

III.4.1 Research Methodology

The research methodology breaks down in four steps. Figure 24 shows the main goals of each step of the research methodology.

Using a contextual model for agile software development, the step one, *"Identifying the characteristics for systems engineering development"*, aims to define the factors and project attributes for systems engineering development. The result of this step is a list of factors and attributes that allow engineering projects to be contextualized.

The step two, *"Analyzing Agile Methods",* aims to determine which agile method is the most adaptable according to the factors and project attributes for systems engineering development. The result of this step is the selection of an agile method to be used for systems engineering development.

Then, the step three, *"Analyzing the Scrum Framework",* aims to deeply analyze and understand Scrum Framework for the management of engineering projects. The result of this step is a graphical view of the Scrum Framework, and the main practices to execute it.

Finally, the step 4, "*Selecting the difficulties to be analyzed*", aims to identify the role of the Scrum Framework, which will face most difficulties. Some alternatives will be listed to try to solve the set of the selected difficulties.

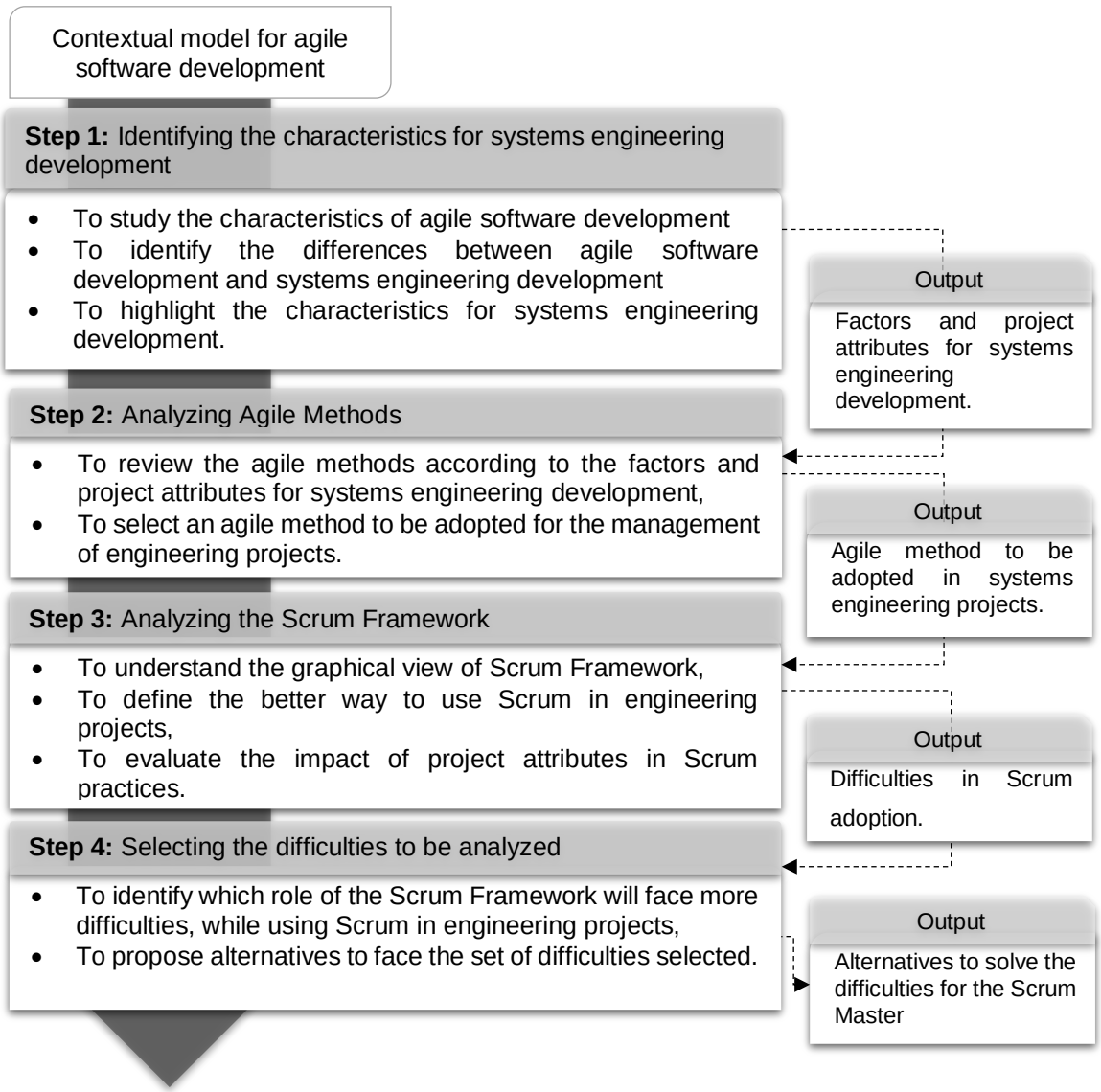

Figure 24. The Research Methodology

This research methodology was defined considering the following aspects:

- A variety of studies show that agile methods can generate a significant improvement over the traditional predecessor methods (V-Model for example) (Repenning, Kieffer, & Repenning, 2017; Stettina & Hörz, 2015),
- Applying agility in managing engineering programs could be operationalized by using agile metrics to evaluate responses, and an agile architecture to make the program and engineering system resilient to requirements uncertainty and change (Oehmen et al., 2012),
- The key for transferring a set of practices from one domain to another is to first understand why they work and then modify those practices in ways that both match the new context and preserve the underlying principles (Repenning et al., 2017).

III.4.2 Step One: Identifying the characteristics for systems engineering development

This step aims to define a contextual model for systems engineering development. Figure 25 details the list of activities of this step. First, starting from the contextual model for agile software development (proposed by Philippe Kruchten (Kruchten, 2013)), the characteristics of agile software development are first identified then interpreted from the organizational factors and project context level attributes proposed in Kruchten's contextual model. Then, differences between software projects and engineering projects are established, taking into account two aspects: the organizational and project attributes, and different engineering projects found in the literature. An examination of different systems life cycles is made to identify and list others organizational factors and project attributes that are not preset in Kruchten proposal but can be part of system engineering development. Finally, a contextual model for systems engineering development is proposed, that consists in five organizational factors and eleven project context attributes.

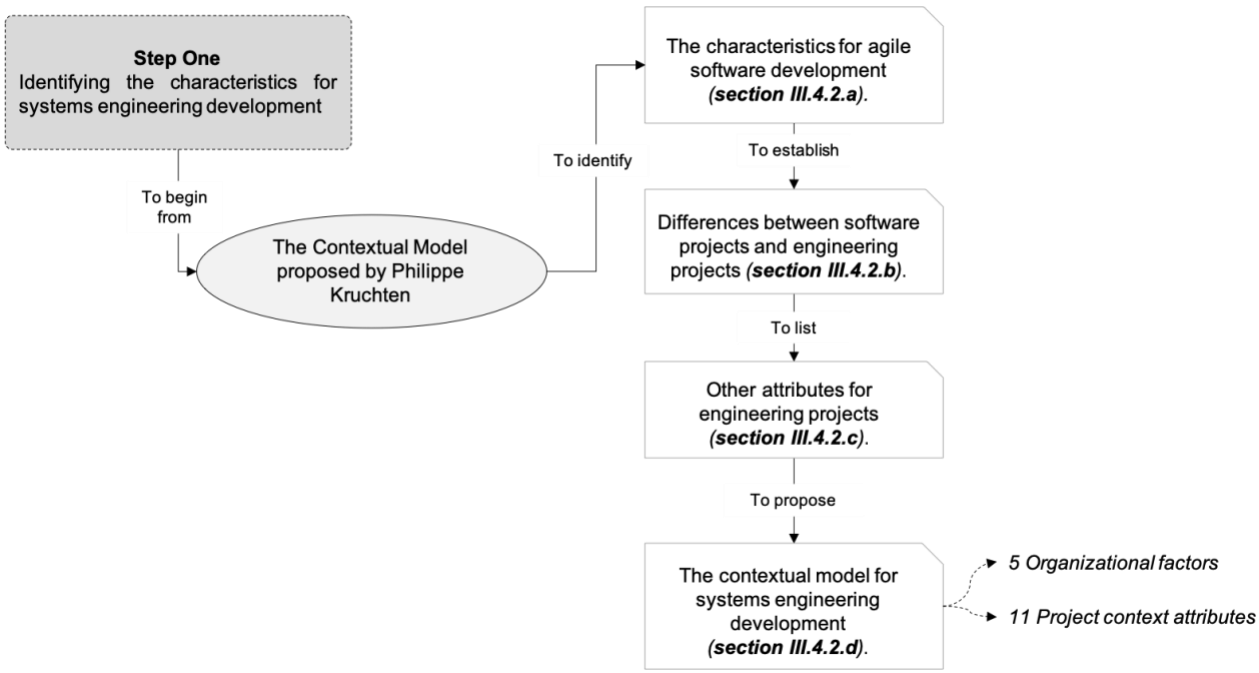

Figure 25. Structure of the Step One

Data, activities and intermediary results of Step 1 are detailed here below.

III.4.2.a Identifying the characteristics for agile software development

The contextual model for agile software development was first introduced in section II.4.1 of the literature review. Philippe Kruchten states that two set of factors make up the context for agile software development. The factors can be portioned in two sets: factors that apply at the level of whole organization/company, and factors that apply at level of the project (Kruchten, 2013).

Figure 26 shows the two sets of factors proposed by Kruchten. Five factors (organizational factors) were defined to define the environmental conditions where the system is being developed, and eight to define the attributes related to the process of the project.

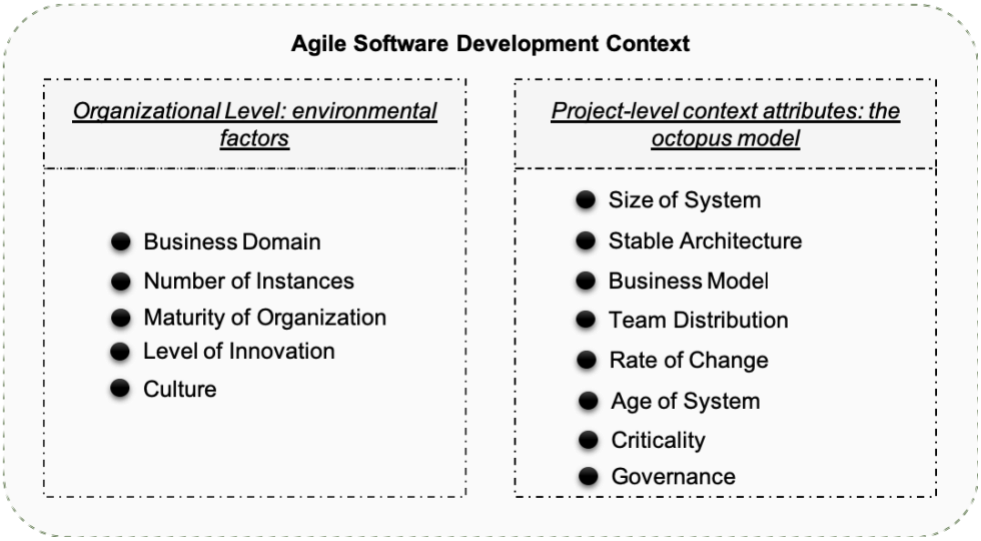

Figure 26. Set of factors for agile software development (Kruchten, 2013).

The organizational factors and the project context attributes contain a description that explain the main characteristics for agile software development. These characteristics were described in section II.4.1. They will be used to identify the differences between software projects (in an agile context) and engineering projects.

III.4.2.b Establishing the differences between software projects and engineering projects

This analysis has the objective to understand how the organizational factors and project attributes are present in engineering projects. The identification of differences, between software (in an agile context) and engineering projects, is proposed by doing an analysis of the two set of factors proposed by Kruchten. First, the five organizational factors are analyzed.

The definition of each factor will help establishing the differences. The differences are made using as a reference some engineering projects found the literature. Then the eight projects attributes are analyzed following the same logic. This analysis also helps to conclude it the set of factors can be used to contextualize systems engineering development.

- *Organizational Level Factors*

  - **Business domain**

    Kruchten Definition: For what domain of activity is this organization developing software? Web-based systems, aerospace embedded systems, small hand-held instrumentation? Is software development the primary business activity of the organization, or at the other end, are we dealing with the IT organization of a business for which software is not at all the primary output.

    How it impacts and differs from non-software projects

    As mentioned in the description of the factor, it helps to identify the type of organization where software is being developed. It was shown in Figure 17 that business domain factor constraints different project-level factors (size of the system, criticality, rate of change, and business model). Agile methods are commonly used and successful when the system size is small, criticality is simple, the system age is greenfield, rate of change is medium or high, and the business model is in house. Business domain factor can help to identify the domain of activity of the organization. This factor could be useful to contextualize industrial projects where the final product is not software.

    A few domains can be identified:

    - Robotics,
    - Mechanics
    - Materials
    - Aerospace
    - Materials
    - Electromagnetics, etc.

    These domains incorporate Hardware/software integration systems, hardware components, systems of systems, and physical prototypes with long turnaround times (Smiley et al., 2018). Dependence on external suppliers is another aspect that differs from software development.

- **Number of instances**

  Kruchten Definition: How many instances of the software system (large or small) will be actually deployed? Are you building one single system, a dozen, a thousand, or millions? One-off systems are often internal to an organization, or developed on demand by a system integrator.

  How it impacts and differs from non-software projects.

  The number of instances factor help us to identify the number of instances/people of the organization involved in the project. This factor impacts the governance factor of the project-level; it is evident that the number of instances/people involved in the project will affect the way it is managed. In systems engineering development the number of instances will depend on the type of the system being developed. We can find systems that are conformed by subsystems (an airplane for example) but is only one system. As mentioned in the business model factor, engineering projects are interdisciplinary, that means that the number of instances in systems engineering development are variable. Even if the system is developed internally different roles can be seen during system development. Some instances were identified based in the cases studies found in the literature, as supply chains, system integrators, operations, and sustainment organizations (Markina-Khusid et al., 2018; Schindel, 2018).

- **Maturity of organization**

  Kruchten Definition: How long has that organization been developing software? How mature are the processes (and the people) relative to software development? Is the organization a small start-up, an SME (small or medium enterprise), or a large multinational firm?

  How it impacts and differs from non-software projects

  The Maturity of Organization factor has not direct influence in project-level. However, it could be linked to systems engineering development. The nature of engineering organizations operating by a command and control top-down hierarchy (Fairbairn, 2018), it can be interpreted that engineering organizations follows disciplined mature process. The maturity of the organization normally depends on how long it has been developing systems. Characteristics such as

process maturity, people, size of the organization, help us to identify how mature the organization is, according to the type of system it develops. This factor could be use in engineering organizations, in order to contextualize systems engineering development.

- **Level of innovation**

  Kruchten Definition: How innovative is the organization? Are you creators or early adopters of new ideas and technologies? Or, treading on very traditional grounds?

  How it impacts and differs from non-software projects

  Innovation is a strong variable in all organizations, and systems engineering is not exception to this. The value of service systems engineering is to do innovation through the use of emerging technologies to propose creation of new service system and value concretion (Walden et al., 2015). The level of innovation factor is influenced by three project-level factors: Age of system, Stable structure, and Rate of change. It can be interpreted that the identification of the level of innovation will depend of the definition of the system architecture, and how this evolves during its development. Almost all organizations consists of a purposeful combination of interdependent resources (people, processes, organizations, supporting technologies, and funding) that interact with each other to coordinate functions, share information, allocate funding, create workflows, and make decisions and their environment (Rebovich, 2006), that means that use of the resources and the coordinate functions could help to identify the level of innovation in an engineering organization. This factor could be directly linked to systems engineering development, the differences in engineering organizations (not exclusively software organizations) could be seen on the project-level factors.

- **Culture**

  Kruchten Definition: In which culture are the projects immersed? Are we speaking here of both national culture and corporate culture? What are the systems of values, beliefs, and behaviors that will impact, support, or interplay with the software development practices?

How it impacts and differs from non-software projects

The Culture factor has not direct influence in project-level, but it could be seen in systems engineering development. Systems engineering pays attention to the culture, sociology, and emotional intelligence of the engineers and the organization (Fairbairn, 2018). The systems of values, beliefs, and behaviors that impact the engineering practices are defined by standards, these standards consider the enterprise environment as a part of their considerations in this context. This factor could be used in engineering organizations, in order to contextualize systems engineering development.

- *Project Level Context Attributes*

  - **Size of System**

    Kruchten Definition: The overall size of the system under development is, by far the greatest factor, as it will drive in turn the size of the team, the number of teams, the needs for communication and coordination between teams, the impact of changes.

    How it impacts and differs from non-software projects

    This factor is present in engineering projects, but some characteristics make it different to non-software projects. Teams distribution in software projects used to be co-located (Smiley et al., 2018). The engineering of complex systems involves multi-discipline teams (Fairbairn, 2018), that means that the project usually is not co-located, and it can be found more than one engineering discipline (systems engineer, mechanical engineer, electrical engineer, and so on) in the project.

    Based on case studies found in the literature, systems engineering teams are:

    - Small, Medium and Big sizes
    - Geographically dispersed
    - Cross-domain

- **Criticality**

  Kruchten Definition: How many people die or are hurt if the system fails? Documentation needs to increase dramatically to satisfy external agencies who will want to make sure that the safety of the public is assured.

  How it impacts and differs from non-software projects

  The current models for systems engineering development establish activities to support compliance, that means that documentation has to be generated in order to satisfy governance and compliance needs for the program (Koehnemann, 2018). This factor is not quite identical, in systems engineering development is more common to find the word “compliance” instead of “criticality”, however, it can be linked. Systems have to be compliant to standards and regulation depending on the industry. There are critical systems that are audited based on specific standards. Based on case studies found in the literature, the prioritization of compliance or required documentation are common activities while developing systems (Batra, Xia, VanderMeer, & Dutta, 2010). It can be interpreted that: Traditional systems engineering practice for pull programs assumes that sell-off is based on verification of compliance with requirements not stakeholder satisfaction (Rosser, Marbach, Osvalds, & Lempia, 2014).

- **Rate of Change**

  Kruchten Definition: Although agile methods are “embracing changes”, not all domains and system experience a very rapid pace of change in their environment. How stable is your business environment and how much risks (and unknowns) are you facing?

  How it impacts and differs from non-software projects

  The rate of change factor is also present in systems engineering development, even if there still are projects with very stable requirement definitions. Repenning et al, states that real work is a constantly evolving mix of routine and uncertainty (Repenning et al., 2017). Routine and uncertainty are aspects present in engineering projects. Traditional engineering projects uses a linear lifecycle model, in which phases and activities occur sequentially for entire projects (Rosser et al., 2014), that means that before the implementation it is important

to understand stakeholders needs and give a completely specifying solution. The differences can be listed as follows:

- The capabilities of the system are defined at the beginning
- Clear stakeholder's needs are defined
- The architecture of the system is well defined
- Changes are made early in the project,
- Use of an up-front design.
- Inflexible to change (Tolentino & Wood, 2018)

- **Business Model**

  Kruchten Definition: What is the money flow? Are you developing an internal system, a commercial product, a bespoke system on contract for a customer, a component of a large system involving many different parties? Is it free, libre, and open-source software (FLOSS)?

  How it impacts and differs from non-software projects

  The Business Model factor can be linked in systems engineering development. There are internal and commercial engineering projects. Systems engineering includes a variety of activities, including technical management, mission and needs analysis, requirements articulation, systems architecture designs, and technical analysis and trades (Rosser et al., 2014). Based on the case studies found in the literature some differences can be identified for non- software projects (Bottani, 2010; Ganguly, Nilchiani, & Farr, 2009; Schindel, 2018):

  - Some systems not only include components (hardware), but also subsystems
  - Organizations face interdisciplinary projects, that means that many different parties are involved

- **Governance**

  Kruchten Definition: How are projects started, terminated? Who decides what happens when things go wrong? How is success or failure defined? Who manage the software project managers?

How it impacts and differs from non-software projects

The governance factor can be interpreted as the path that should be followed to do things in the development of the project. Systems engineering development governance differs from software development. For software development the life cycle starts from project plan, to systems requirement analysis, to system design, to coding, and finally to maintenance (Pressman, 2010). Systems engineering life cycle focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, the proceeding with design synthesis and system validation considering the complete problem, this includes: operations, cost and schedule, performance, training and support, test, disposal, and manufacturing (INCOSE, 2017).

It can be interpreted that systems engineering governance differs in:

- Definition of customer needs: these are the input to start the requirements definition
- The system design is validated before staring the realization of it
- The system validation considers operations, cost, support, test, and manufacturing.

- **Age of System**

Kruchten Definition: Are we looking at the evolution of a large legacy system, bringing in turn many hidden assumptions regarding the architecture, or the creation of a new system with fewer constraints?

How it impacts and differs from non-software projects

In engineering projects evolution and/or creation of a new large systems can be found. The definition of the age of the system in engineering projects is useful to identify what kind of system will be developed. Regarding the architecture, some aspects of the system are explicitly modeled as members of an invariable foundation (Schapiro & Henry, 2012), that means that the evolution of the system can be predicted from the beginning. This factor could be useful to contextualize systems engineering development, according with its description is not limited to software development, because in engineering projects it is possible to create a new system, to evolve an existing one.

- **Stable Structure**

  Kruchten Definition: Is there an implicit, obvious, de facto architecture already in place at the start of the project? Most projects are not novel enough to require a lot of architectural effort. They follow commonly accepted patterns in their respective domain. Many of the key architectural decisions are done in the first few days, by choice of middleware, operating system, programming languages, and so on.

  How it impacts and differs from non-software projects

  The Stable structure factor is related with the architecture of the system. Most software projects have a stable structure, and the definition of the architecture is based according to operating system, programming language, etc.; that means that there is not physical (hardware) integration during the project. For that reason, in software development is very common to find a stable structure. In systems engineering development is quite different, the definition of the architecture requires a diverse engineering team (systems architects, systems engineers, and simulation programmers) and dependences can be found during its definition. Traditional systems engineering processes, where architects finalized architecture during the initial concept development stages and engineers develop system simulations during later lifecycle phases, do not support a rapid architecture evolution (Maheshwari, Raz, DeLaurentis, Murphy, & Kolawole, 2018). It can be interpreted that in systems engineering development the architecture of the system is not common stable, and once it is fixed, enabling reconfiguration is quite difficult.

  Another consideration is that in large and complex systems there are dependencies between the system capabilities and architectural elements (Rosser et al., 2014). We can conclude that the characteristics that differ systems engineering development are:

  - Unstable architecture,
  - Dependences between system capabilities and architectural element,
  - Integration of physical elements (hardware).

- **Team Distribution**

  Kruchten Definition: Linked often to the size of the project, how many teams are involved and are not collocated? This increases the need for more explicit communication and coordination of decisions, as well as more stable interfaces between teams and between the software components that they are responsible for.

  How it impacts and differs from non-software projects

  The team distribution factor is defined for the identification of roles involved in the development of the project. There are twelve roles defined in systems engineering with two views. One view considers the job of systems engineers to be coordinating, tracking, and managing the engineering of the system and its subsystems (Program Management Roles), the other view considers the job to be a set of specific life-cycle tasks (Life-Cycle Roles). The roles that integrates each view are (Sheard, 1996):

  - Program Management Roles: Technical Manager, Glue among subsystems, Information Manager, Coordinator, Customer Interface.
  - Life Cycle Roles: Requirement Owner, System Designer, Validation and Verification Engineer. Logistics and Operations engineer, and System Analyst.

  In contrast, the roles defined in software development are: Project Manager, system administrator, designer, and programing (Zhu, Zhou, & Seguin, 2006). It can be noticed that the way to define the team distribution in engineering projects is not quite similar for software projects.

The analysis presented above identifies the differences between software development and systems engineering development. The identified differences help to understand, whether the organizational factors and project attributes could be used to contextualize systems engineering development. Table 10 summarizes the results of this analysis.

Factor		Useful to contextualize systems engineering development?
Organizational level	Business Domain	Factor need caution to the adoption
	Number of instances	Useful
	Maturity of the organization	Useful
	Level of Innovation	Useful
	Culture	Useful
Project-level context attributes	Size of System	Factor need caution to the adoption
	Stable structure	Factor requires adaptation
	Business model	Useful
	Team distribution	Factor need caution to the adoption
	Rate of change	Factor need caution to the adoption
	Age of system	Factor need caution to the adoption
	Criticality	Factor requires adaptation
	Governance	Useful

Table 10. Summary of the differences between agile software projects and engineering projects.

The factors that need caution, to be adopted in engineering projects, are shaded in light grey, and the factors that require adaptations are shaded in black gray. It can be noticed (from Table 10) that only one factor (Business Domain), from the organizational factors, needs caution to be adopted in the contextualization of engineering projects. The reason is because in engineering organizations different domains can be found for the project context attributes, the color variation is more variable. Four project attributes (Size of System, Team Distribution, Rate of Change, and Age of System) needs caution to be adopted in engineering projects, the reason is because many engineering disciples are present in engineering projects. Two project attributes (Stable Structure and Criticality) requires adaptations to be used in the contextualization of engineering projects. The number of instances, Maturity of the organization, Level of innovation, Culture, Business model, and Governance factors can be used for the contextualization of engineering projects.

The results of this analysis show that Kruchten' contextual model could be somehow used to contextualize systems engineering development. However, a question comes

out of this analysis: Are there other factors that are not implicit in the Kruchten model? These questions will be addressed in the following activity of the step one.

III.4.2.c Listing other factors or attributes for systems engineering development

An examination will be made to identify if there are any other factors that have not been mentioned by Kruchten, and can be used to contextualize systems engineering development. By using different systems life cycles, an analysis will be proposed to identify new factors in systems engineering development. Once the new factors or project attributes are listed, a list of the new factors will be proposed.

A system progresses through a common set of life cycle stages where it is conceived, developed, produced, utilized, supported, and retired (Walden et al., 2015). Figure 27 introduces different life cycles models, these models are used for the development of systems, included software development.

The number and name of the stages may vary according to the context in which the life cycle model is being applied. Similarities can be found in the following models, for example the study period phase is present in the six models presented in Figure 27. In other life cycle models this stage is called differently (exploratory stage, concept studies, etc.).

Many industries employ an exploratory research activity to study new ideas or enabling technologies and capabilities, which then mature into the ignition of a project.

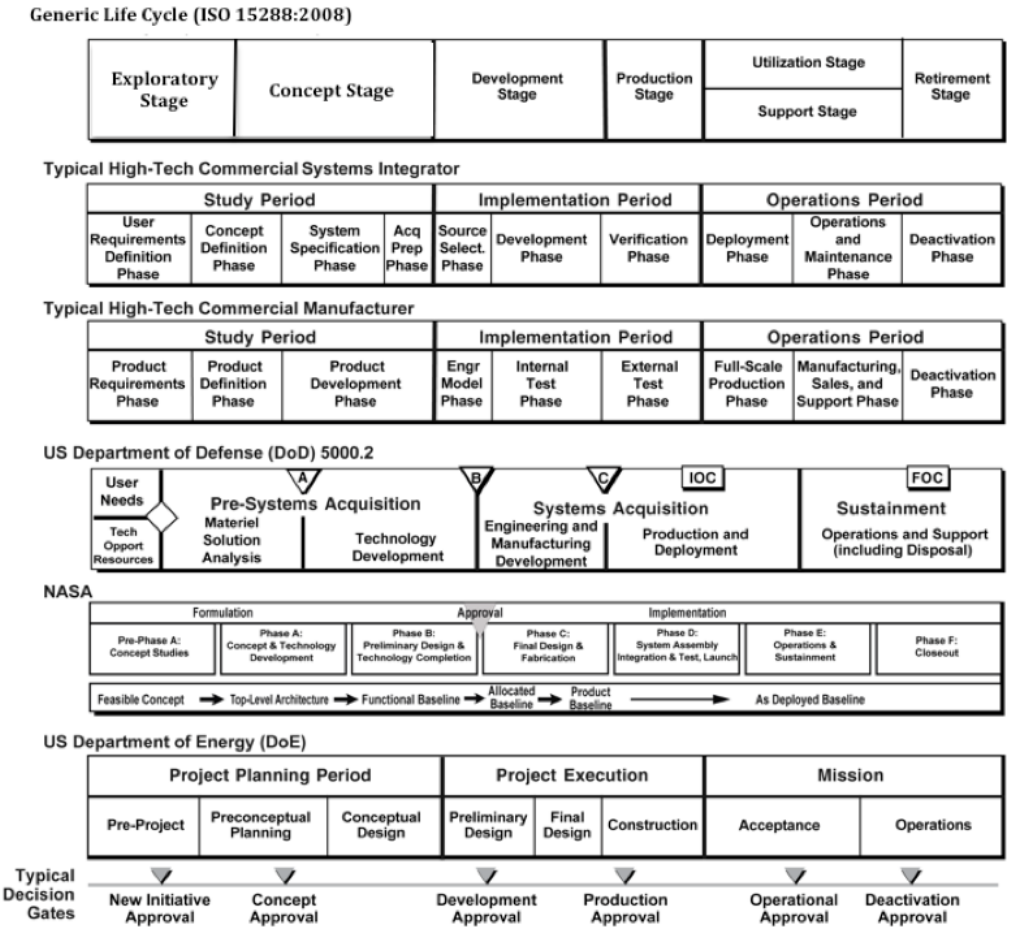

Figure 27. Comparisons of Life Cycle Models (Forsberg, Mooz, & Cotterman, 2005)

It can be noticed (from Figure 27) that many traditional system life cycles models include the retirement stage, the purpose of this stage is to store, archive, or dispose of the system (Walden et al., 2015). This characteristic is not implicit in the factors prosed by Kruchten, but it is important in systems engineering context. The six cycles also include retirement stage, this characteristic is not implicit either. Other aspects that were not proposed in Kruchten model, were the complexity of the system, and the use and creation of different technologies. For the complexity of the system, Kruchten involves the meaning of complexity in the size and stable architecture project attributes. In the case of the use/creation of new technologies**,** he states about the use of new technologies to innovate the system (in the level of innovation factor), but for engineering projects, there is a possibility of creation of new technology (US DoD, and NASA life cycle). Lots of different engineering disciplines are involved in systems engineering, and the connection between components in a system is important, that means that the development of complex systems is very common in systems engineering.

After the analysis of the different systems life cycles, it can be concluded that, in addition to the contextual model proposed by Kruchten, some project attributes should be defined including: complexity, use/creation of different technologies, and deactivation of the system.

The new three project attributes could be defined as follows:

- *System Complexity*

  Linked to the size and type of architecture, how the complex is the work to be done? The system involves independently systems of system? Is the system being developed in multi-platform or in cross disciplinary fields? The organizations that create and utilize systems face challenges which are associated with the complexities of them (IEC/IEEE, 2015). Fixed relationships can be found in the interactions between many parts of the system (Walden et al., 2015),
- *Technology of the System*

  Is there a level of technology readiness in the project? Are you inserting a new technology? Is there an associated demand of relevant technology? Systems engineering should support project management; this includes system/project constraints as technology limitations (Walden et al., 2015),
- *Operation of the System*

  Is the system need to be retired from its environment? Disposal stage, in systems engineering, defines the constraints to permanently terminate the system's functions and delivery services. The transformation of the system in a socially and physically acceptable state, according to the environment and society is the primary characteristic of this factor.

The three preceding activities (III.4.2.a, b and c) helped to highlight the differences between agile software projects and engineering projects. Some organizational factors and project attributes need caution, and others require adaptations to be adopted in engineering projects. Three new attributes were defined to be considered for the contextualization of systems engineering aspects. The results of these three activities will help to propose the contextual model for systems engineering development. A new contextual model, for engineering projects, will be described in the following activity.

III.4.2.d Proposing the Contextual model for systems engineering development.

This activity aims to propose a contextual model for systems engineering development. First, the organizational factors, that need caution or require adaptations, will be addressed to better redefine them (listed in Table 10). Then the project attributes will be addressed by following the same logic. Finally, the list of the organizational factors and project attributes, for engineering projects, will be presented at the end of this activity.

- Organizational level

In this level, the Business Domain was the only factor affected to the adoption in engineering projects. The word “software” could be replaced for “system”. The new definition of this factor is proposed as follows:

  - *Business Domain*

    For what domain the organization is developing the system? The system incorporates hardware components, systems of systems, and physical prototypes? The identification of the primary business activity can define the business domain of the organization (robotics, aerospace, materials, mechanics, etc.).

- Project level context attributes

For the project attributes, six attributes were affected to the adoption in engineering projects. The modifications were based by using the differences (listed before) found in literature. The new definition of each attribute is proposed as follows:

  - *Size of System*

    The size of the system under development will drive the size of the team involved. The impact of changes will be coordinated accordingly to the multi-discipline, cross-domain, and the location of teams. Number of person-months, complexity of the system, and budget are all possible proxies for the size,

  - *Type of Architecture*

Is the architecture of the system already in place at the start of the project? The definition of the architecture implies a diverse engineering team? How many dependences exist in the definition of the architecture? Is the architecture stable or not? Most engineering projects require a lot of architectural effort and the key architectural decisions are done during the initial concept development stages. Other engineering projects has stable architectures. The dependences between system capabilities and architectural elements could be stable, and also unstable,

- *Team Distribution*

  Team distribution is often linked to the size of the project; how many roles/teams are involved in the development of the system? Are the teams/roles are collocated? The use of stable interfaces between teams and the systems components are needed, as well as, explicit communication and coordination of decisions,

- *Rate of Change*

  Linked to the type of architecture, how stable is your business environment and how much risks, or uncertainties are you facing? Traditional engineering projects uses a linear lifecycle model, there are still projects with very stable requirement definitions and a global vision of the solution is identified,

- *Age of System*

  Are we looking at the evolution of a large legacy system with many constraints regarding the architecture? Are we developing a new system with fewer constraints? In engineering projects, the evolution of the system could be predicted from the beginning of the project,

- *Criticality*

  Documentation has to be generated in order to satisfy the safety of people during the use of the system. Systems have to be compliant to standards and regulation depending on the industry. There are critical systems that are audited based on specific standards.

The contextual model, for systems engineering development, is the final result of the set of activities followed in the Step One. Figure 28 shows the list of organizational factors and project attributes for engineering projects. Five organizational factors, and eleven project level contextual attributes were defined. Table 11 presents the seventeen factors, and their brief definition.

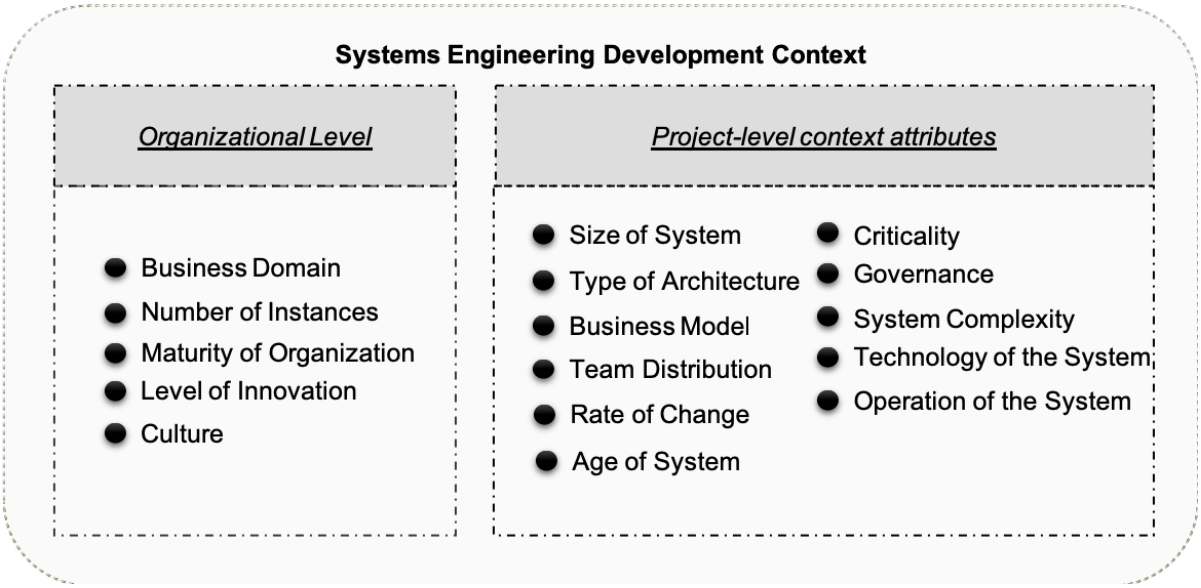

Figure 28. Contextual Model for systems engineering development

Level	Factor/Attribute	Brief Definition
Organizational	*Business domain*	The business domain in which the organization operates
	Number of instances	The number of instances involved in the organization
	Maturity of the organization	The maturity of the process used for the development of system un the organization.
	Level of Innovation	The level of innovation involved in the development of systems in the organizations
	Culture	The values, beliefs, and behaviors, that impact the system development practices
Project attributes	*Size of System*	The overall size of the system under development
	Type of structure	The type of system architecture
	Business model	The business model in which the system is developed
	Team distribution	The number of teams/roles involved in the project
	Rate of change	How much uncertainties and risk the project is facing
	Age of system	The type of system (new or evolution from an existing one) developed during the project
	Criticality	The compliance that the system has to accomplish
	Governance	How the project is managed
	System Complexity	How the complex is the work to be done to develop the system
	Technology of the system	The technology involved in the development of the system
	Operation of the system	Deactivation of the system when it is no longer useful or used

Table 11. Contextual model for systems engineering development

The Step one helps to conclude that:

- The factors to contextualize agile software development could be used to contextualize systems engineering development. Some adaptations were made to refer engineering projects.
- The relationship between Culture and Business Model factors was not established by Kruchten. However, it can be found in literature that the structure of the organization has a valuable role in reinforcing its culture (Fairbairn, 2018). This relationship can be established to create the adoption of a new mindset (in our case agile mindset) in projects where multiple disciplines are present.
- The contextual model for systems engineering projects could help to define the adoption of an agile method for the management of engineering projects.

III.4.3 Step Two: Analyzing Agile Methods

The Step Two, *"Analyzing Agile Methods"* aims to select an agile method that could be used for the management of engineering projects. Figure 29 shows the structure of this step. By using the contextual model for systems engineering development, first, criteria will be defined for some project attributes. Then, an analysis, similar to the one in the section II.4.1 , will be made in order to identify how the agile methods are present in the project attributes. Finally, the selection of the most adaptable agile method will be made.

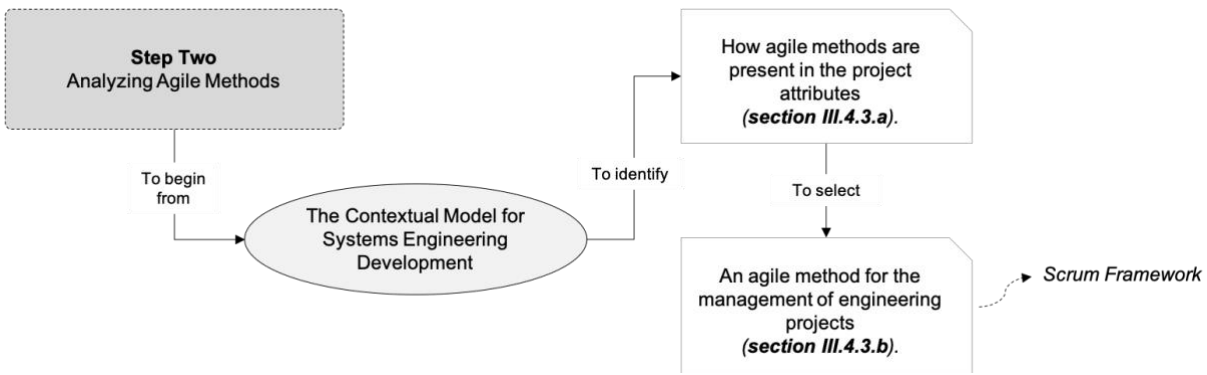

Figure 29. Structure of the Step Two

III.4.3.a Identifying agile methods presence in the project attributes

Table 12 shows the distribution of the criteria for some project attributes. The criteria are defined according to common characteristics in the engineering projects. Only six project attributes contain different criteria. It will help to better understand the prescience of agile methods in the project attributes.

Factor	Criteria	Abbreviation
Size of System	Small	S
	Medium	M
	Large	L
Type of structure	Stable	S
	Unstable	U
Team distribution	Small	S
	Medium	M
	Large	L
	Extra Large	XL
Rate of change	Low	L
	High	H
Criticality	Low	L
	Medium	M
	High	H
System Complexity	Low	L
	High	H

Table 12. Criteria for some project level attributes

The rest of the project attributes (Business Model, Age of System, Governance, Technology of the System, and Operation of the System) will be evaluated directly.

Six agile methods were introduced in the literature review. A comparative analysis was introduced, in section II.4.1., to identify how these methods were present in the contextual model proposed by Philippe Kruchten. Table 13 introduces a similar comparative analysis using the contextual model for systems engineering development, and the different criteria for the project attributes.

	Size			Type of the Architecture		Team Distribution				Rate of change		Criticality			System Complexity		
	S	M	L	S	U	S	M	L	XL	L	H	L	M	H	L	H	**Total**
Scrum	*	*		*	*	*	*			*	*	*	*		*	*	12/16
XP	*			*		*				*	*	*			*		7/16
Kanban	*			*		*				*	*	*			*		7/16
Crystal	*	*	*	*		*	*	*	*	*	*		*	*		*	13/16
DSDM		*	*		*			*	*		*			*		*	8/16
FDD	*			*		*	*			*	*	*			*		8/16

Table 13. Vision of agile methods in systems engineering development project attributes

III.4.3.b Selecting the agile method for the management of engineering projects

Table 13 summarizes how agile methods cover the project attributes. In this first analysis, 16 criteria were defined in six project attributes. It can be noticed that Scrum and Crystal methods are the ones that cover more criteria of the 16 criteria defined. Crystal has 13/16 criteria, and Scrum only has 12/13. However, Crystal methods do not cover unstable architectures. On the other hand, Scrum covers stable and unstable architectures. In terms of complexity, Crystal is based on the size and critically, and it needs to put in place some aspects to accommodate the additional requirements, while Scrum deals with complexity using the daily meeting practice. Daily meetings are held at the same time and place each day to reduce complexity (Schwaber & Sutherland, 2017). It can be interpreted that the complexity of the project is evaluated continuously. Scrum is also well known and used for the development of systems in different domains. Up to this point Scrum and Crystal seem to be the agile methods that can be adopted for the management of engineering projects, but before concluding which would be the most adaptable, it is important to evaluate the rest of the project attributes.

Table 14 introduces the last 5 project attributes. This table only considers Scrum and Crystal methods.

	Business model	Age of system	Governance	Technology of the system	Operation of the system
Scrum	Multi-context	New system / renew system	Well defined	Considered in its practices	Not listed
Crystal	Software context only	New system	Well defined	Not listed	Not listed

Table 14. Vision of Scrum and Crystal in project level context attributes

It can be noticed that Scrum has more advantages than Crystal in the last 5 project level context attributes. Scrum can be used: to develop a new system or renew an existing one, to face changes that are associated with technology evolution, and different business model. Governance as well defined in both methods, and the operation of the system is not implicit in the practices of the methods.

The Step two help to conclude that:

- Two agile methods (Scrum and Crystal) could be used for the management of engineering projects,
- Crystal has more prescience in the first analysis introduce in Table 13, and Scrum is not only used in software development, it is also widely used for products, services, and the management of the organization,
- By seeing all the project attributes together, Scrum is the agile method with the greatest coverage in project level attributes for systems engineering development,
- An analysis of the Scrum Framework has to be done to deploy it for the management of engineering projects.

This analysis of the Scrum Framework will be introduced in the following step of the research methodology.

III.4.4 Step Three: Analyzing the Scrum Framework

The Step three, “Analyzing the Scrum Framework”, aims to introduce the use of Scrum Framework for the management of engineering projects. Figure 30 shows the main activities of this step. First, the distribution of the Scrum Framework will be analyzed to well understand how it works. Scrum Framework is globally distributed in: values, teams and their associated roles, events and artifacts. These aspects follow rules that bind them together. The rules are listed in the Scrum Guide™. By using the rules, or the Scrum Guide™, the main practices will be identified. The Scrum practices will be proposed like Scrum Team Practices (STP) and Scrum Events Practices (SEP); then these practices will be adopted as a guide for the management of engineering projects. Finally, an analysis, to evaluate the level of impact of the project attributes in the Scrum Practices, will be proposed. The adoption of Scrum Practices, in systems engineering projects, may have difficulties, which may be associated with the content of the framework, and the characteristics of the project attributes. These difficulties will be listed and addressed in the final step of the research methodology.

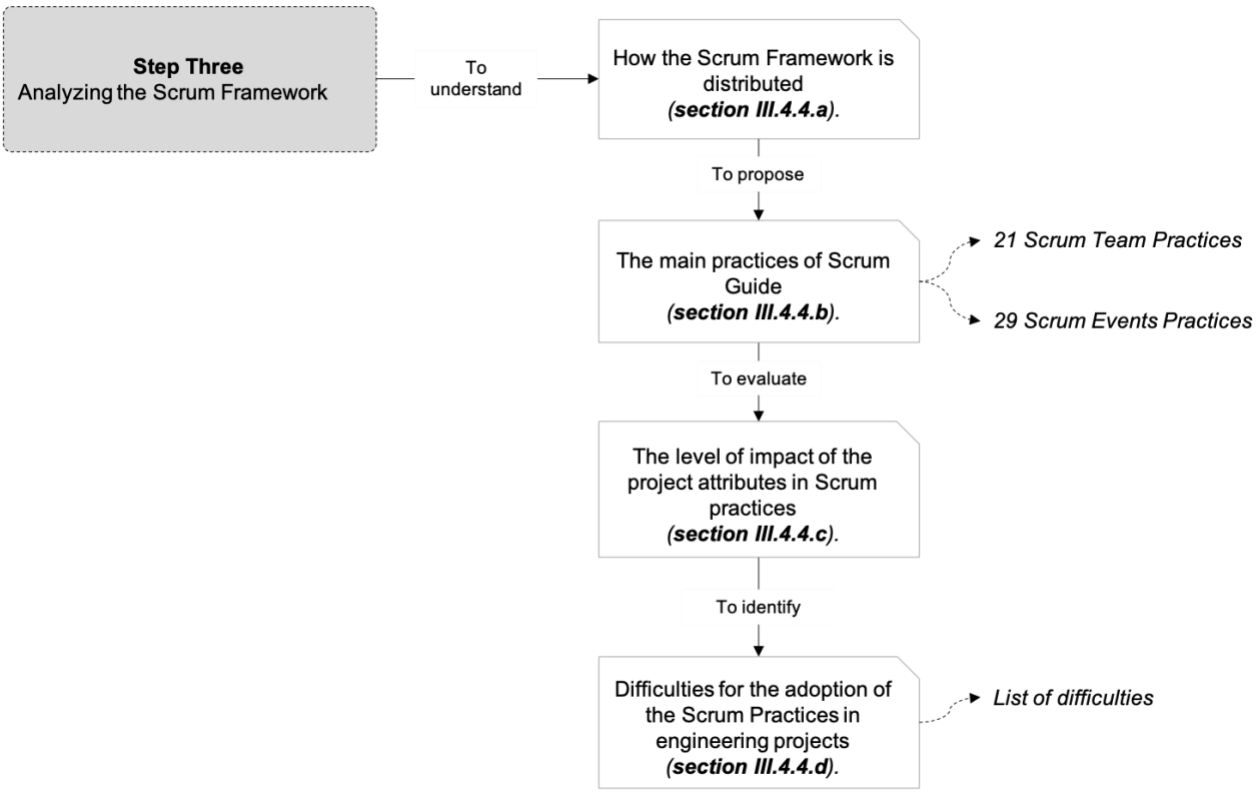

Figure 30. Structure of the Step Three

III.4.4.a Understanding Scrum Framework

The Scrum rules govern the relationships and interaction between roles, events, and artifacts. Figure 31 shows an interpretation of the Scrum Framework, it can be noticed that events, artifacts, and roles have independent interactions, and they have relationships between them. Scrum rules bind them together to be unified and governed in the same vision by following the core of values such as courage, focus, respect, commitment and openness.

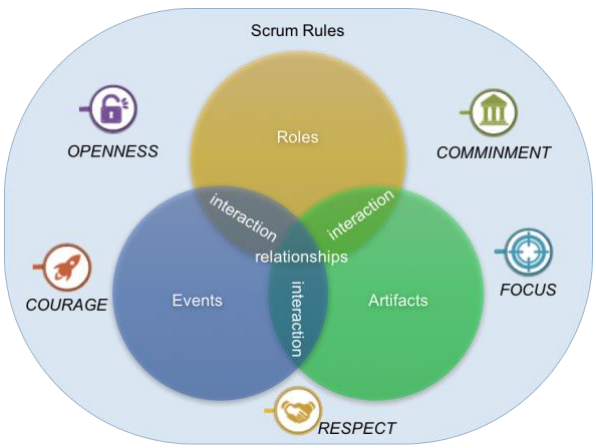

Figure 31. Relationships and interaction of scrum roles, events and artifacts.

The Scrum theory is founded in the implementation of empirical process control, the implementation integrates three pillars: *transparency*, *inspection*, and *adaptation.* The graphical view of Scrum Framework was first introduced in the literature review chapter of this work (section II.1.4). The graphical view shows the order of how roles, events, and artifacts are used. A new interpretation of the graphical view is shown in Figure 32, this view helps to identify the aspects mentioned before, and the pillars of Scrum Theory. It can be noticed, in the figure, that the sprint planning, daily scrum, sprint review, and sprint retrospective, are events. Product backlog, sprint backlog, and increment are artifacts.

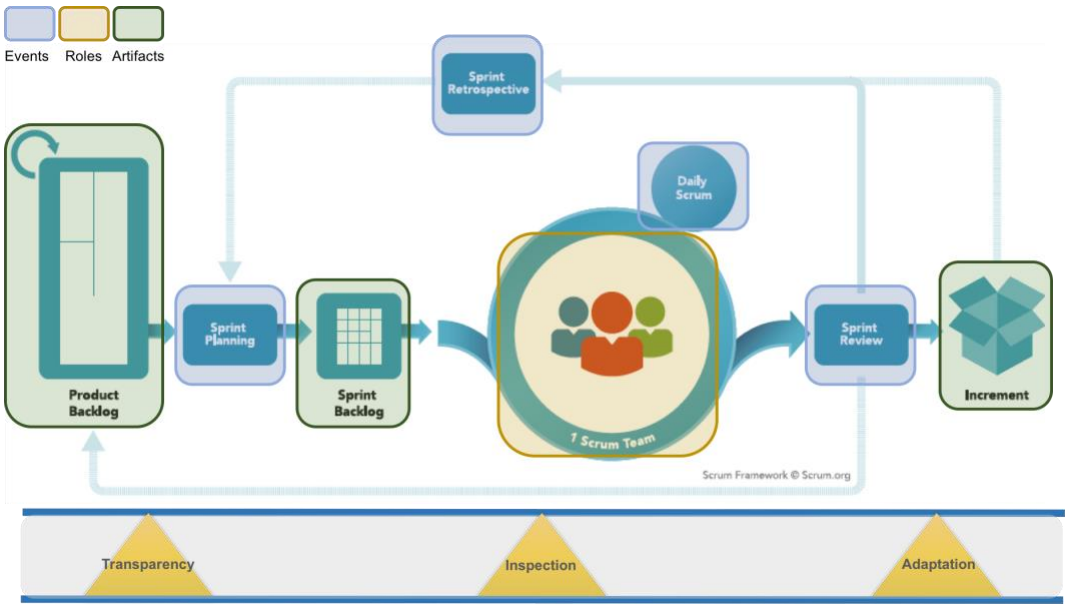

Figure 32. Graphical view of roles, events, artifacts and pillars in scrum framework.

The entire framework is supported by the three pillars. Scrum Framework begins with the definition of the Product Backlog (the list of everything that is known to be needed in the product), this definition helps to plan the Sprint Planning. Sprint Planning plans the work to be performed in each sprint; all the sprints to be performed become the Sprint Backlog, which means that the Sprint Backlog is the product Backlog items selected for the Sprint, plus a plan for delivering the product increment. Once the work is planned and distributed, the Scrum team accomplish the work, and hold daily meetings to inspect it. The inspection of the Sprint is done in the Sprint Review phase. Sprint Review allows the Development Team to inspect the Sprint and if there are not changes the Sprint become an Increment, (the sum of all the Product Backlog Items completed during a Sprint and the value of the Increments of all previous Sprints).

Otherwise if there are improvements, The Sprint Retrospective phase is activated. Scrum Retrospective helps the Development Team create a plan for improvements to be enacted during the next Sprint, and then the process begins again in the Sprint Planning Phase.

This new graphical view can be used as the first vision when a new engineering project will be planed. It is also important to consider that the Sprint is the most important element of the Scrum. The Scrum Guide™ states that the "Sprint" is the heart of Scrum Framework (Schwaber & Sutherland, 2017). Sprints have consistent duration throughout a development effort. Figure 32 does not introduce the "Sprint", but the definition of the Sprint is well stablished in the Scrum Guide™ 2017.

Figure 33 shows the elements that contain the Sprint. Each sprint may be considered one-month horizon project, the Sprint is used to accomplish something during the entire development. It can be interpreted; from the Figure 33, that the Sprints are planned and inspected during the entire Scrum Framework.

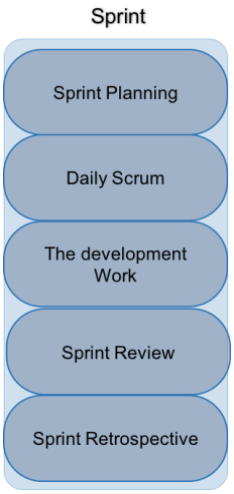

Figure 33. The heart of Scrum: The Sprint

The scrum values will be used throughout the integration of Scrum in engineering projects, that means that the sense, of each value, will be considered as they were described in the literature review chapter.

III.4.4.b Proposing the main Scrum Practices

This activity aims to list the main Scrum Practices. It can be interpreted from the Scrum Guide that the framework considers two aspects: *" the way teams are distributed ", and " the way work is organized "*. The Scrum Practices will be identified by following these two aspects, and will be used as a complement of the graphical vision of the Scrum Framework.

- *The Scrum Team Practices (STP)*

The STP practices will be defined first. It is important to remember that The Scrum Team covers three roles (the roles are well explained in section II.1.4):

- The Product Owner (PO), the one in charge for the vision of the product,
- The Development Team (DT), professionals who do the work to develop the product, and
- The Scrum Master, the one in charge for promoting and supporting Scrum.

The scrum team optimizes flexibility, creativity, productivity and feedback, by delivering "increments" in iterative/ incremental mode (Schwaber & Sutherland, 2017). The Scrum Team Practices are listed in Table 15 with the role (s) in charge of the practice.

Id	Practice	Why use?	PO	DT	SM
STP1	Clear definition of Product Backlog (PB) items	To know the list of everything is needed in the system	*		
STP2	Prioritize the items in the PB	To order the items in the PB to best achieve goals and missions	*		
STP3	Optimize the value of the work	To know the performance of the development team	*		
STP4	Ensure visibility, transparency, and clarity of the PB	To show what the Scrum Team will work on next	*		
STP5	Good understanding of the PB's items	To ensure that all levels in the DT understand the items of the PB	*		
STP6	Self-Organized	To give team members the option to choose what they work on and who they work with	*	**	*
STP7	Cross-functional	To improve the flow of work when several teams are involved in system development	*	**	*
STP8	Start with a small team	To improve the productivity of the development team		*	

STP9	Ensure goals, scope and product domain	To increase the understanding by everyone on the Scrum Team as well as possible	*		**
STP10	Find techniques to effective PB management	For the effective management of the PB	*		**
STP11	Help the DT for the good understanding of needs	To help the Scrum Team understand the need for clear and concise product backlog items	*		**
STP12	Understand product planning	To understand product planning in an empirical environment	*		**
STP13	Maximize value	To ensure the PO knows how to arrange the PB	*		**
STP14	Scrum events facilitation	To facilitate scrum events as requested or needed	*	*	**
STP15	Coach the DT	To ensure that the DT works in self-organization and cross-functionality		*	**
STP16	Crate high-value	To help the DT to create high-value products		*	**
STP17	Ensure progress	To remove impediments to the DT's progress		*	**
STP18	Lead and coach the organization	To help the organization for the adoption of scrum			*
STP19	Plan with the stakeholders and employees of the organization	To plan and understand the scrum implementation with the organization			*
STP20	Create change	To cause change that increases the productivity of the scrum team			*
STP21	Detect differences between expected and real results	To detect artifact transparency by inspecting artifacts, sensing patterns, listening closely to what is being said, and to increase the transparency of the artifacts	*	*	**

Table 15. The Scrum Team Practices

There are STP practices that are in charge of one role, two roles, and others of the whole Scrum Team. Table 15 shades the STP practices that involves more than one role, light gray for those involving two roles, and dark gray for those involving the whole Scrum Team. For the STP practices that involves more than one role, a double red **"*"** was defined in order to indicate who is the role in charge of that STP practices. For example (see Table 15), STP6 and STP7 practices are used by the whole Scrum Team, however, these practices are associated with the DT, that means that this practice will be mainly used by the DT. Another example, STP9-STP11 practices involve the PO and the SM, but these practices will be mainly used by the SM, because SM is at service of the PO. It can be interpreted, from this table, that the Scrum Master is the role, of the Scrum, that uses more STP practices.

- *The Scrum Events Practices (SEP)*

The identification of the Scrum Events Practices follows the same structure of the STP practices. Table 16 lists the SEP practices and adds the Event related to each practice. Scrum Guide consider the events as time-boxed events, that means that events have a maximum fixed duration, and they are specifically designed to enable critical transparency and inspection.

Id	Practice	When or Why use?	PO	DT	SM	Event
SEP1	Define the sprint goal	To know what is to be built, a design, and flexible plan that will guide the building it, the work, and the resultant product increment	**	*	*	Sprint
SEP2	Avoid change of the sprint goal	No changes are made that would endanger the sprint goal	*	**	*	
SEP3	Maintain objective quality	Quality goals do not decrease		*		
SEP4	Define the sprint scope	To help the PO and DT negotiate the scope of the sprint	**	*	*	
SEP5	Cancel the sprint if necessary	When the sprint goal become obsolete, or when the PO is under influence from the stakeholders, DT, or the SM.	**	*	*	
SEP6	Consider one-month horizon for the sprint	When a Sptint's horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase.	*	*	**	
SEP7	Define the work to be performed	To know what can be delivered in the increment resulting from the upcoming sprint, and to how will the chosen work will get done	**	*	*	Sprint Planning
SEP8	Predict functionality	To forecast the functionality that will be developed during the sprint by the DT		*		
SEP9	Start with the design of the system and the work needed	To help the DT to convert the product backlog into a working product increment		*		
SEP10	Clarify the selected PB	To determine if the DT has too much or too little work, according to the selected items from the PB	*	**		
SEP11	Explain how the work done became an increment	To explain to the PO and the SM how the DT accomplish the Sprint Goal and create an increment		*		
SEP12	Hold daily 15-minutes meeting	To optimize team collaboration and performance by inspecting		**	*	Daily Scrum

		the work since the last daily scrum				
SEP13	Inspect Progress toward the Sprint Goal	To inspect how progress is trending toward completing the work in the sprint backlog		*		
SEP14	Focus on progress toward the Sprint Goal	To help the DT identify the work already done and the work to be done		*		
SEP15	Adapt, or replan the sprint's work	To adjust the sprint work if needed	*	**	*	
SEP16	Improve communication	To eliminate unnecessary meetings		*	**	
SEP17	Identify impediments to development	To remove impediments that affect the development of the system		**	*	
SEP18	Promote quick decision-making	To eliminate waste of time		*	**	
SEP19	Inspect the Increment	To elicit feedback and foster collaboration	*	*	**	Sprint Review
SEP20	Adapt the product backlog if needed	To identify any changes in the product backlog	**	*	*	
SEP21	Consider all the stakeholders to the sprint inspection	To ensure that all persons, involved in the development of the system, are present for the increment inspection	**	*	*	
SEP22	Explain what have been done and what has not been done	To identify what went well during the sprint, what problems it ran into, and how the problems were solved		*		
SEP23	Answer the question about the increment	To give a clear understanding of the work done to the stakeholders		*		
SEP24	Deliver dates based on progress to date	To define what to do next to subsequent sprint planning	*			
SEP25	Identify valuable inputs to subsequent Sprint Planning	To help the entire group collaborates on what to do next	**	*	*	
SEP26	Identify the most valuable thing to do next	To have a review of how the marketplace or potential use of the system might have changed	**	*	*	
SEP27	Define the probable PB items for the next Sprint	To define the probable PB items for the next Sprint, once the Sprint review is finished.	**	*	*	
SEP28	Review timeline, budget, potential capabilities and marketplace	For the next anticipated releases of functionality or capability of the system	**	*	*	
SEP29	Inspect the last Sprint	To know how the last sprint went with regards people, relationships, process, and tools	*	**	*	Sprint Retrospective
SEP30	Order the items that went well	To identify the major items that went well and potential improvements	*	**	*	

SEP31	Create a plan for implementing improvement	For the continuous improvement of the product	*	**	*	
SEP32	Increase product quality by improving work process	To identify improvements that will be implemented in the next sprint	*	*	**	
SEP33	Focus on inspection and adaptation	To provide a formal opportunity to make improvements at any time	*	*	**	
SEP34	Product Backlog refinement	To add detail, estimate, and order the items in the product backlog.	**	*		Product Backlog
SEP35	Update Items at any time	To make sure that the list of everything, to be needed in the product, is complete		*		
SEP36	Order the product backlog items	To identify the items that need to be high detailed.		*		
SEP37	Make visible all the work	To make visible all the work that the DT identifies as necessary to meet the sprint goal.		*		Sprint Backlog
SEP38	Make a highly visible sprint backlog	To give a real-time picture of the work that the DT plans to accomplish during the sprint		*		
SEP39	Change the Sprint Backlog if needed	To add new work if needed or to remove elements of the plan if they are unnecessary		*		

Table 16. The Scrum Events Practices

SEP practices are globally dedicated to the planning and management of the sprint. Different SEP practices involves more than one role. Light gray shade covers the STP practices involving two roles, dark gray shade involving the whole Scrum Team, and the double red "**" indicates the role in charge in each SEP practice. To identify the limitations of Scrum Framework for the management of engineering projects, the following activity will evaluate the impact of the project attributes in the Scrum Practices. The analysis, of the level of impact, will help identify the limitations and difficulties while using Scrum Framework in Systems Engineering context.

III.4.4.c Evaluating the level of impact of Project Attributes in Scrum Practices

This activity aims to evaluate how the Scrum Practices are affected by the project attributes. First, a definition of "levels of impact" will be proposed to evaluate the level of impact of the project attributes in Scrum Practices. Then a nomenclature will be added to the project attributes. The identification of the level of impact in Scrum

Practices will help to highlight the difficulties and limitations of the Scrum Framework in Systems Engineering context.

- *Definition of the "level of impact"*

Three levels of impact will be considered to identify the impact of the project attributes in Scrum Practices. The three levels are defined in Table 17.

Level of Impact /Color	Description
Minimal or not impact (1)	The scrum practice has minimal impact or dos not has impact by the project attribute, that means that the practice dos not affect its adoption in engineering projects
Medium impact (2)	The scrum practice has medium impact by the project attribute, that means that the practice needs caution, or may have some limitations when it's being used in engineering projects.
High Impact (3)	The scrum practice has high impact by the project attribute, that means that some difficulties can be present while using the practice for engineering projects.

Table 17. The three levels of impact

- *Nomenclature for the project level context attributes.*

The project level attributes will be divided in two parts, first we will see how scrum practices impacts the first sixth attributes, we will use the distribution of Table 18.

Attribute	ID	Criteria
Size of System	**A1**	Small **(S)**
		Medium **(M)**
		Large **(L)**
Type of Architecture	**A2**	Stable **(S)**
		Unstable **(U)**
Team Distribution	**A3**	Small **(M)**
		Medium **(M)**
		Large **(L)**
		Extra Large **(XL)**
Rate of Change	**A4**	Low **(L)**
		High **(H)**
Criticality	**A5**	Low **(L)**
		Medium **(M)**
		High **(H)**
Systems Complexity	**A6**	Low **(L)**
		High **(H)**

Business model	**A7**
Age of system	**A8**
Governance	**A9**
Technology of the system	**A10**
Operation of the system	**A11**

Table 18. Nomenclature for the project attributes

Table 18 shades (in gray) the nomenclature for the attributes, and bold the criteria of each attribute.

- *Identifying the impact of project attributes in scrum practices*

This task aims to identify what are the limitations and difficulties of the Scrum Framework in project attributes for engineering projects. Table 19 shows the distribution of the level of impact of the project attributes in the Scrum Practices, this table contains the 21 Scrum Team Practices, and 39 Scrum Events Practices. The previously defined “level of impact” were used, and assigned to each Scrum Practice. The description of the practice and the description of the project attribute were considered to evaluate the level of impact; Table 19 only contains the results according to the criteria and nomenclature defined before in this section. Scrum roles are also included in the distribution of Table 19, it will help to identify (once the difficulties and limitations have been listed) the role, or roles that will be involved in the difficulty or limitation of the Scrum Practices. It can be interpreted, from this table, that sometimes the entire Scrum Team have to face difficulties (3 level of impact) and limitations (2 level of impact), but only one element of the Scrum Team will be the most affected.

For example (see Table 19), STP6 practice involves the entire Team, and is highly impacted by the A3 attribute (L and XL), that means that the whole team will be affected by a difficulty while using this practice, but only the Development Team would be in charge of facing the difficulty. Another example, SEP16 involves the Development Team and the Scrum Master roles, and it has a medium impact by the A1 attribute (M), in this case the DT and SM will be affected by a limitation while using this practice, but the SM will be the more affected of two because this practice is in charge of this role.

					Project Level Context Attributes																				
Practice		**Scrum Team**			**A1**			**A2**		**A3**				**A4**		**A5**			**A6**		**A7**	**A8**	**A9**	**A10**	**A11**
ID	**#**	**PO**	**DT**	**SM**	**S**	**M**	**L**	**S**	**U**	**S**	**M**	**L**	**XL**	**L**	**H**	**L**	**M**	**H**	**L**	**H**					
STP	1	*			1	1	2	1	3	1	2	3	3	1	1	1	1	1	1	1	1	1	1	2	1
	2	*			1	1	1	1	2	1	2	3	3	1	1	1	1	1	1	2	1	1	1	2	1
	3	*			1	1	1	1	2	1	2	3	3	1	2	1	1	1	1	2	2	1	1	1	1
	4	*			1	1	2	1	2	1	2	3	3	1	1	1	1	1	1	1	1	1	2	1	1
	5	*			1	1	2	1	2	1	2	3	3	1	1	1	1	1	1	1	1	1	2	1	1
	6	*	**	*	1	1	1	1	2	1	2	3	3	1	2	1	1	2	1	3	2	1	2	1	1
	7	*	**	*	1	1	1	1	3	1	2	3	3	1	3	1	2	2	1	2	2	1	2	1	1
	8		*		1	1	2	1	1	1	1	3	3	1	1	1	1	1	1	2	2	1	2	1	1
	9	*		**	1	2	3	1	2	1	1	3	3	1	2	1	1	2	1	2	2	1	1	1	1
	10	*		**	1	2	2	1	2	1	1	2	2	1	2	1	1	1	1	2	1	2	1	1	1
	11	*		**	1	1	2	1	2	1	1	2	2	1	1	1	1	1	1	2	1	1	1	1	1
	12	*		**	1	2	2	1	2	1	2	3	3	1	2	1	2	2	1	2	1	1	1	1	1
	13	*		**	1	2	3	1	2	1	2	3	3	1	3	1	2	2	1	3	1	2	2	2	1
	14	*	*	**	1	2	3	1	2	1	2	3	3	1	3	1	1	1	1	3	2	1	2	2	1
	15		*	**	1	2	3	1	2	1	2	3	3	1	2	1	1	1	1	3	1	1	2	1	1

	16		*	**	1	1	2	1	2	1	1	2	2	1	2	1	1	1	1	3	2	2	2	2	2
	17		*	**	1	1	2	1	3	1	2	3	3	1	3	1	1	1	1	3	2	2	2	2	1
	18			*	1	1	2	1	2	1	2	3	3	1	1	1	1	1	1	1	2	1	2	1	1
	19			*	1	2	3	1	2	1	2	3	3	1	2	1	1	1	1	2	2	1	2	1	1
	20			*	1	2	3	1	2	1	2	3	3	1	1	1	2	2	1	2	2	2	2	2	1
	21	*	*	**	1	2	3	1	1	1	2	3	3	1	2	1	1	1	1	1	1	1	1	1	1
SEP	1	**	*	*	1	1	2	1	1	1	2	3	3	1	1	1	1	1	1	1	1	1	1	1	1
	2	*	**	*	1	1	1	1	2	1	1	2	2	1	2	1	1	2	1	2	1	1	2	1	1
	3		*		1	1	2	1	1	1	1	2	2	1	2	1	2	2	1	2	1	1	2	1	1
	4	**	*	*	1	1	2	1	2	1	2	3	3	1	2	1	1	2	1	2	1	1	1	1	1
	5	**	*	*	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	1	2	1	2	1	1
	6	*	*	**	1	2	3	1	3	1	2	3	3	1	3	1	1	1	1	3	2	1	2	2	1
	7	**	*	*	1	2	2	1	3	1	2	3	3	1	2	1	1	2	1	2	1	1	2	2	1
	8		*		1	2	3	1	3	1	2	3	3	1	2	1	2	2	1	2	1	1	1	2	1
	9		*		1	1	2	1	2	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	1
	10	*	**		1	2	3	1	1	1	2	3	3	1	3	1	1	1	1	3	1	2	2	1	1
	11		*		1	1	1	1	2	1	2	3	3	1	3	1	1	1	1	2	1	2	2	1	1
	12		**	*	1	2	3	1	1	1	2	3	3	1	1	1	1	1	1	2	1	1	2	1	1
	13		*		1	2	3	1	2	1	2	3	3	1	2	1	1	1	1	2	1	1	1	2	1

	14		*		1	2	2	1	1	1	2	3	3	1	2	1	1	1	1	2	1	1	1	2	1
	15	*	**	*	1	2	3	1	1	1	2	2	2	1	2	1	1	1	1	2	1	1	1	2	1
	16		*	**	1	2	3	1	1	1	2	3	3	1	1	1	1	1	1	1	2	1	2	1	1
	17		**	*	1	2	3	1	2	1	2	3	3	1	2	1	2	2	1	3	1	1	2	1	1
	18		*	**	1	2	3	1	2	1	2	3	3	1	1	1	1	1	1	1	1	1	1	1	1
	19	*	*	**	1	2	3	1	1	1	2	3	3	1	2	1	2	2	1	2	1	1	1	1	1
	20	**	*	*	1	2	2	1	1	1	2	3	3	1	2	1	1	1	1	2	2	1	2	2	1
	21	**	*	*	1	2	3	1	1	1	2	3	3	1	1	1	1	1	1	1	1	1	1	1	1
	22		*		1	1	2	1	1	1	2	2	3	1	1	1	1	1	1	1	1	1	1	1	1
	23		*		1	1	2	1	1	1	2	2	2	1	1	1	1	1	1	1	2	1	2	1	1
	24	*			1	1	2	1	2	1	2	3	3	1	2	1	1	1	1	2	1	1	1	2	1
	25	**	*	*	1	1	2	1	1	1	2	3	3	1	2	1	1	1	1	3	2	1	2	2	2
	26	**	*	*	1	2	3	1	2	1	2	3	3	1	3	1	1	1	1	3	1	2	2	2	2
	27	**	*	*	1	2	3	1	1	1	2	3	3	1	3	1	1	1	1	3	1	1	2	2	1
	28	**	*	*	1	1	1	1	2	1	2	3	3	1	2	1	1	1	1	2	2	1	2	2	1
	29	*	**	*	1	1	2	1	1	1	2	3	3	1	1	1	1	1	1	1	1	1	1	1	1
	30	*	**	*	1	1	2	1	1	1	2	3	3	1	2	1	1	1	1	2	1	1	1	1	1
	31	*	**	*	1	1	2	1	2	1	2	3	3	1	3	1	1	1	1	2	1	1	1	1	1
	32	*	*	**	1	1	2	1	2	1	2	3	3	1	2	1	1	1	1	2	1	1	1	1	1

	33	*	*	**	1	1	1	1	2	1	2	3	3	1	2	1	1	1	1	2	1	1	1	1	1
	34	**	*		1	2	3	1	1	1	2	3	3	1	2	1	1	2	1	2	2	1	2	2	1
	35		*		1	2	3	1	1	1	2	3	3	1	2	1	1	1	1	1	1	1	1	1	1
	36		*		1	2	3	1	2	1	2	3	3	1	2	1	2	2	1	2	2	1	2	1	1
	37		*		1	2	3	1	2	1	2	3	3	1	3	1	1	1	1	3	2	1	2	2	1
	38		*		1	1	2	1	1	1	2	3	3	1	2	1	1	1	1	2	2	1	2	1	1
	39		*		1	1	2	1	1	1	2	3	3	1	2	1	1	1	1	2	1	1	1	1	1

Table 19. Impact of the project level context attributes in Scrum Practices

To better understand the analyses of Table 19, make use of Table 17 and Table 18

Figure 34 summarizes the distribution of Table 19. According to the sixty Scrum Practices listed, A3 (Team Distribution) project attribute is the one that most impacts Scrum Practices. A1 (Size of the System) attribute has medium and high level of impact of the Scrum Practices when the system is medium or large. A2 (Type of Architecture) attribute only impacts Scrum Practices when the architecture of the system is unstable. A4 (Rate of Change) attribute impacts Scrum Practices when uncertainties are highly present in the development of the system. A6 (System Complexity) attribute affects Scum Practices when the complexity of the system is high.

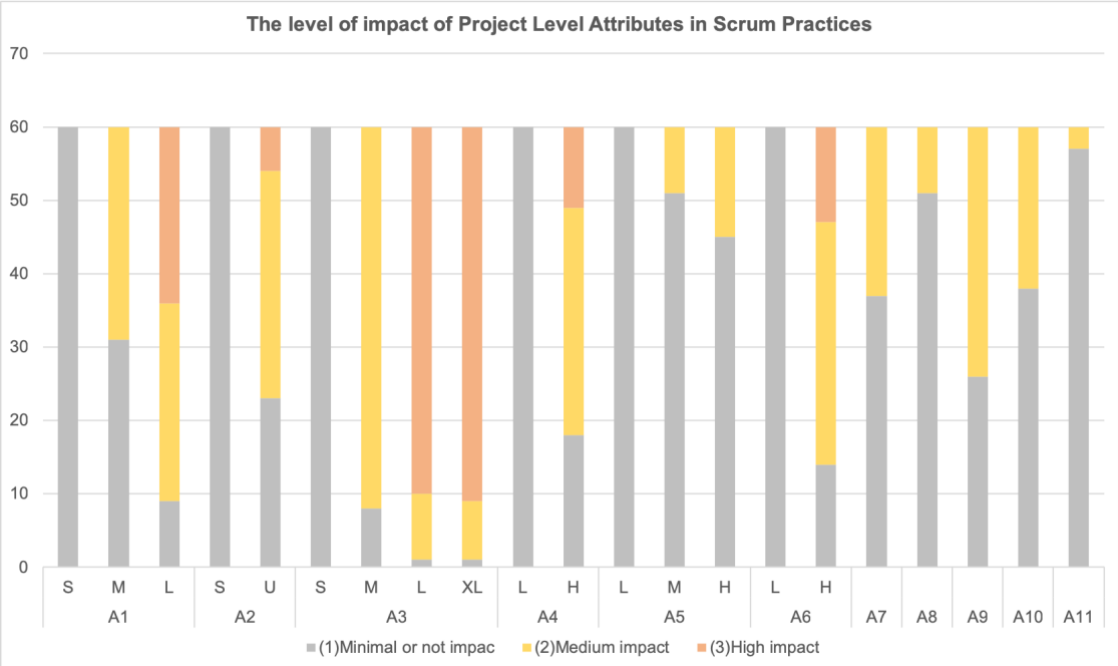

Figure 34. The level of Impact of Project Level Attributes in Scrum Practices

A5 (Criticality), A7(Business Model), A8 (Age of System), A9 (Governance), a10 (technology of the System), and A11 (Operation of the System) attributes have medium level of impact in Scrum Practices. Other interpretation of the Table 19 can be made according to the two aspects followed to list the Scrum Practices: the Scrum Team Practices (the way teams are distributed) and the Scrum Events Practices (the way work is organized).

Figure 35 introduces the level of impact of project attributes in STP practices. The A1 (Size) attribute affects the STP practices when systems are medium or large, it means that for medium systems 9/21 STP practices needs caution, or may have limitations when they are used in engineering projects, for large systems 10/21 STP practices

may have some difficulties and 7/21 STP practices needs caution while using in engineering projects.

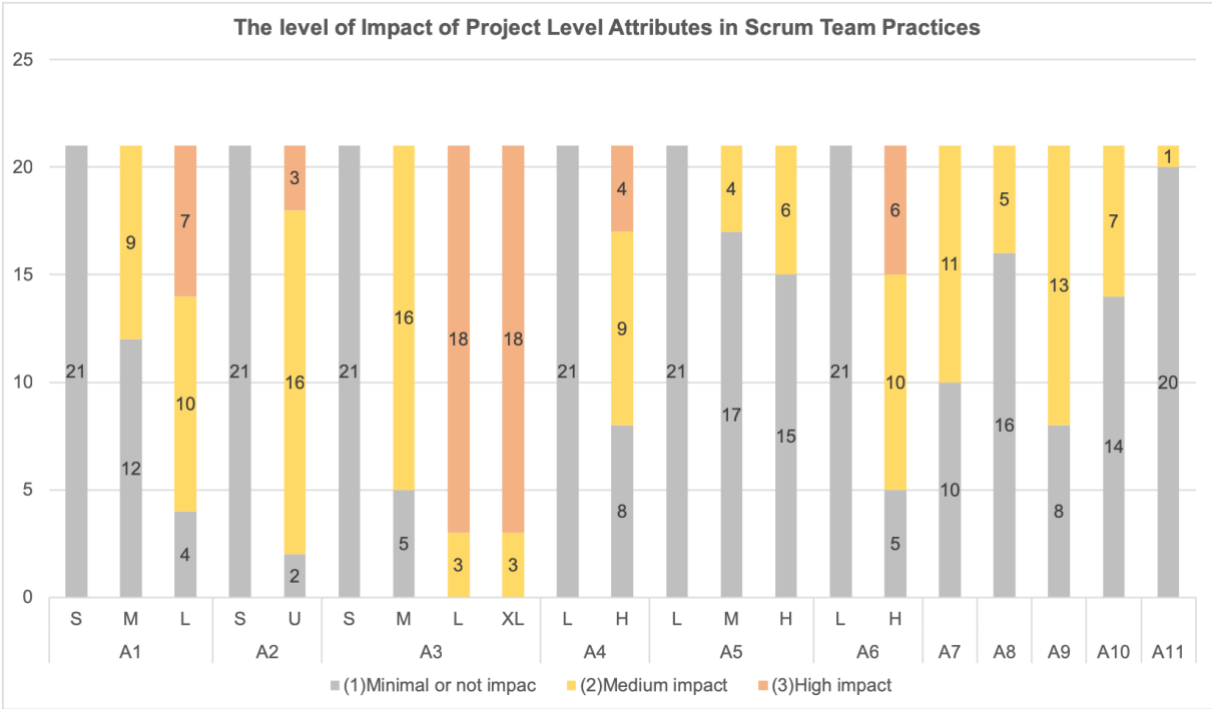

Figure 35. Project Level Context Attributes impact in Scrum Team Practices.

The A2 (Type of Architecture) attribute affects STP practices when the type of architecture is unstable (16/21 STP practices needs caution, or may have limitations, and 3/21 may have some difficulties). The A3 (Team Distribution) attribute affects STP practices when the distribution of the team is medium, large and extra-large. The A4 (Rate of Change) attribute affects STP practices when there is a high level of uncertainties in the project (9/21 practices may have limitations, and 4/21 difficulties). STP practices have a minimal or not impact by the A5 (Criticality) attribute. 10/21 STP practices needs caution and 6/21 may have some difficulties when the complexity of the system (A6) is high. STP practices globally need caution in the A7 (Business Model), A8 (Age of Systems), A9 (Governance), and A10 (Technology of the System) attributes. For the A11 (Operation of the System) attribute, there is a minimal impact or not impact in STP practices.

For SEP practices, Figure 36 summarizes the level of impact of project attributes in these practices. Thirty-nine SEP practices were listed, the distribution of the level of impact is almost similar as the STP practices. The A1 (size) attribute also affects the medium systems and large systems, that means that the way work is organized will be

affected for this attribute and others attributes like unstable architecture of the system (A2), large and extra-large team distribution (A3), high rate of change (A4), and high system complexity (A6).

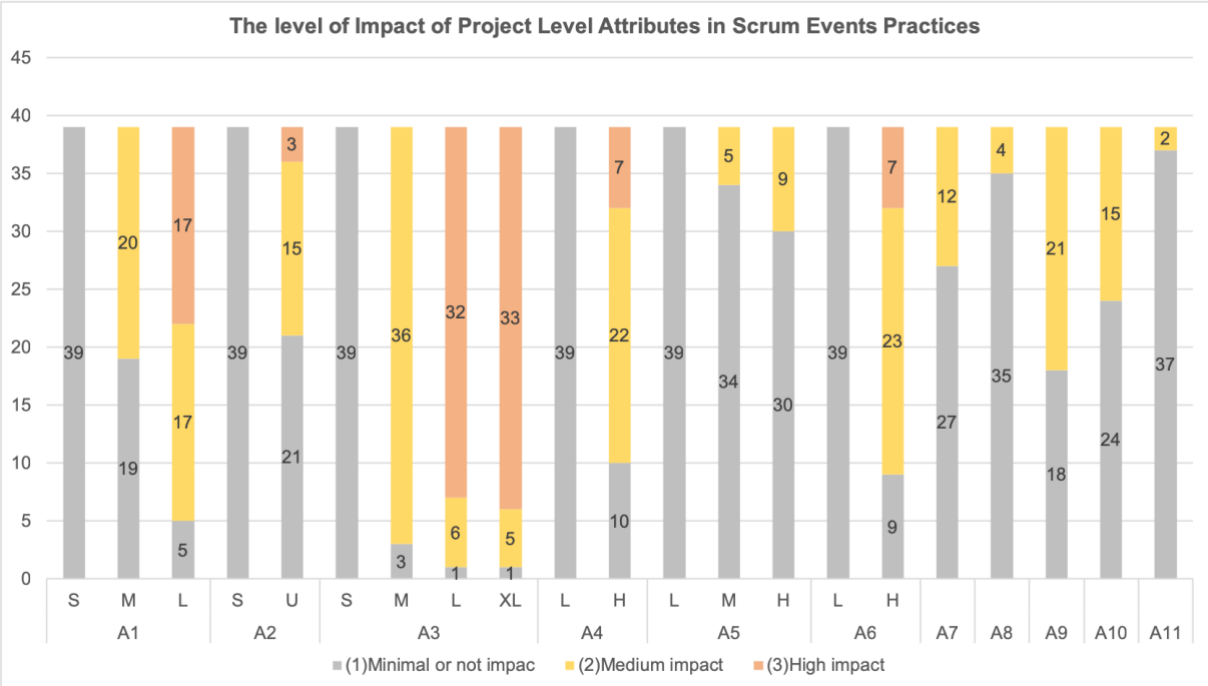

Figure 36. The Impact of Project Attributes in SEP practices

It can be concluded that the set of Scrum Practices (STP and SEP practices) have a medium and high level of impact by the same project attributes.

Another interpretation of the results of the Figure 34 can be made. Considering only the medium and high level of impact, it is possible to calculate the level of percentage that each project attribute affects the Scrum Practices. The attribute, that has minimal or not impact in Scrum Practices, will be no considered (including the criteria defined in each attribute) , the reason is because if the level of impact is minimal, or there is not level of impact, Scrum Practices can be used without problems in engineering projects. Sixty Scrum Practices were listed, this number is considered like 100%. According to the criteria defined for the impact (1-minimal or not impact, 2-medium impact, and 3- high impact) Figure 37 shows the distribution of the percentage of the medium and high impact in scrum practices.

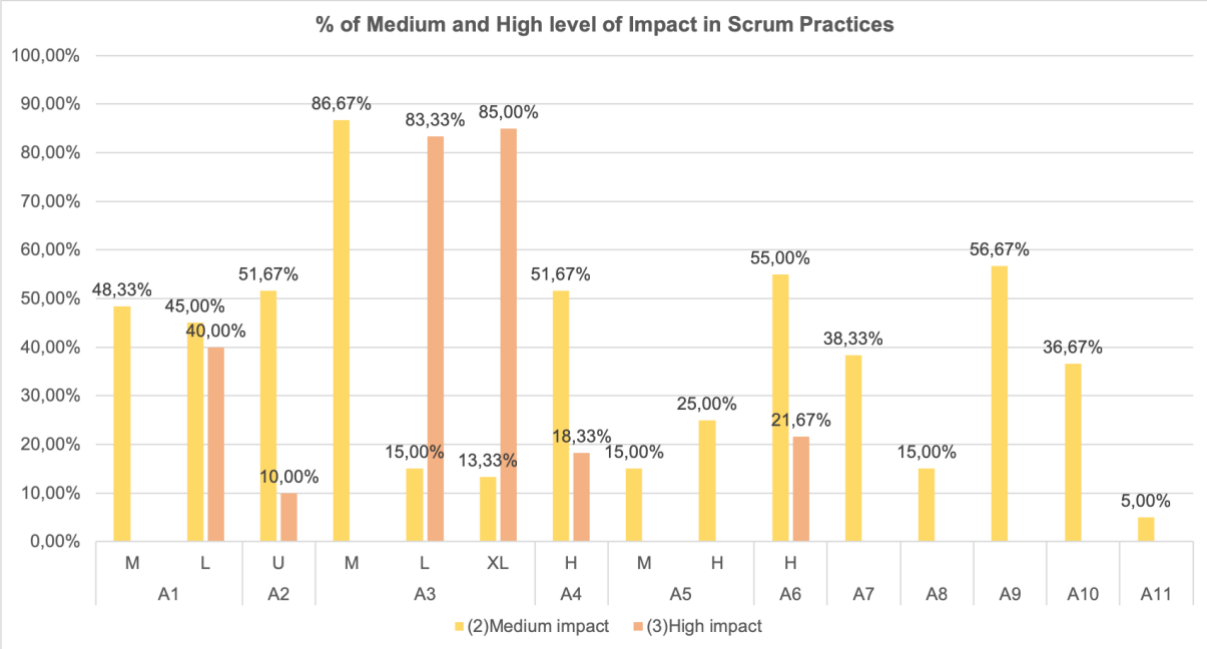

Figure 37. Percentage of medium and high impact of the project attributes in scrum practices.

The calculations were made from the amount of “2” and “3” present in each project attribute. It can be noticed, from Figure 37, that some project attributes have more than 50% medium level of impact, this is the case for, the A2 (Type of Architecture) attribute with 51,67%, A3 (Team Distribution) attribute with 86,67%, A4 (Rate of Change) attribute with 51,67%, A6 (System Complexity) attribute with 55,00%, and A9 (Governance) attribute with 56,67%. That means that Scrum Practices need caution or may have limitations when they are being deployed in systems engineering projects. Only one project attribute has more than 50% high level of impact, the A3 (Team Distribution) attribute with 83,33% for Large Team Distribution and 85,00% for Extra-Large Team Distribution attribute. That means that Scrum Practices may have some difficulties while using them in systems engineering projects.

The series of tasks, carried out previously, helped to identify how the project level context attributes, for systems engineering development, affected the Scrum Practices. By defining three levels of impact, and the use of criteria associated with the project attributes was possible to identify the percentage of impact for each project level attribute in Scum Practices. The view presented in Figure 37 will be used and analyzed more deeply to highlight the difficulties and limitations of each Scrum Practice.

III.4.4.d Identifying the difficulties for the adoption of Scrum Practices in Engineering Projects

This activity will highlight the difficulties and limitations that the Scrum Practices may have while using them in systems engineering context. The limitations are associated with the medium level of impact (2, yellow), and the difficulties with the high level of impact (3, orange). The description of the Scrum Practices and project attributes will be the base to list the possible difficulties and/or limitations of Scrum Framework in systems engineering development. The results of this activity will help to find the main project attributes that impacts the process to use the Scrum Practices in Systems Engineering projects. The project attributes with high level of impact will be analyzed first, then the project attributes with medium level of impact.

- ***High level of Impact of the Project Level Context Attributes in Scrum practices.***

Figure 38 shows the specific Scrum Practices that are highly impacted by the project attributes. The distribution presented in this figure considers the attribute and its criterion as independent each other.

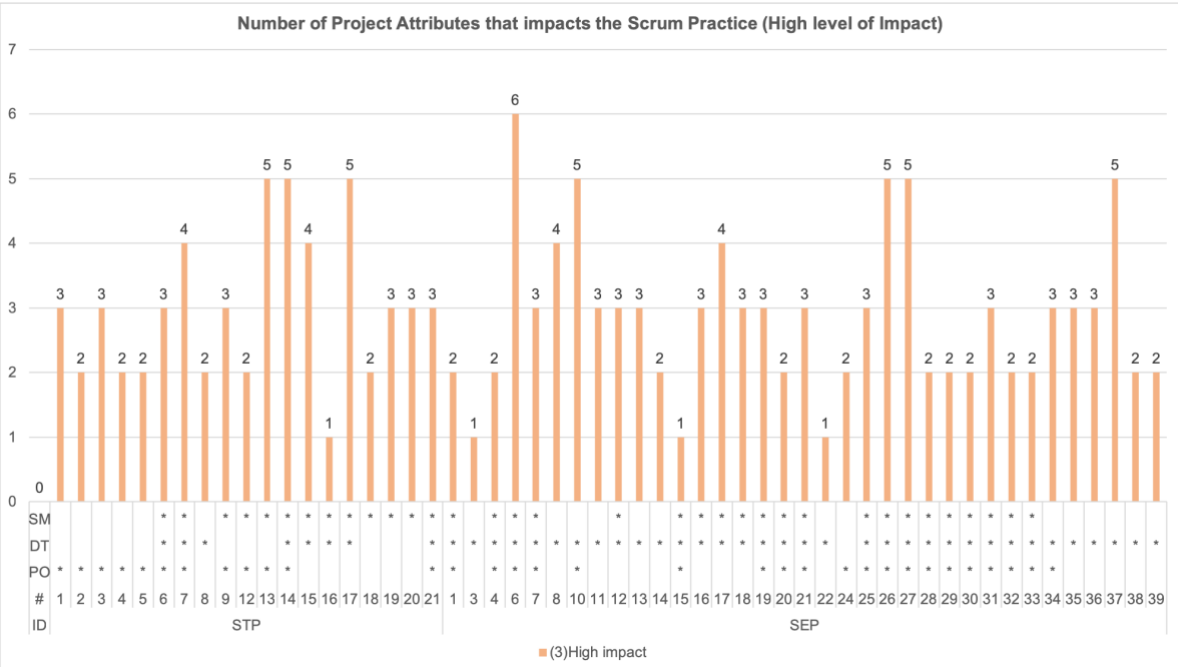

Figure 38. Number of Project Attributes that impacts Scrum Practices (High level of Impact).

For example, Size of System attribute contains Small, Medium and Large criterion, that means that instead of having one attribute, there are three. So, the total number if attributes are twenty-one. This interpretation only helps to identify the total of project attributes that impact each Scrum Practice. For example, for the STP1, two attributes affect this practice, and for SEP6, seven attributes affect this practice. The Scum Team was also considered to see which team member (PO, DT, SM) is in charge of the practice. The same logic must be followed to understand the rest of the figure.

Continuing the same vision, Figure 39 presents the percentage of high level of impact of the project attributes in the practices listed previously. It can be noticed, from Figure 39, that Scrum Practices are impacted when the Size of System is Large (with 40,00%). Small Size and Medium Size of the System criterion were also defined, but they do not have a high level of impact in Scrum Practices, that is the reason it does not appear in the figure. The same logic can be used for the rest of the attributes where different criteria were defined.

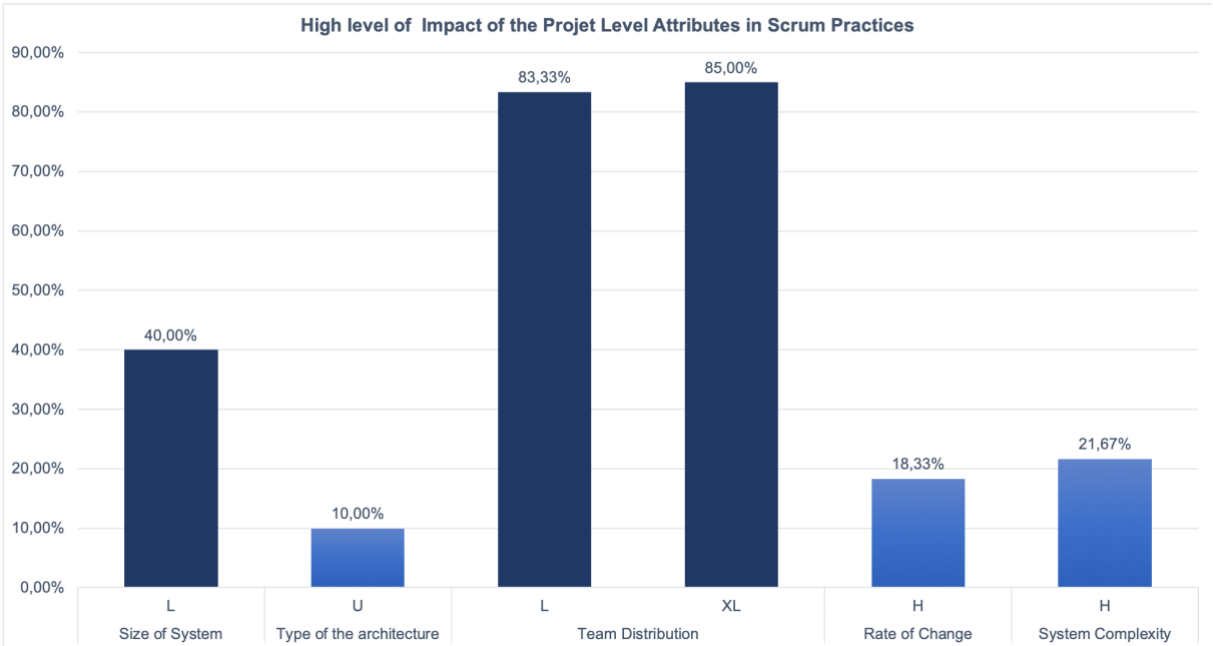

Figure 39. High level of Impact of the project attributes in Scrum Practices.

Difficulties will be identified according to the distribution of Figure 39, the attributes with more percentage of impact in Scum Practices will be analyzed first, then a decreasing way will be used to analyze the rest of the project attributes.

- *Team Distribution attribute Impact in Scrum Practices*

The Scrum Practices highly impacted by this attribute are listed in Table 20. This attribute includes two criteria Large and Extra-Large Team Distribution. 50/60 (83,33%) Scrum Practices are affected by Large – Team Distribution, and 52/60 (85,00%) by Extra Large- Team Distribution. This table also identified the role in charge of the practice, white shade for the Product Owner, light gray for the Development Team, and dark gray for the Scrum Master. The PO may have difficulties in 16 Scrum Practices, the DT in 19, and the SM in 17.

		Scrum Team			A3 – Team Distribution	
ID	**#**	**PO**	**DT**	**SM**	**L**	**XL**
STP	1	*			3	3
	2	*			3	3
	3	*			3	3
	4	*			3	3
	5	*			3	3
	6	*	**	*	3	3
	7	*	**	*	3	3
	8		*		3	3
	9	*		**	3	3
	12	*		**	3	3
	13	*		**	3	3
	14	*	*	**	3	3
	15		*	**	3	3
	17		*	**	3	3
	18			*	3	3
	19			*	3	3
	20			*	3	3
	21	*	*	**	3	3
SEP	1	**	*	*	3	3
	4	**	*	*	3	3
	6	*	*	**	3	3
	7	**	*	*	3	3
	8		*		3	3
	10	*	**		3	3
	11		*		3	3
	12		**	*	3	3
	13		*		3	3
	14		*		3	3
	16		*	**	3	3
	17		**	*	3	3
	18		*	**	3	3
	19	*	*	**	3	3
	20	**	*	*	3	3
	21	**	*	*	3	3
	22		*			3
	24	*			3	3
	25	**	*	*	3	3
	26	**	*	*	3	3

	27	**	*	*	3	3
	28	**	*	*	3	3
	29	*	**	*	3	3
	30	*	**	*	3	3
	31	*	**	*	3	3
	32	*	*	**	3	3
	33	*	*	**	3	3
	34	**	*		3	3
	35		*		3	3
	36		*		3	3
	37		*		3	3
	38		*		3	3
	39		*		3	3

Table 20. STP and SEP highly practices impacted by Team Distribution Attribute

The Scrum Guide implies three principal roles that build the Scrum Team. The Product Owner: one person who may represent the desires, and changes of a committee in the Product Backlog, The Development Team: professionals who do the work of delivering a releasable increment of the product, and The Scrum Master: the person responsible for promoting and supporting Scrum (Schwaber & Sutherland, 2017). However, engineering projects are multi-disciplinary, that means that they involve different roles.

As described in section III.4. systems engineering considers the job of systems engineers to be coordinating, tracking, and managing the engineering of the system and its subsystems (Program Management Roles), and the job to be a set of specific life-cycle tasks (Life-Cycle Roles). This roles distribution may cause some difficulties of Scrum Practices in Large and Extra Large – Team distribution. They are listed as follows:

- Large – Team Distribution

Scrum Team Practices

Practices in charge of the Product Owner (PO).

The difficulties that the product owner may have are globally related with the definition and administration of the Product Backlog, it includes:

- Difficulties to clearly expressing the PB items. The PO has to interact with all the stakeholders to define the PO, in large team distribution the PO

could lose visibility of all the stakeholders due to the large distribution of the team.

- Ordering the list of the PO items could be complicated if the stakeholders are not in the same place. Large Development Teams generate too much complexity for an empirical process to be useful (Schwaber & Sutherland, 2017). The PO may require too much coordination in Large-Team Distribution.
- PO should optimize the value of the work the DT performs, this practice may become complex, if there is only one PO in a large team distribution.
- PO should ensure visibility, transparency, and clarity of the PO to the DT. The PO may have complications in the execution of this practice, in large team distribution, due to the amount of people involved in the development of the system.
- The PO may have complications to ensure the understanding of the Product Backlog items in large team distribution. Large team distribution involves a lot of people, and the PO should reach all the levels in the DT.

Practices in charge of the Development Team (DT)

- Having more than nine members requires too much coordination (Schwaber & Sutherland, 2017). Large teams may have more than nine members, that means that the DT may face some difficulties being self-organized (how to turns Product Backlog into Increments of potentially release functionality) and cross-functional (have all competencies needed to accomplish the work without depending on others not part of the team) if the work that they have to do is not well defined.
- Optimal DT size is small enough to remain nimble and large enough to complete significant work (Schwaber & Sutherland, 2017). Scrum states that a small enough team is less than nine members, and large enough more than three members. The practice Start with a small team may have some limitations when it is adopted in large teams of engineering projects. There is a possibility that teams have more than nine members.

Practices in charge of the Scrum Master (SM)

The Scrum Master is a servant-leader for the scrum team (PO, DT, and the Organization), and helps those outside the scrum team understand which of their interactions are helpful and which are not (Schwaber & Sutherland, 2017). The SM may require too much coordination while applying the practices in its charge in large teams distribution. Too much coordination, a complex multi-disciplinary, and no centralized team could make the practices more complex to be executed, these practices are:

- *SM in service of PO:*
  - Ensure goals, scope and product domain,
  - Find techniques to effective Product Backlog management,
  - Help the DT for the good understanding of needs,
  - Understand product planning,
  - Maximize the value. To ensure the Product Owner knows how to arrange the Product Backlog to maximize value,
  - Scrum events facilitation
- *SM in service of DT:*
  - Coach the DT. In large teams, dependences are present, and sometimes the elements of the team are in different place (country, continent, etc.). The SM may have difficulties to ensure that the DT works in self-organization and cross-functionality. There is the possibility that in some organizational environments Scrum is not yet fully adopted and understood. It could affect coaching the DT, *create high-value*, and *ensure progress*, because the DT is not familiar with the Framework.
- *SM in service of the Organization:*

  The SM may have difficulties in large team distribution and the organization. The difficulties are associated with the number of instances in the organization, in large team distribution the number of instances may be also large, and the SM may require a lot a time and coordination while applying the follow practices

- Lead and coach the organization. To help the organization for the adoption of scrum,
- Plan with the stakeholders and employees of the organization. To plan and understand the scrum implementation with the organization,
- Create change. To cause change that increases the productivity of the scrum team,
- Detect differences between expected and real results. To detect artifact transparency by inspecting artifacts, sensing patterns, listening closely to what is being said, and to increase the transparency of the artifacts.

Scrum Events Practices

Practices in charge of the Product Owner (PO)

The PO may have difficulties in eleven SEP practices while using them in large team distribution. The difficulties are also associated with the number of instances, stakeholders, elements in the DT, a complex multi-disciplinary, and no centralized team. These characteristics could be present in large engineering projects. According to this, the PO may face difficulties in:

- Define the sprint goal,
- Define the sprint scope
- Define the work to be performed
- Adapt the product backlog if needed
- Consider all the stakeholders to the sprint inspection
- Deliver dates based on progress to date
- Identify valuable inputs to subsequent Sprint Planning
- Define the probable PB items for the next Sprint
- Review timeline, budget, potential capabilities and marketplace
- Product Backlog refinement

Practices in charge of the Development Team (DT)

As mentioned before, Schwaber and Sutherlad states that having more than nine members requires too much coordination and large DT generate too much complexity for an empirical process to be useful. (Schwaber & Sutherland, 2017). In addition, if the members of the team are not centralized and it is multidisciplinary. The DT may face difficulties in:

- Predict functionality,
- Start with the design of the system and the work needed,
- Clarify the selected PB,
- Explain how the work done became an increment,
- Hold daily 15-minutes meeting,
- Inspect Progress toward the Sprint Goal,
- Focus on progress toward the Sprint Goal,
- Identify impediments to development,
- Explain what have been done and what has not been done,
- Inspect the last Sprint,
- Order the items that went well,
- Create a plan for implementing improvement,
- Update Items at any time,
- Order the product backlog items,
- Make visible all the work,
- Make a highly visible sprint backlog,
- Change the Sprint Backlog if needed.

Practices in charge of the Scrum Master (SM)

The SM may have difficulties in five SEP practices while applying them in large team distribution. The difficulties that the SM could face are associated with the number of people involved for the development of the system (large team distribution in this case). The practices that may become complex to executed are:

- Consider one-month horizon for the sprint. One-month horizon may not be enough to plan all the sprint. When a Sprint's horizon is too long the definition

of what is being built may change, complexity may rise, and risk may increase (Schwaber & Sutherland, 2017),

- Improve communication,
- Promote quick decision-making,
- Inspect the Increment,
- Increase product quality by improving work process,
- Focus on inspection and adaptation

The difficulties listed to Large-Team distribution are mainly related to the coordination needed to executed Scrum Practices. Scrum Guide established the rules (the Scrum Practices) according to the distribution of the Scrum Team. These roles may be in conflict with Systems Engineering roles, and consequently with the activities present in engineering projects. There is one Product Owner and Scrum Master, and a small Development Team, that means that they are in charge of different Scrum Practices that may become complex while being used with large teams distribution. Aspects as the location of the Development Team, interdisciplinary teams with different locations are present in systems engineering projects that directly impacts the Scrum Practices while they are used in Large Team Distribution.

- Extra-Large - Team Distribution

The Extra Large – Team Distribution attribute affects the same Scrum Team and Events Practices as the Large-Team Distribution. This attribute affects all the STP practices, that means that the difficulties referred above for the Large- Team Distribution are also present in Extra Large – Team Distribution attribute. The difficulties for this attribute are also associated with the coordination of the work and people involved. That means that the Scrum Practices may become more complex for the PO, SM, and DT to be executed in extra-large teams.

Other difficulties linked to Team Distribution attribute can be:

a) The roles of systems engineering projects may not be compatible with the Scrum Team proposition,

b) Product Owner's decisions are visible in the content and ordering of the Product Backlog (Schwaber & Sutherland, 2017), so, Could the Product Owner be seen or replaced as the Project Manager?
c) Could the roles that are involved in the life-cycle task of engineering projects be considered as part of The Development Team?
d) Is it possible to define different Products Owners, and Scrum Master in large, and extra-large teams distribution?

- *Size of System attribute Impact in Scrum Practices*

The Scrum Practices highly impacted by this attribute are listed in Table 21. This attribute includes Large Size of Systems criterion. 24/60 (40,00%) Scrum Practices are impacted by Large Size of System criterion. This table also identified the role in charge of each practice affected, white shade for the Product Owner, light gray for the Development Team, and dark gray for the Scrum Master.

The PO may have difficulties in 4 Scrum Practices, the DT in 9 Scrum Practices, and the SM in 11 Scrum Practices.

		ST			A1- Size of System
ID	**#**	**PO**	**DT**	**SM**	**L**
STP	9	*		**	3
	13	*		**	3
	14	*	*	**	3
	15		*	**	3
	19			*	3
	20			*	3
	21	*	*	**	3
SEP	6	*	*	**	3
	8		*		3
	10	*	**		3
	12		**	*	3
	13		*		3
	15	*	**	*	3
	16		*	**	3
	17		**	*	3
	18		*	**	3
	19	*	*	**	3
	21	**	*	*	3
	26	**	*	*	3
	27	**	*	*	3
	34	**	*		3
	35		*		3

	36		*		3
	37		*		3

Table 21. STP and SEP practices highly impacted by Size of System Attribute

The size of the system under development will drive the size of the team involved, that means that if the system is large, the team involved is also large. Large systems tend to be multi-discipline, cross-domain, and their subsystems can be geographically dispersed.

The difficulties that may be present are described as follows:

- Large Size Systems.

Scrum Team Practices

Practices in charge of the Scrum Master (SM)

The SM may face some difficulties linked to the size of the system. The characteristics mentioned before (multi-discipline, cross-domain, and geographically dispersed) could affect the following practices and they may become complex while applying them in large size of systems.

- *SM in service of PO:*
  - Ensure goals, scope and product domain. Increasing the understanding by everyone on the Scrum Team may require a lot of coordination regarding the size of the system,
  - Maximize value. Ensuring that the PO knows how to arrange the Product Backlog may become a difficult task in cross-domain and geographically dispersed large systems.
- *SM in service of DT:*
  - Coach the DT. As mention before, the size of the system will drive the size of the team, that means that the SM may have difficulties to ensure that the DT works in self-organization and cross-functionality.
- *SM in service of DT:*
  - Plan with the stakeholders and employees of the organization. This practice may become difficult to execute because the number of elements involved in large systems.

- Create change, It could increase the productivity of the Scrum Team, but creating change when there are a lot of dependences or a lot of elements involved in the system may become complex,
- Detect differences between expected and real results. Detecting artifact transparency by inspecting artifacts, sensing patterns, listening closely to what is being said, and to increase the transparency of the artifacts, may be difficulted due to the number of elements In the system.

Scrum Event Practices

The same characteristics for large size systems affects SEP practices. The SEP practices that may be difficult to execute in this attribute are:

Practices in charge of the Product Owner (PO)

The SEP practices in charge of the PO will be executed in the Sprint Review, and they have to be executed in a four-hour meeting (Schwaber & Sutherland, 2017). This time may be not enough for large size system and also affects the following practices.

- Consider all the stakeholders to the sprint inspection. The PO may not consider all the persons involved for the increment inspection,
- Identify the most valuable thing to do next. a review of how the marketplace or potential use of the system might have changed,
- Define the probable PB items for the next Sprint. A four-hour meeting may be not enough to define the probable Product Backlog items for the next Sprint in large size systems,
- Product Backlog refinement. Add detail, estimate, and order the items in the product backlog may increase the complexity of the system.

Practices in charge of the Development Team (DT)

As well as the PO, the DT have to executed SEP practices with different time-events. Some SEP practices should be executed in eight-hours meeting (Sprint Planning) a four-hour meeting (Sprint Review), a fifteen-minutes meeting(Daily Scrum), and a

three-hour (Sprint Retrospective). This time may not be enough for large size systems, and the PO may have some difficulties to:

- Predict functionality. To forecast the functionality that will be developed during the sprint by the DT. because some parts of the system may depend on others that have not yet been realized,
- Clarify the selected PB. To determine if the DT has too much or too little work, according to the selected items from the PB,
- Hold daily 15-minutes meeting. To optimize team collaboration and performance by inspecting the work since the last daily scrum,
- Inspect Progress toward the Sprint Goal. To inspect how progress is trending toward completing the work in the sprint backlog,
- Adapt, or replan the sprint's work. To adjust the sprint work if needed,
- Update Items at any time. To make sure that the list of everything, to be needed in the product, is complete,
- Order the product backlog items. To identify the items that need to be high detailed.
- Make visible all the work. To make visible of the work that the DT identifies as necessary to meet the sprint goal,

Practices in charge of the Scrum Master (SM)

Considering the characteristics for large size systems and the time-hours defined by the Scrum Guide. The SM may have difficulties to execute the following practices.

- Consider one-month horizon for the sprint. for large systems may not be enough to covers all the work to be performed in the sprint. When a Sptint's horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase,
- Improve communication. To eliminate unnecessary meetings,
- Promote quick decision-making. To eliminate waste of time,
- Inspect the Increment. To elicit feedback and foster collaboration

Size attribute has a strong relationship with Team Distribution attribute, as it was established in its description, the size of the system will drive the size of the team

involved. This relationship can be clearly seen in Figure 39, the Size and Team Distribution attributes have the highest percentages of high impact in the Scrum Practices.

- *System Complexity Attribute Impact in Scrum Practices*

The Scrum Practices highly impacted by this attribute are listed in Table 22. 13/60 (21,67%) Scrum Practices are affected in High System Complexity. White shade indicates the Scrum Practices in charge of the Product Owner, light gray of the Development Team, and dark gray of the Scrum Master. The PO may have difficulties in 3 Scrum Practices, the DT in 4 Scrum Practices, and the SM in 6 Scrum Practices.

		Scrum Team			A6- System Complexity
ID	**#**	**PO**	**DT**	**SM**	**H**
STP	6	*	**	*	3
	13	*		**	3
	14	*	*	**	3
	15		*	**	3
	16		*	**	3
	17		*	**	3
SEP	6	*	*	**	3
	10	*	**		3
	17		**	*	3
	25	**	*	*	3
	26	**	*	*	3
	27	**	*	*	3
	37		*		3

Table 22. STP and SEP highly practices impacted by System Complexity Attribute

Systems tend to involve independently systems of system, and are developed in a multi-platform or in cross disciplinary fields, that means that fixed relationships can be found in the interactions between many parts of the system. These characterizes are related with the complexity of the system. Two criteria were defined in this attribute, but only high system complexity highly impact Scrum Practices. The difficulties that covers STP and SEP practices can be listed as follows:

Scrum Team Practices

Practices in charge of the Development Team (DT)

- The DT may face difficulties using the “Self- Organized” practice. Due to the high complexity of the system, The DT may not be able to organize and manage their own work when they fixed relationships between different parts of the system. Cross disciplinary field could also affect the ST being self-organized.

Practices in charge of the Scrum Master (SM)

- The SM may face some difficulties while applying practices as: Maximize value, Scrum events facilitation, Coach the DT, and Create high-value. These practices may become complex due to the multi-platform and cross disciplinary fields involved to the development of the system
- Ensure progress. To remove impediments to the DT’s progress, ensure progress, when there are dependences between many parts of the system, may become difficult. At some point of the project different activities can be carried out in parallel, or may have dependences between them. This aspect can be strongly related to the high complexity of the system.

Scrum Events Practices

Practices in charge of the Product Owner (PO)

The PO may have difficulties applying the following practices in systems where its complexity is high:

- Identify valuable inputs to subsequent Sprint Planning. To help the entire group collaborates on what to do next,
- Identify the most valuable thing to do next. To have a review of how the marketplace or potential use of the system might have changed,
- Define the probable Product Backlog items for the next Sprint. To defines the probable PB items for the next Sprint, once the Sprint review is finished.

These practices are present in the Sprint Planning Event, a four-hour meeting may be not enough considering a multi-platform or in cross disciplinary fields.

Practices in charge of the Development Team (DT)

- Consider one-month horizon for the sprint. If there are fixed relationships between parts of the system, one-month horizon may not be enough to accomplish the sprint.

Practices in charge of the Scrum Master (SM)

- Clarify the selected PB. To determine if the DT has too much or too little work, according to the selected items from the PB. The SM may invest a lot of time applying this practice, complex systems might require many requirements,
- Identify impediments to development and Make visible all the work can be hard to do to the SM when fixed relationships are present between many parts of the system.

- *Rate of Change attribute Impact in Scrum Practices*

Table 23 lists the Scrum Practices that are highly impacted by this attribute. 11/60 (18,33%) Scrum Practices are impacted when the Rate of Change of the system is high. White shade indicates the Scrum Practices in charge of the Product Owner, light gray of the Development Team, and dark gray of the Scrum Master. The PO may have difficulties in 2 Scrum Practices, the DT in 5 Scrum Practices, and the SM in 4 Scrum Practices.

		Scrum Team			A4- Rate of Change
ID	**#**	**PO**	**DT**	**SM**	**H**
STP	7	*	**	*	3
	13	*		**	3
	14	*	*	**	3
	17		*	**	3
SEP	6	*	*	**	3
	10	*	**		3
	11		*		3
	26	**	*	*	3
	27	**	*	*	3
	31	*	**	*	3
	37		*		3

Table 23. STP and SEP highly practices impacted by Rate of Change Attribute

Rate of Change attribute refers to the stability of business environment and how much risks, or uncertainties are being faced. This attribute could be linked to the type of architecture of the system. Traditional engineering projects uses a linear lifecycle

model. Repenning et al, states that real work is a constantly evolving mix of routine and uncertainty (Repenning et al., 2017). Routine and uncertainty are aspects present in engineering projects. The capabilities of the system are defined at the beginning.

Scrum Team Practices

Practices in charge of the Development Team (DT)

- The DT may face problems being Cross-functional. Improving the flow of work when several teams are involved in the development system may be hard face a high presence of uncertainties.

Practices in charge of the Scrum Master (SM)

The SM may face difficulties due to the high Rate of Change of the system:

- Maximize value. The SM may have problems to ensure that the PO knows how to arrange the Product Backlog due to the presence of many uncertainties and an unstable business environment,
- Scrum events facilitation. The DT might ask for many scrum events, which arise to the need to face uncertainties and risk in high rate of change in the development of the system,
- Ensure progress. The SM may not be able, or may be difficult to ensure progress and remove impediments to the DT progress, face the high rate of change.

Scrum Events Practices

Practices in charge of the Product Owner (PO)

The PO may face difficulties in the following SEP practices

- Identify the most valuable thing to do next. Thus practice may be difficult, for the PO, to perform in an unstable business environment with high level of uncertainties,

- Define the probable PB items for the next Sprint. The PO may be not able to define the probable Product Backlog items for the next Sprint face to high level of uncertainties in the system development.

Practices in charge of the Development Team (DT)

The DT may face difficulties face to many uncertainties, unstable business environment, and many risks in the development system, in the following practices.

- Clarify the selected PB. Determining the amount of work that the DT has to do can be complicated if the items of the Product Backlog are constantly changing,
- Explain how the work done became an increment. The DT may have complications to explain to the PO and the SM how the Sprint Goal become an increment if it is constantly changing,
- Create a plan for implementing improvement. The DT may not be able to create an effective plan, for the continuous improvement of the product, due to the unstable business environment and high rate of change during the development of the system.
- Making visible all the work that the DT identifies as necessary to meet the sprint goal may become a difficult practice for the DT, the sprint goal constantly change due to the gig level of uncertainties in the development of the system.

Practices in charge of the Scrum Master (SM)

- Consider one-month horizon for the sprint may not be enough time to accomplish all the elements of the sprint face to the high level of uncertainties.

- *Type of the Architecture attribute Impact in Scrum Practices*

Table 24 lists the Scrum Practices that are highly impacted by this attribute. 6/60 (10%) Scrum Practices are impacted when the type of architecture of the system is unstable. White shade indicates the Scrum Practices in charge of the Product Owner, light gray of the Development Team, and dark gray of the Scrum Master. The PO may have

difficulties in 2 Scrum Practices, the DT in 2 Scrum Practices, and the SM in 2 Scrum Practices.

		Scrum Team			A2- Type of Architecture
ID	#	PO	DT	SM	U
STP	1	*			3
	7	*	**	*	3
	17		*	**	3
SEP	6	*	*	**	3
	7	**	*	*	3
	8		*		3

Table 24. STP and SEP highly practices impacted by Type of Architecture Attribute

This attribute refers to the type of system architecture. Is the architecture of the system already in place at the start of the project? In systems engineering development the definition of the architecture requires a diverse engineering team (systems architects, systems engineers, and simulation programmers) and dependences can be found during its definition. Large and complex systems involves dependencies between the system capabilities and architectural elements (Rosser et al., 2014). Most engineering projects require a lot of architectural effort and the key architectural decisions are done during the initial concept development stages, that means that the reconfiguration of the architecture, once is fixed, is quite difficult. The dependences between system capabilities and architectural elements could be stable, but also instable.

Scrum Team Practices

- The PO may have problems to clearly define the Product Backlog items. Dependences between the systems capabilities and architectural elements of the system are present in unstable architectures. The PO may not have visibility of this dependences.
- The DT may have problems to improve the flow of the work in the development face to dependences linked to the unstable architecture of the system.
- The SM may not be able to remove the impediments due to the dependence between system capabilities and architectural elements and ensure the progress of the DT.

Scrum Events Practices

- The PO may face problems to know what can be delivered in the increment resulting from the upcoming sprint, and to how will the chosen work will get done. The dependences between the elements of the system, in unstable architectures, may have activities in parallel, it can affect the definition of the work to be performed.
- Predict functionality. The DT may not be able to forecast functionality of the system during the sprint.
- Consider one-month horizon for the sprint. As mentioned before, most engineering projects require a lot of architectural effort. The SM may have problems to define the architecture in one-month horizon.

- ***Medium level of Impact of the Project Level Context Attributes in Scrum practices.***

For medium level of impact, Figure 40 shows the specific Scrum Practices that may need caution, or may have some limitations when they are being used in engineering projects. The distribution presented in this figure also considers the attribute and its criterion as independent each other.

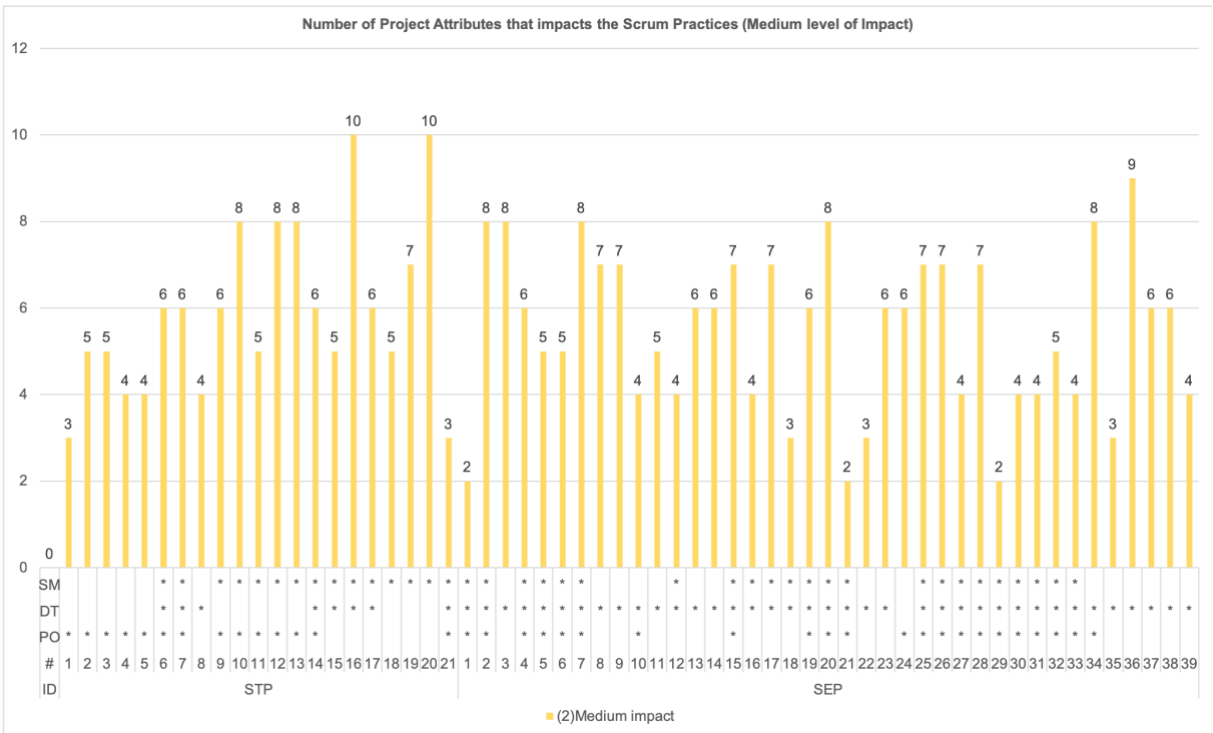

Figure 40. Number of Project Attributes that impacts the Scrum Practices (Medium level of Impact)

The distribution of Figure 40 helps to identify the roles of the Scrum Team that will face limitation in the use of each practice. For example, the Product Owner (PO) may face some limitations while executing STP1-6, and STP18-20, the Development Team (DT) while executing SEP35-39, and the Scrum Master (SM) while executing STP18-20 and SEP4-5. It is important to remember that the Scrum Practices where the entire Scrum Team is involved, only one role is responsible for the practice (this definition was made in the analysis of the level of impact in Table 19). Another example, STP 16 and STP19 could have limitations by ten attributes. Figure 41 presents the percentage of medium level of impact of the project attributes in the practices listed previously. It can be interpreted, from Figure 41, that all the project attributes, defined to contextualize systems engineering development, impact the Scrum Practices in a medium level. Four project attributes impact more than 50% of the Scrum Practices (in dark blue), six project attributes impact the Scrum Practices within the range of 15-49%, and only the Operation of the System attribute has 5% of impact in Scrum Practices.

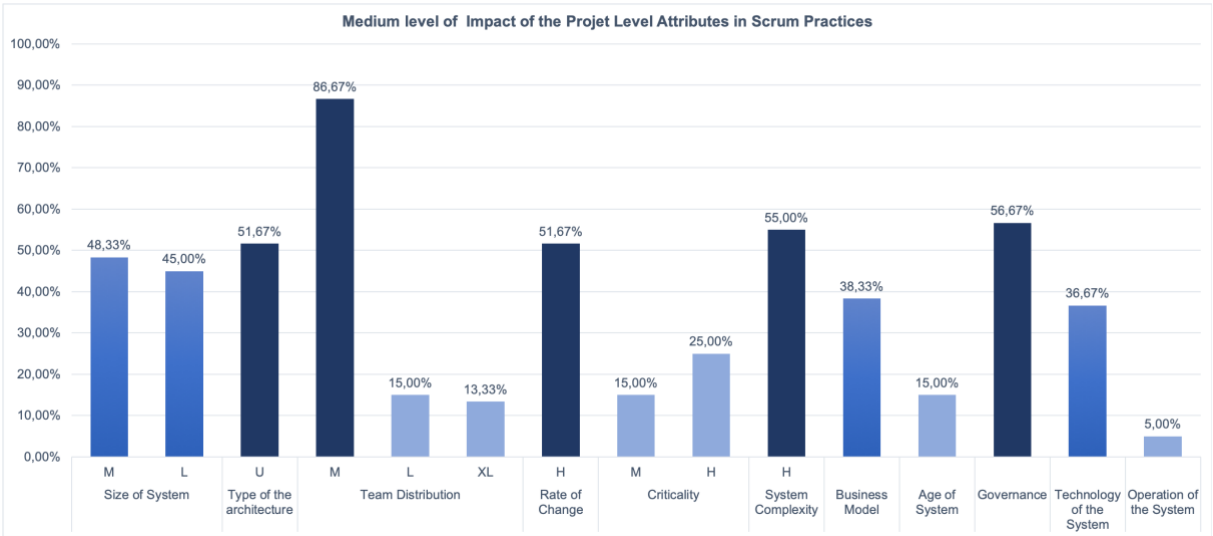

Figure 41. Medium level of Impact of the Project Attributes in Scrum Practices

Regarding the project attributes from the Figure 41. The main limitations that the Scrum Team may face while using the Scrum Practices in systems engineering projects are:

- Only one Product Owner, and one Scrum Master for all the project, when the distribution of the team is not centralized, or the governance of the project is cross-domain and geographically dispersed could limit the practices that these two roles are in charge. First, because they may need a lot of time to coordinate

all their practices, and second, because they may not have the experience to achieve their practices.

- The Development Team size is small enough (more than 3 elements) to remain nimble and large enough (less than nine members) to complete significant work within each sprint (Schwaber & Sutherland, 2017). This characteristic could limit the use of some Scrum Practices in the development of engineering projects. Medium and large systems involve medium and large teams, and it could generate too much complexity for an empirical process as Scrum.
- Self-organized and cross functional teams are characteristics that may help to better accomplish the work of the Scrum Team, but these characteristics may have limitations against systems where dependences between subsystems are present, or with activities (which were defined in the sprint) are being executed in parallel.
- Maximize the value of the Product Backlog, or create a high-value products, are practices that may have limitations. The limitations could be associated whit a lack of indicators or some measure to ensure that the Scrum Team is being delivering high-value products, or that the Product Owner has well organized the Product Backlog items.
- Events (and their associated practices) like the Sprint planning (with eight-hour for one-month sprint), Daily Scrum (with a 15-minute daily meeting), Sprint Review (with a four-hour meeting for one-month sprint), and Sprint Retrospective (with a three-hour meeting for one-month sprint) may have limitations according to the defined time to accomplished them, in the development of the systems with the criteria of each attribute.

Figure 42 shows the percentage of STP practices, in charge of each role, with medium level of impact. It can be noticed from the figure, that the SM may face most of the limitations in the STP practices.

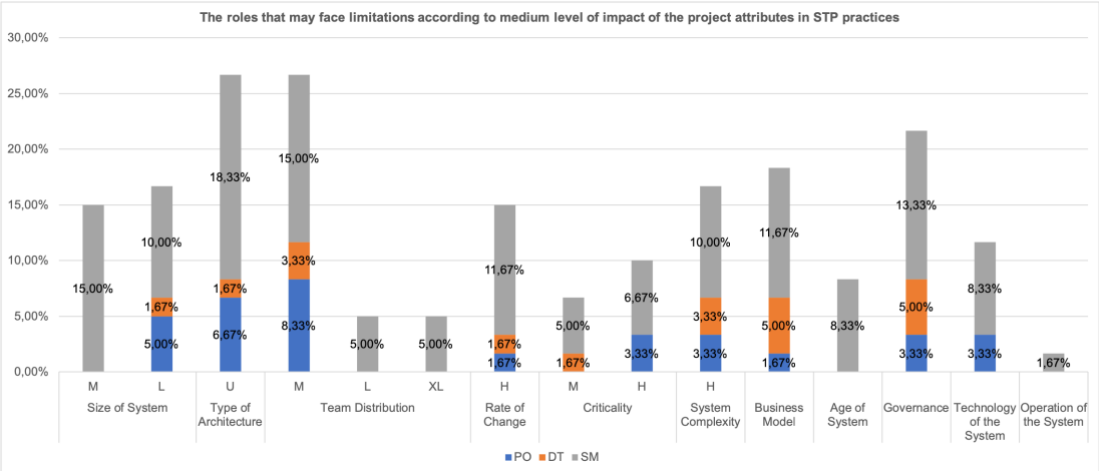

Figure 42. Percentage of STP practices in charge of each role of the Scrum Team

In the case of SEP practices, Figure 43 shows the distribution of the percentage of STP practices that each role may face limitations. For example, of the sixty defined Scrum Practices, the DT may face a few limitations while use them in engineering projects. More precisely, if the size of the system is medium or large, the DT may be face difficulties 16,67% STP practices. If the Team Distribution is Medium, the DT may face limitations in 30% of the STP practices.

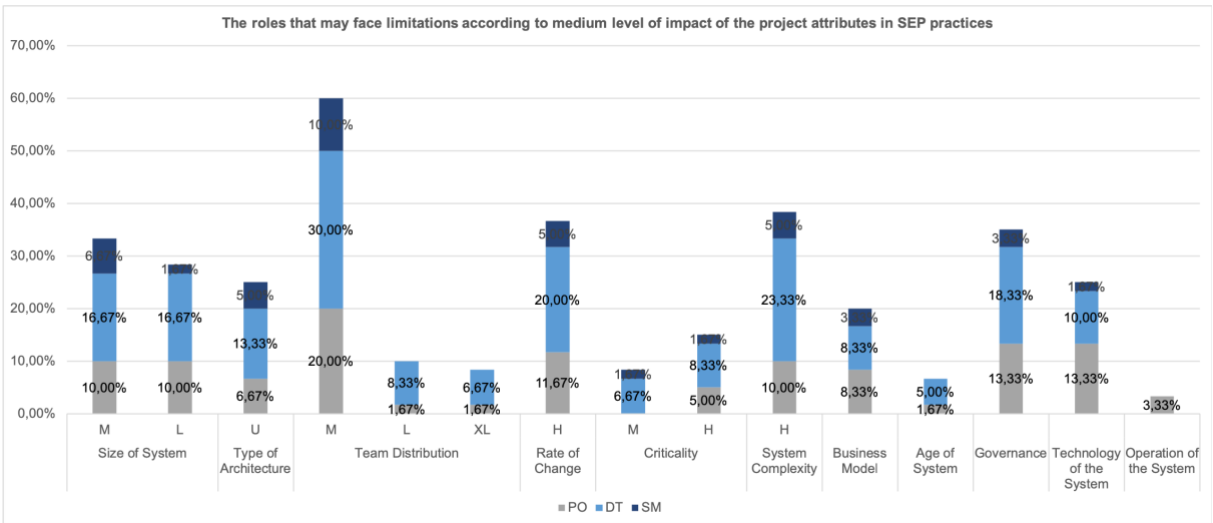

Figure 43. Percentage of SEP practices in charge of each role of the Scrum Team

Twenty-one Scrum Team Practices (STP), and thirty-nine Scrum Events Practices (SEP) were listed. Both, based on the rules defined in the Scrum Guide™. The STP practices were distributed in the way team are distributed, and the STP practices in the way work is organized. Each practice was identified with the role in charge of it, and the event in which it is executed. This scheme helped to define the best way to adopt Scrum Framework in systems engineering projects. The project level context attributes

were defined in step one, these attributes helped to contextualize systems engineering projects. The project attributes also helped to identify the limitations and difficulties present in the adoption of Scrum Practices in engineering projects. In order to identify how the Scrum Practices were affected by the project attributes, a definition of levels of impact was proposed.

Three levels of impact were defined. The (1) minimal or not level of impact, the (2) medium level of impact, and (3) high level of impact. The one (1) level of impact indicates that the practice can be used, as defined, in engineering projects. The two (2) level of impact indicates that the practice may need caution, or may have some limitations when it's being used in engineering projects, and the three (3) level of impact indicates that some difficulties may be present while using the scrum practice for engineering projects. The definition of the levels of impact mainly helped to identify the Scrum Practices that may have difficulties while using them in engineering projects. The difficulties were listed according to the project attribute, that affected the practice, and the role un charge of the practice. Five project attributes mainly affected the Scrum Practices. These attributes were Size of System, Type of the Architecture, Team Distribution, Rate of Change, and System complexity.

The Step Three helps to conclude that Scrum Practices:

- Can be used, without major problems, in projects where the size of the system, to be developed, is small, the architecture is stable (no changing requirements), the distribution of the team is small (the size of the system will drive the size of the team), the rate of change is low, and the complexity of the system is also low,
- May have limitations in projects where the size of the system, team distribution, criticality of the system, under development, are medium, and the architecture of the system is unstable with high rate of change and high system complexity,
- Can have difficulties in projects, where the size of system, under development, is large, consequently the distribution of the team will be the same (or extra-large), and the type of architecture is unstable causing high rate of change and high complexity of the system.

A set of the listed difficulties will be analyzed in the following step to propose some alternatives to solve them.

III.4.5 Step Four: Selecting the difficulties to be analyzed

The Step Four, *"Selecting the difficulties to be analyzed",* aims to propose alternatives to solve a set of difficulties identified in the step three. Figure 44 shows the list of activities of this step. First, we will identify the role with more practices in charge. This analysis will help to identify the role, which will face difficulties to execute the Scrum Practices. Then, a set of difficulties will be analyzed. The selected set of difficulties will be those that are in charge of the role with more practices. Finally, alternatives will be proposed to try to solve the difficulties.

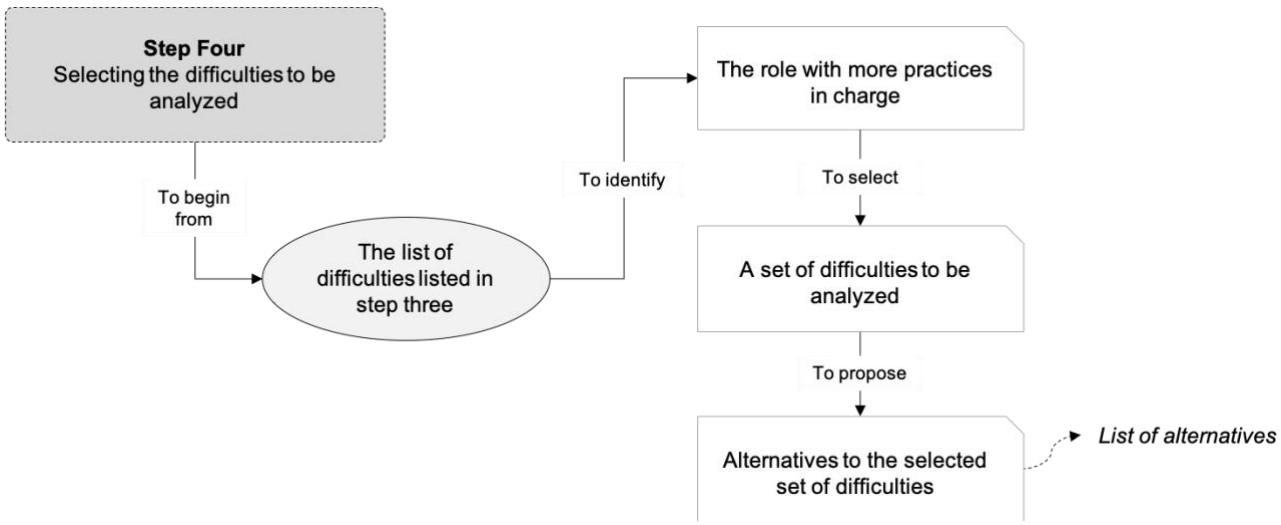

Figure 44. Structure of Step Four

III.4.5.a Identifying the distribution of the Scrum Practices by role

Table 25 resumes the distribution of the Scum Practices by role in the Scrum Team. It can be noticed, from this table, that the Development Team (DT) is in charge of more Scrum Practices (24) than the Product Owner (PO) with 17 practices, and the Scrum Master (SM) with 19 of the sixty Scrum Practices listed. It can be interpreted that the DT is the role, which will have to face more difficulties during the use of Scrum Practices in engineering projects.

	Scrum Practices	PO & DT &SM	PO & DT	PO & SM	DT & SM
PO	17	10	1	-	-
DT	24	7	1	-	2
SM	19	6	-	5	5
Total	**60**	**23**	**2**	**5**	**7**

Table 25. Distribution of the Scrum Practices by role

Table 25 also presents the interactions between the roles in the Scrum Practices. There are twenty-three practices where the PO, the DT, and the SM are involved, but each role is in charge of a specific number of the twenty-three practices (ten for PO, seven for the DT, and six for the SM). To know the distribution, of the Scrum Practices with high level of impact, Figure 45 shows the Scrum Practices highly impacted by the project attributes and the role in charge. Let's remember that five project attributes (A1- Size of System, A2- Type of Architecture, A3- Team Distribution, A4- Rate of Change, and A6- System Complexity) highly impacted the Scrum Practices. It can be noticed from this figure, that 57 Scrum Practices highly impacted are in charge of the DT, 55 of the Scrum Master, and 43 of the Product Owner. That means that the DT is the role with more highly practices impacted in charge, and consequently the role that may have more difficulties to face. Table 26 shows detailed distribution of the Figure 45.

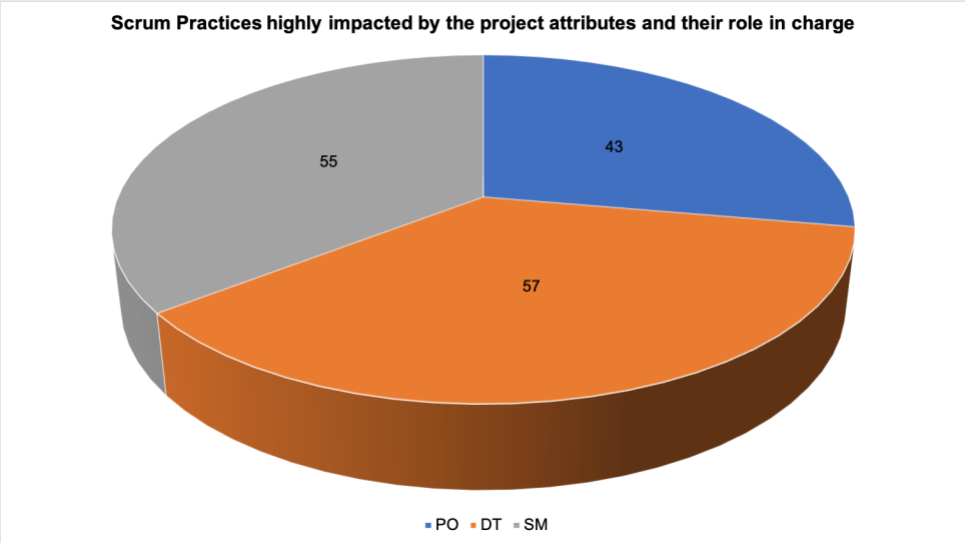

Figure 45. Distribution by role of the Scrum Practices highly impacted by the project attributes

	A1	A2	A3		A4	A6	
	L	U	L	XL	H	H	**Total**
PO	4	2	16	16	2	3	**43**
DT	9	2	18	19	5	4	**57**
SM	11	2	16	16	4	6	**55**
Total	**24**	**6**	**50**	**51**	**11**	**13**	

Table 26. Detailed distribution of the Scrum Practices highly impacted by role

The detailed distribution, presented in Table 26 does not consider the Scrum Practices in common between the five project attributes. For example, the DT is in charge of fifty-seven Scrum Practices highly impacted by five project attributes. Perhaps, between the practices of attribute A1 and A2, there are practices in common. The same situation for the SM and the PO. Table 27 lists the highly impacted common Scrum Practices among the five project attributes. The letter "C" was used to indicate in which project attribute the practice appeared. This distribution helps to identify, the number of practices that are common among the project attributes, it can be interpreted, from this table, that seven practices, in charge of the PO (white shading), are common among the five project attributes. Twelve practices, in charge of the DT (light grey shading), and twelve, in charge of the SM (dark grey shading) are common among the five project attributes.

		Scrum Team			A1	A2	A3		A4	A6
ID	**#**	**PO**	**DT**	**SM**	**L**	**U**	**L**	**XL**	**H**	**H**
STP	1	*				C	C			
	6	*	**	*			C			C
	7	*	**	*		C	C	C	C	
	9	*		**	C		C	C		
	13	*		**	C		C	C	C	C
	14	*	*	**	C		C	C	C	C
	15		*	**	C		C	C		C
	17		*	**		C	C	C	C	C
	19			*	C		C	C		
	20			*	C		C	C		
	21	*	*	**	C		C	C		
SEP	6	*	*	**	C	C	C	C	C	C
	7	**	*	*		C	C	C		
	8		*		C	C	C	C		
	10	*	**		C		C	C	C	C
	11		*				C	C	C	
	12		**	*	C		C	C		
	13		*		C		C	C		
	16		*	**	C		C	C		
	17		**	*	C		C	C		C
	18		*	**	C		C	C		
	19	*	*	**	C		C	C		
	21	**	*	*	C		C	C		
	25	**	*	*			C	C		C
	26	**	*	*	C		C	C	C	C
	27	**	*	*	C		C	C	C	C
	31	*	**	*			C	C	C	
	34	**	*		C		C	C		
	35		*		C		C	C		
	36		*		C		C	C		
	37		*		C		C	C	C	C

Table 27. Highly impacted common Scrum Practices among the five project attributes

III.4.5.b Selecting the set of difficulties to be analyzed

The analysis presented previously helped to identify the role, of the Scrum Team, who has more highly impacted practices in its charge. It can be noticed (see Table 26) that the Development Team (with fifty-seven), and the Scrum Master (with fifty-five) are the roles with more highly practices impacted in their charge. Considering these values, it is possible that the set of difficulties, to analyzed, are those in charge of the Development Team. However, it is important to consider the global mission of each one. The Development Team have to following characteristics that are implicit in the listed Scrum Practices, the DT acts collectively to determine how the achieve the sprint goal. On the other hand, the Scrum Master serves the DT, this role acts as the protector of the Development Team, and ensures that many of the practices, in charge of the Development Team, are carried out. The Scrum Master also protect the scrum process, and helps those outside the Scrum Team to understand the interactions between the Scrum Team. Based on these aspects, the set of difficulties, to be analyzed, will be the ones in charge of the Scrum Master. The objective is to help the Scrum Master to identify alternatives to face the difficulties while using the Scrum Practices in engineering projects.

The difficulties that the Scrum Master may face are related with large engineering projects. The characteristics of these projects are considered according to the project attributes defined in step one (five in this case). When the system under development is large, with unstable architecture, high rate of change and complexity, and large or extra-large team distribution, the Scrum Master will face difficulties. Aspects like: too much coordination, complex multi-disciplinary and no centralized teams, multi-discipline systems, cross-domain systems, geographically dispersed systems, unstable business environment, a lot of uncertainties, dependences between the elements of the system, many instances involved in the organization, make the following practices difficult to be executed:

Scrum Team Practices

- *SM in service of PO:*
  - Ensure goals, scope and product domain to the understanding by everyone on the Scrum Team may require a lot of coordination,
  - Find techniques to effective Product Backlog management,
  - Help the DT for the good understanding of needs, to help the Scrum Team understand the need for clear and concise product backlog items,
  - Understand product planning, to understand product planning in an empirical environment,
  - Maximize value, to ensure that the PO knows how to arrange the Product Backlog,
  - Scrum events facilitation. The DT might ask for many scrum events, which arise to the need to face uncertainties and risk in high rate of change in the development of the system,
  - Ensure progress. To remove impediments to the DT's progress, ensure progress, when there are dependences between many parts of the system, may become difficult. At some point of the project different activities can be carried out in parallel, or may have dependences between them. This aspect can be strongly related to the high complexity of the system. The SM may not be able to remove the impediments due to the dependence between system capabilities and architectural elements and ensure the progress of the DT.

- *SM in service of DT:*
  - Coach the DT. There is the possibility that in some organizational environments Scrum is not yet fully adopted and understood. It could affect coaching the DT, *create high-value*, and *ensure progress*, because the DT is not familiar with the Framework.

- *SM in service of the Organization:*
  - Lead and coach the organization, to help the organization for the adoption of scrum,

- Plan with the stakeholders and employees of the organization, to plan and understand the scrum implementation with the organization,
- Create change, to cause change that increases the productivity of the scrum team, creating change when there are a lot of dependences or a lot of elements involved in the system may become complex,
- Detect differences between expected and real results. Detecting artifact transparency by inspecting artifacts, sensing patterns, listening closely to what is being said, and to increase the transparency of the artifacts, may be difficulted due to the amount of elements in the system.

Scrum Events Practices

- Consider one-month horizon for the sprint. One-month horizon may not be enough to plan all the sprint. When a Sprint's horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase (Schwaber & Sutherland, 2017),
- Improve communication, to eliminate unnecessary meetings,
- Promote quick decision-making, to eliminate waste of time,
- Inspect the Increment, to elicit feedback and foster collaboration,
- Increase product quality by improving work process,
- Focus on inspection and adaptation,
- Clarify the selected PB. To determine if the DT has too much or too little work, according to the selected items from the PB. The SM may invest a lot of time applying this practice, complex systems might require many requirements,
- Identify impediments to development and Make visible all the work can be hard to do to the SM when fixed relationships are present between many parts of the system.

III.4.5.c Proposing alternatives to the selected set of difficulties

The practices, listed in the previous task, will be analyzed one by one to identify alternatives that may help the SM with the execution of the them. Table 28 lists the set of STP and SEP practices. These practices may present some difficulties according to the following characteristics:

- Demand for lot of coordination, face to large systems and large (or extra-large) multi-disciplinary teams,
- Multi-disciplinary teams and system elements geographically dispersed,
- Multi-discipline systems and dependences between the systems,
- Unstable environment business
- Presence of uncertainties during the development of the system, and
- Many instances involved in the organization.

Alternatives will be proposed, in each practice, in order to face the difficulties caused by the characteristics listed above.

Scrum Team Practices	Alternatives for good execution in large projects
Scrum Master (SM) in service of the Product Owner (PO)	
Ensure goals, scope and product domain to the understanding by everyone on the Scrum Team.	Some alternatives that the Scrum Master could follow are: • Identify the number of domains involved, • Assign one SM per domain involved, or per location (if needed), • Ensure that the assigned SMs understand the goals, scope and product domain, • Ask SMs to ensure goals, scope and product domain with their team, • Identify tools to better communicate with other SMs
Find techniques to effective Product Backlog management,	Some alternatives, that the SM should follow to find techniques, can be: • Create a space for discussion of the project with the PO, • Involve experts from domain that are not dominated, • Search for information on past experiences
Help the DT for the good understanding of needs, to help the Scrum Team understand the need for clear and concise product backlog items	The alternatives, that the SM should follow, are: • Centralize the doubts of each SMs assigned • Identify tools to better communicate with other SMs
Understand product planning, to understand product planning in an empirical environment	Some alternatives that the SM should follow to better execute this practice are: • Identify important knowledge from another projects experience, • Make decisions based on what is known, • Plan the product planning based on the knowledge acquired
Maximize value, to ensure that the PO knows how to arrange the Product Backlog,	The alternatives to better executed this practice are: • Participate in meetings between the client and the PO, • Create a link with systems experts, • Use tools to ensure the good arrange of the Product Backlog
Scrum events facilitation.	Some alternatives, to the SM, are: • Validate if the scrum event is really needed • Use priority levels to facilitate events

	• Ask to SMs (within the tam) to take charge of the activity
Ensure progress. To remove impediments to the DT's progress, ensure progress	Some alternatives to the SM are: • Identify and prioritizes the dependences between parts of the system, • Identify the activities that are carried out in parallel, • Activate a Daily Scrum if needed, • Use knowledge from other projects.
Scrum Master (SM) in service of the Development Team (DT)	
Coach the DT. To create high-value, and ensure progress,	Some alternatives to the SM are: • Provide the DT with elements that enable it to be self-organized and cross-functional, • Use tools where the DT can see the progress of the system, • Define the meaning of value in the system, • Identify the places where Scrum is not well known and help people understand it.
Scrum Master in service of the Organization	
Lead and coach the organization, to help the organization for the adoption of scrum	Some alternatives to better execute this practice are: • Identify the stakeholders of the organization involved in the project, • Provide information to the organization to better understand Scrum.
Plan with the stakeholders and employees of the organization, to plan and understand the scrum implementation with the organization	Some alternatives to better execute this practice are: • Present a structured plan to show how the project will be developed using Scrum, • Relaying on the graphic view of the Scrum Framework.
Create change, to cause change that increases the productivity of the scrum team,	Some alternatives to better execute this practice are: • Indicates to the organization the dependences of the system, • Identify who, in the organization, is in charge of the dependences, • Ask to the organization to make fast decisions.
Detect differences between expected and real results.	Some alternatives to better execute this practice are: • Identify who is in charge of the unexpected results, • Apply knowledge to improve the unexpected results.
Scrum Events Practices	Alternatives for good execution in large projects
Consider one-month horizon for the sprint. One-month horizon may not be enough to plan all the sprint. When a Sprint's horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase (Schwaber & Sutherland, 2017)	Some alternatives to better execute this practice are: • Use as many sprints as necessary, • Define with all the member of the Scrum Team if the duration of the sprint can change, • Identify the complexity and the risks if the duration of the sprint change, • Activate events that help to increase the complexity and risk identified.
Improve communication, to eliminate unnecessary meetings,	Some alternatives to better execute this practice are: • Use tools to document important meetings points, • Use tools to improve communication
Promote quick decision-making, to eliminate waste of time,	Some alternatives to better execute this practice are: • Use knowledge from experience to quick decision-making, • Identify the responsible of the dependence or element in the system to quick decision-making,

	• Provide information that help the Scrum Team apply quick decision-making.
Inspect the Increment, to elicit feedback and foster collaboration,	Some alternatives to better execute this practice are: • Ask the Scrum Team for help to elicit feedback, • Use tools to record feedback and foster collaboration.
Increase product quality by improving work process,	Some alternatives to better execute this practice are: • Identify tools to increase product quality, • Ask for help to the Scrum Team.
Focus on inspection and adaptation,	Some alternatives to better execute this practice are: • Identify tools to inspect and to adapt, • Ask for help to the Scrum Team.
Clarify the selected Product Backlog. To determine if the DT has too much or too little work, according to the selected items from the PB.	Some alternatives to better execute this practice are: • Use tools to see the amount of work according to the Product Backlog.
Identify impediments to development and Make visible all the work	Some alternatives to better execute this practice are: • Use tools to identify the progress of the project and to identify impediments • Use events to erase the impediments

Table 28. Alternatives for the Scrum Master to face difficulties

III.5. Conclusions

This chapter presents the presence of agility in systems engineering. First, an overview of the characteristics of Agile- Systems Engineering (A-SE) and Agile Systems-Engineering (AS-E) was introduced. A-SE is mainly focused in the rapid change of system engineering processes, for products/systems development, in an easy way. On the other hand, AS-E is mainly focused in the ability of a system to rapidly change from one state to another in an easy way. By listing the characteristics of A-SE and AS-E, we found that the agile methods, mentioned in the literature review, are focused in the management of the process for systems development. For that reason, the thesis work positioned in A-SE.

In the second section, an exploration introduce agility in systems engineering was described. Different entities and consultants were questioned to find out how they approach the subject. Then, in the third section, the standards to manage the processes of systems engineering development were introduced. The definition of systems engineering standards can be found since 1969, with the MIL-STD- 499 standard, then a number of different standards was stablished. The most common used standards are the SEBoK, the Handbook, and the ISO/IEC-15288. Different

criteria were proposed by Rui Xue to compare systems engineering standards, after the comparison, the ISO/IEC-15288 was the most suitable reference to integrate Systems Engineering and Project Management. The third section also introduced an analysis to identify the presence of agility in ISO/IEC-15288 standard. The analysis proposes a method to evaluate if the ISO/IEC 15288 could be compliant with any principles of the Agile Manifesto. The results of this analysis show that some of the technical management processes could be somehow aligned with the agile principles.

The last section of this chapter, a research methodology, to integrate agility in systems engineering, was proposed by following four steps. The step one aimed to propose a contextual model for systems engineering development. The model was defined using an existing model proposed by Philippe Kruchten. Five organizational factors and eleven project attributes made up the contextual model for engineering projects. In the step two, the project attributes were used to identify how they were present in the agile methods. The results of this analysis showed that the Scrum Framework was the most suitable to be deployed in engineering projects. A new interpretation, of the Scrum Framework, was made in the step three. The main practices of Scrum were listed and divided in Scrum Team Practices and Scrum Events Practices. Then the practices where analyzed to identify how they were impacted by the project attributes. Step three also highlighted the difficulties while using the scrum practices according to the project attributes. Finally, the step four proposed some alternatives to solve a set of difficulties.

IV. Case Studies

This chapter introduces the analysis of two case studies. Section IV.1. introduces the first case study, a small educational project aiming to design and realize a remotely controlled robot. Section IV.2. describes the second case study, a large engineering project aiming to develop an automotive application for a car engine management. Both case studies meet the characteristics identified in Chapter III. An analysis of each project will be made in order to carried out this type of projects in an agile manner, more precisely, making use of the Scrum Practices. First, we will identify how each project was managed, and the issues encountered during its development. Then we will contextualize both projects by using the contextual model and the Scrum practices (described in Chapter III), and propose improvements for them. Finally, the conclusion of this chapter is drawn in section IV.3.

Both projects are projects that have already been carried out, and the proposed improvements have not been experimented. We select these projects because they integrated different disciplines during their development, and involved hardware and software integration. The first case study is a small project; according to the results presented in section III.4.4.c, Scrum Practices can be used without any restriction in this kind of projects, and the improvements presented in the first case study can also help to train students to a basic practice of agile management. The second case study is a large project; regarding the results of section III.4.4.c, the Scrum Practices may face some limitations in large engineering projects. The improvements, for the second project, are focused in a new distribution of the team, and a new manner to face uncertainties during the development of the project.

IV.1. First project: An educational project

This section aims to propose the use of Scrum Practices in an educational project. Frist, the context, objectives, and description of the project will be listed. Then, the project will be characterized according to the contextual model for systems engineering development. Finally, the graphical view of the Scrum Framework and the Scrum

Practices presented in chapter III will be used to introduce Scrum for the management of this project.

IV.1.1 Project description and context

Every year, the Institut National des Sciences Appliquées de Toulouse (INSA Toulouse, Université de Toulouse, France) proposes a project to master students with an objective to design and realize a remotely controlled LEGO Mindstorms robot. The six to eight students project team is supervised by a professor, in charge of explaining the robot mission and helping with the management of the three-months project. The students' assignments are to define the necessary resources and contributing systems, and to organize their work and their team in order to be able to deliver the robot on time.

The project includes the development of the robot as well as the writing of a complete report, in which technical and organizational aspects have to be detailed. Students traditionally follow the V-cycle model to develop the system. The project is divided into six stages: initiation, preliminary development, principal development, robot realization, test and close. The mission of the robot is decomposed into three phases. It first has to reach the camp by autonomously following a line traced on the ground and to signal the operator that it arrived (send a 'bip'). There, an operator remotely takes the control of the robot to realize the tasks assigned to the robot. These tasks consist in dropping several wooden tokens that are embedded in a storage module in the robot, to different predefined zones. During the remotely controlled part of the mission the operator can visualize the robot environment thanks to a camera in order to avoid the different traps on the area (crevasses, low light areas, low Wi-Fi/Bluetooth areas) to drive the robot following the best route. Once the job accomplished, the robot returns to the rest zone.

From a given set of robot requirements, students have to define a logical and physical architecture for the robot, to construct it and to test it against its specifications. They first have to organize, divide and plan the project before dealing with more technical aspects (system analysis, conception, integration, tests, etc.). The project team has

three months to lead the project. The project team defines six stages to structure the project. Figure 46 shows the stages of the project.

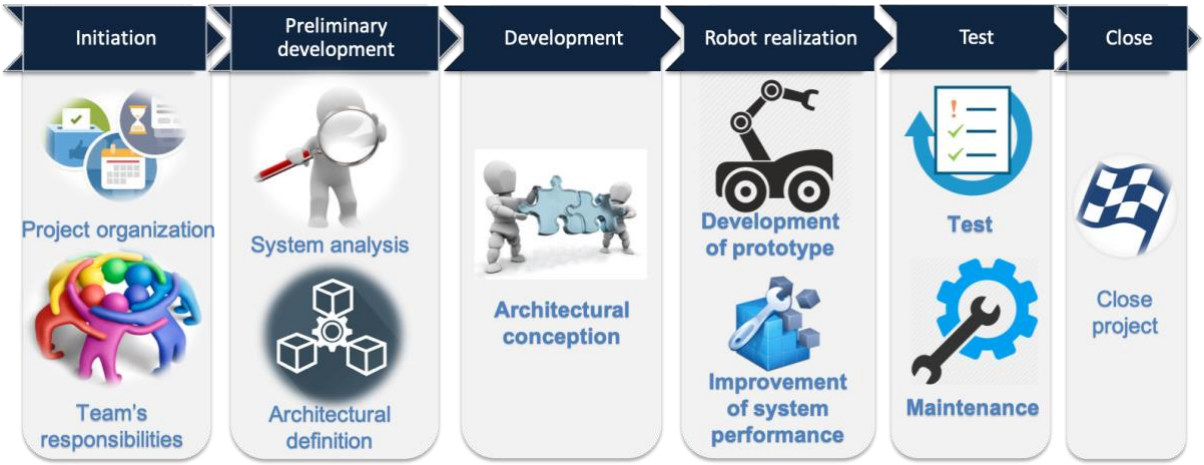

Figure 46. The six stages of the project

- **Initiation:** organization of the project group and definition of the responsibilities of each team member.
- **Preliminary development:** definition of requirements, of alternatives for the robot architecture and of test cases.
- **Principal development:** definition of the specifications and of the architectures of sub-systems; integration, validation and verification.
- **Robot realization:** development of a robot prototype with LEGO bricks, improvement of the robot performance.
- **Test:** students define test cases to verify the robot behavior and performance.
- **Close:** students have to close the project before the deadline.

To lead the project, students use traditional methods, based on a predefined schedule, with cost constraints and quality objectives. They completely plan everything from the beginning, however wisely including some margins. The robot requirements are initially given to the students, as well as some physical constraints, such as the size and shape of tokens for instance. During the development stages, changes are not expected, or even allowed. This kind of management method is adapted to a V-model development cycle.

IV.1.1.a Project Objectives

The objective of the project is to deliver an autonomous remotely supervised wheeled robot. It includes the design and construction of the robot using LEGO building blocks. The robot can either be remotely operated or move autonomously on a delimited area. This area contains two distinct zones: a ‘rest zone’ where the robot is initially located, a ‘camp’ where the robot has several tasks to accomplish.

IV.1.2 Characterizing the project according to the contextual model for systems engineering development.

To introduce the Scrum Practices in this project, we will identify the project attributes according to the contextual model for systems engineering development. Figure 47 shows the characteristics of the attributes of the project.

Size of System:	**Small**
Type of Architecture:	**Stable**
Team Distribution:	**Small**
Rate of Change:	**Low**
Criticality:	**Low**
System Complexity:	**Low**
Age of system:	**New System with fewer constraints**
Governance:	**One Project Leader and One Development Team (6 to 8 members)**
Technology of the System:	**Use of existing technology**
Operation of the system:	**It's not specified.**

Figure 47. Characteristics of the Connected Robot project

Considering the analysis presented in section III.4.4.c, we can conclude that the Scrum practices could be used for the management of the connected robot project.

IV.1.3 Introducing Scrum for the development of a Connected Robot

This section aims to experiment the deployment of the Scrum Practices in the student project. It also aims to identify the benefits of agile management in systems engineering projects. We will introduce the Scrum Practices in three project stages (see Figure 48): principal development, robot realization and test phases.

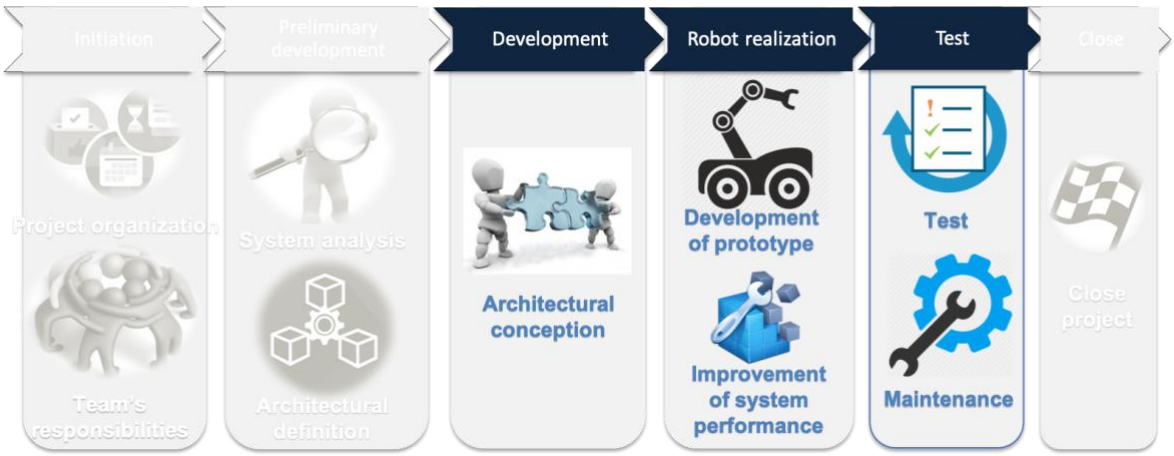

Figure 48. The three stages of the project to deploy the Scrum Practices

The robot mission is to move in a predefined environment, autonomously or remotely operated. More specifically the robot shall achieve its mission following three successive phases:

- Phase 1: The robot shall move autonomously from the 'rest zone' to the 'camp' and send an audible signal once it arrives at the 'camp'.
- Phase 2: The robot shall drop tokens in predefined zones under the operator directions, avoiding traps on the route (one crevasse, two low light areas and three low Wi-Fi/Bluetooth areas).
- Phase 3: The robot shall join the 'rest zone' under the operator commands from an operating terminal.

Before starting the Sprint Planning, students should decide who is going to be the Product Owner, the Scrum Master and who will contribute to the Development Team; the goal of this role definition is to focus on the scrum framework recommendations. One option that the students can follow to define the roles is proposed in Figure 49. Following the description of the project, the Product Owner could be the teacher in

charge of supervising the students, the Scrum Master could be one of the six members, and the Development Team the rest of the students.

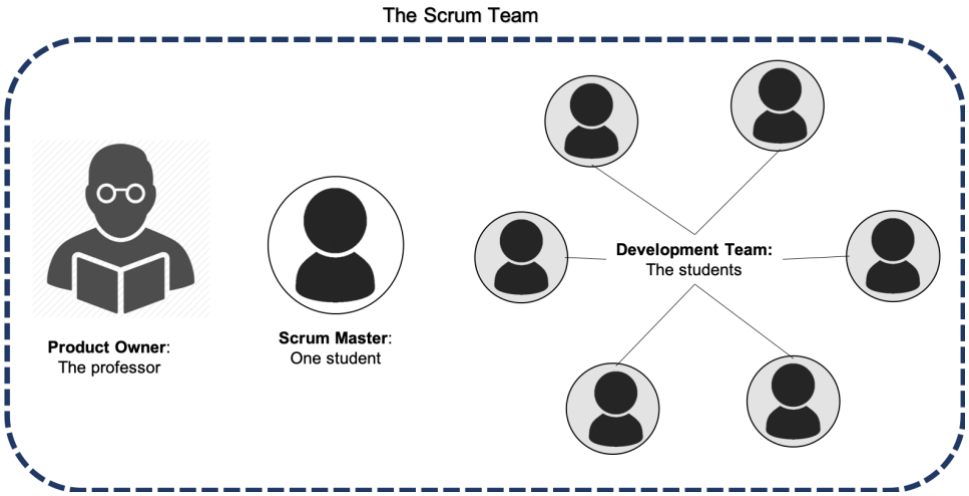

Figure 49. The Scrum Team distribution for the Connected Robot project

Students can define the global architecture of the robot using the given requirements, that allows them to identify the different interactions between stakeholders and the robot. The robot can be decomposed into three subsystems as shown in Figure 50: the Robot Structure and Motion Subsystem, the Vision Subsystem and the Human Machine Interface subsystem.

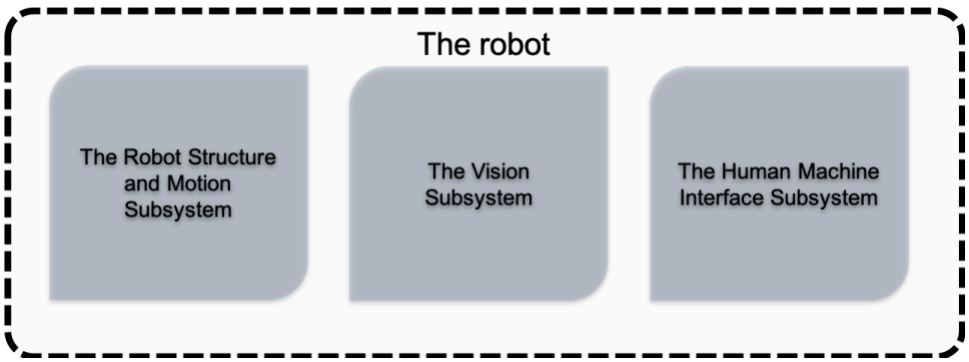

Figure 50. Global architecture of the robot

The robot structure and motion subsystem corresponds to the physical architecture of the robot with the motion subsystem; it has to answer all requirements of the specifications. The vision subsystem will allow an optimal view to the robot to accomplish its mission and the human machine interface subsystem will allow to remotely command the robot. Figure 51 shows the graphical view of the project using the Scrum Framework. It can be noticed that the three phases are considered as the

Product Backlog, and can be used to initiate the Sprint Planning. By analyzing which parts of subsystems (the robot structure and motion, the vision, and the human machine interface) are invoked at each phase of the mission, the Sprint Planning can be initiated.

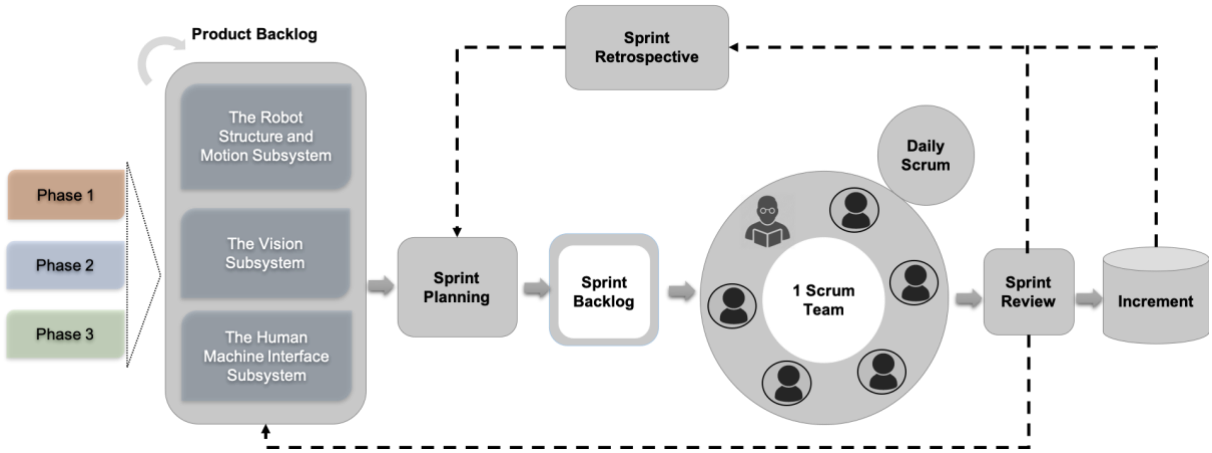

Figure 51. Graphical View of the project using Scrum

Figure 52 shows the decomposition of each subsystem of the robot into parts to define and plans the sprints on the basis of the phases of the robot mission.

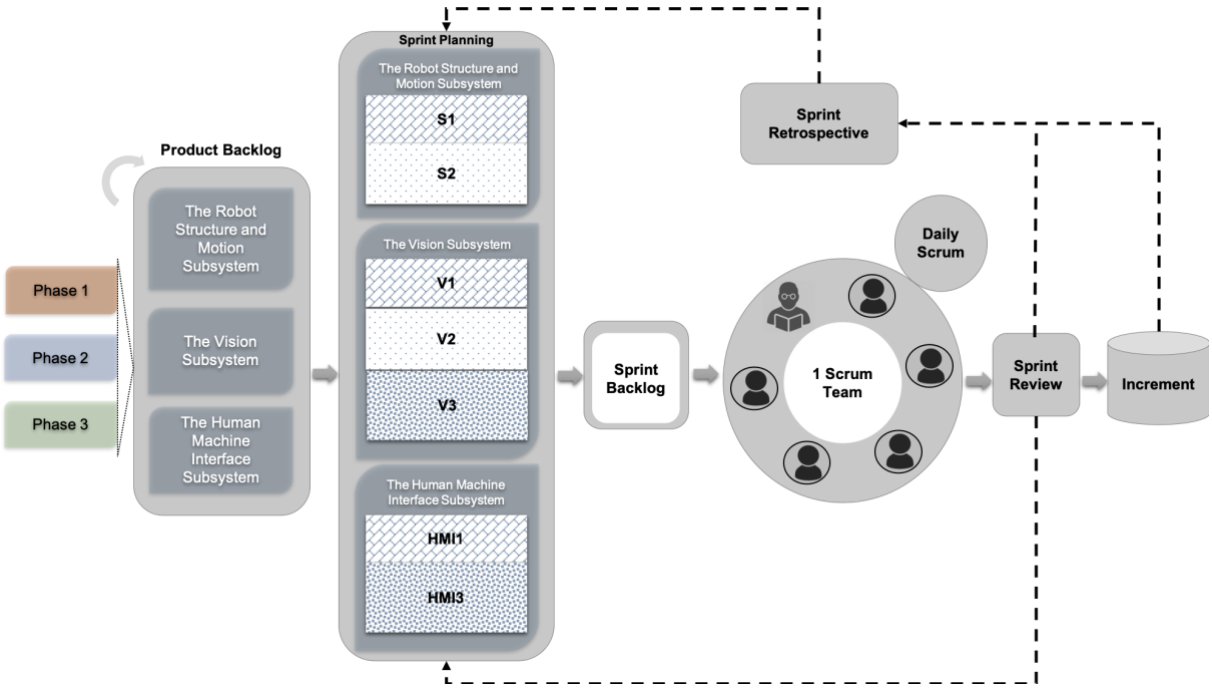

Figure 52. Decomposition of the robot subsystems into parts

It can be noticed from Figure 52 that S1 and S2 are subsystems (parts) of the robot structure and motion subsystems, V1, V2 and V3 of the vision subsystem, HMI1 and HMI3 of the human machine interface subsystem. We propose to identify which subsystem is invoked at each phase of the robot mission using the decomposition of

the subsystems. The Product Owner has to prioritize the features inside the Product Backlog, then plans the sprints with the Development Team. Our proposition aims to develop S1, V1 and HMI1 at one sprint, S2 and V2 at another sprint, and V3 and HMI3 in another sprint.

Following the Framework, the sprint Backlog can be defined as illustrated in Figure 53. Three sprints are proposed, their development has to follow the graphical view and the Scrum Practices described in section III.4.4. The sprints have to follow a daily sprint meeting and the sprint review as stablish in the scrum framework. The increment is considered as the result of the integration of each subsystem to accomplish each phase of the global mission.

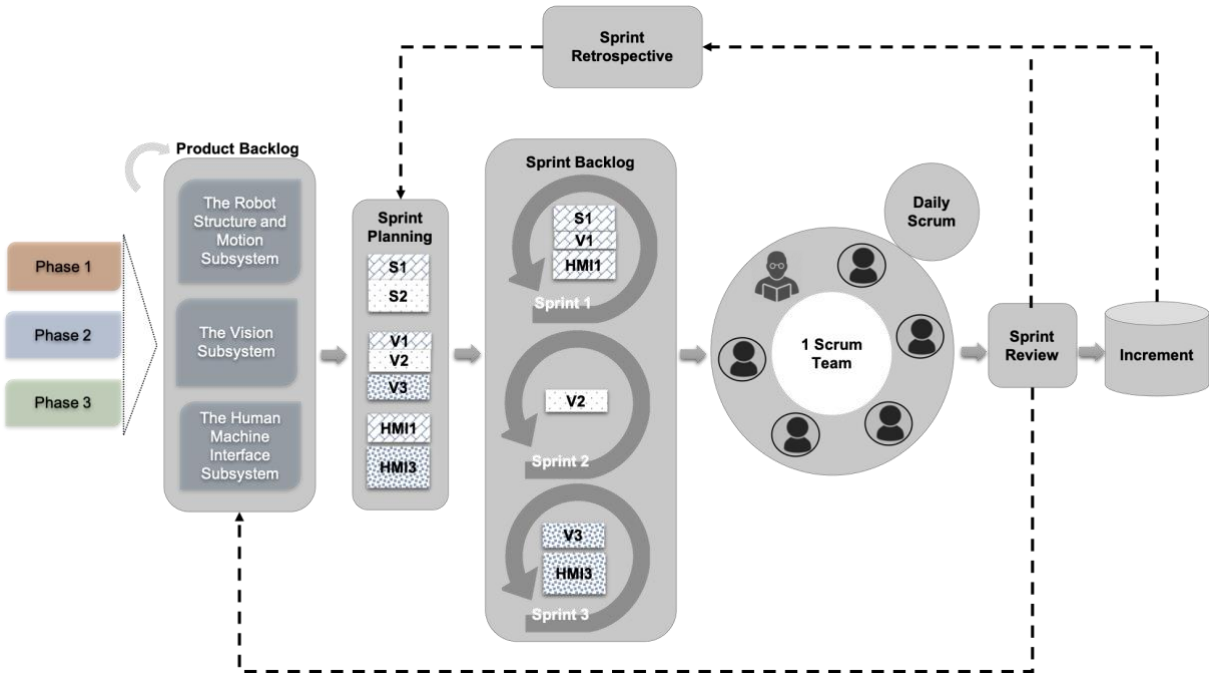

Figure 53. The definition of the Sprints

To illustrate how sprints can be detailed we define the first sprint as illustrated in Figure 54. The first artifact of this sprint is the integration of three subsystems to accomplish Phase 1. It needs the subsystem S1 (from the robot Structure motion and subsystem), V1 (from the Vision subsystem), and HMI1 (from the human machine interface subsystem).

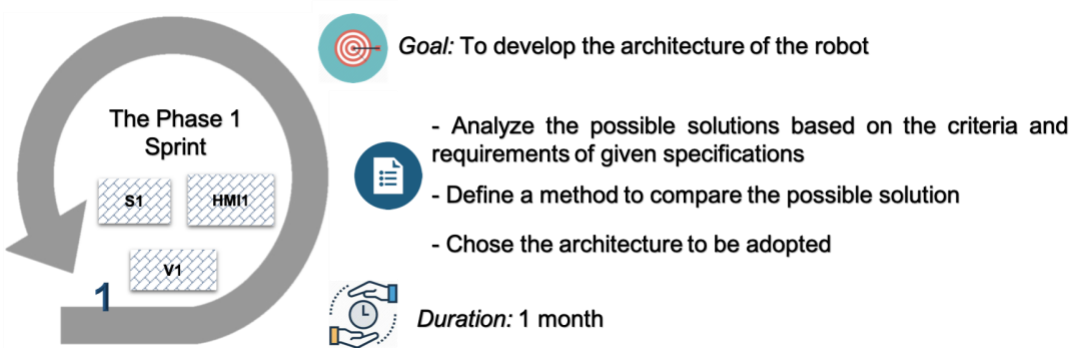

Figure 54. Structure of the first sprint

The activities can be defined considering the subsystems needed and the integration of them. The definition of the other two sprints should follow the same analysis. Students should follow the Scrum team Practices. It allows to deliver artifacts that contains hardware and software at the same time. During the sprint development, in both cases, close communication should be effective in order to cover all the requirements defined in the robot specifications.

Figure 55 illustrates the global vision of the development of the robot using Scrum Framework and the Scrum Practices.

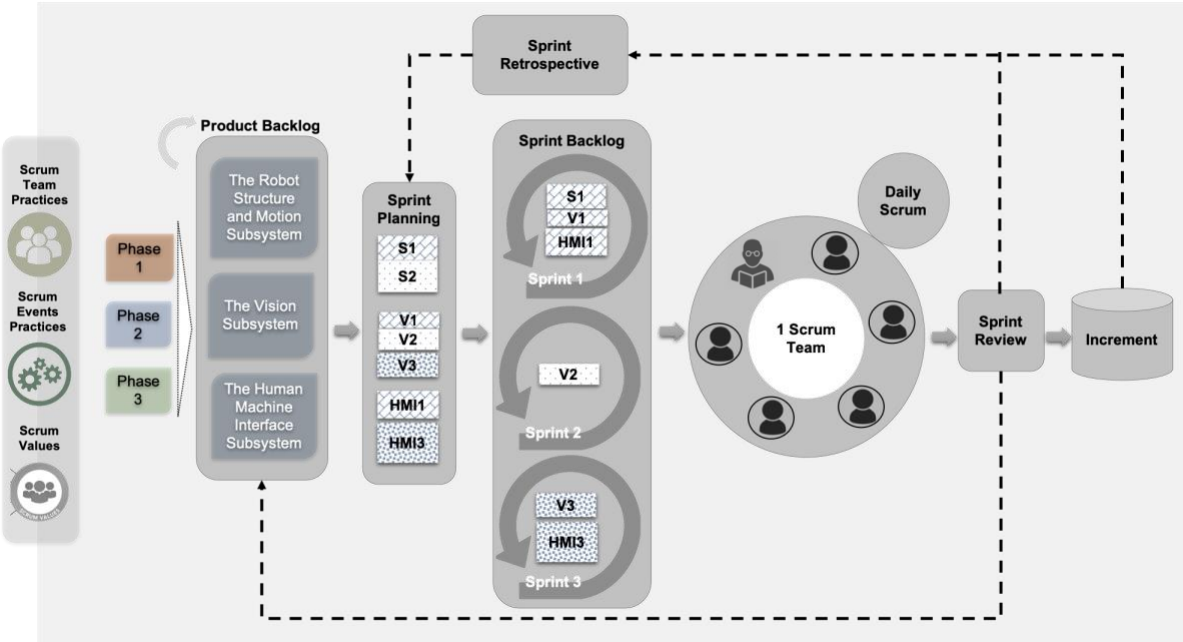

Figure 55. Developing a Connected Robot using Scrum

The implementation of Scrum, in this educational project, is proposed following the three phases that the robot has to follow to accomplish a global mission. We can find that developing the robot with Scrum could help students to deliver functional modules

in each increment and reduce time for testing it. For example, they delivered functional parts of the robot including hardware and software at the same time in each sprint.

This project was managed by using the traditional V-cycle. Students follows six stages to accomplish the project. Our proposal changes this vision and focuses on the three phases that the robot has to follow to accomplish a global mission. From the use of Scrum Practices on this education project, we can find that developing the robot in an agile manner could help students to deliver functional modules in each increment and reduce time for testing.

IV.2. Second project: An industrial project

This section aims to analyze an industrial project. The objective of the project is to develop an automotive application for engine management. We will start with the description and context of the project. Then, the project will be characterized according to the contextual model for systems engineering development. Finally, we will propose the use of Scrum to develop the automotive application.

IV.2.1 Project description and context

The case-study is on an automotive application for engine management. “An engine, or motor, is a machine designed to convert one form of energy into mechanical energy (Wikipedia, 2018). Heat engines burn a fuel to create energy which is then used to do mechanical work. Internal combustion engines are heat engines in which the combustion of a fuel occurs with an oxidizer in a combustion chamber, the expansion of the high temperature and high pressure gases move it over distance to generate mechanical work . Electric motors convert electrical energy into mechanical motion; pneumatic motors use compressed air to do mechanical work; and clockwork motors in wind-up toys use elastic energy (Wikipedia, 2018).

So, to manage the transformation of fuel into mechanical energy in a fuel economic manner and to limit as much as possible pollution, engine management systems have seen the light of day during the 1980’s and 1990’s. Such engine management systems

are the logical next step of the step-wise introduction of electronics in engine control and include more and more diagnostics, but also real-time control, taking into account sensor measurement applying control actions via actuators. Many of today's innovative internal combustion engine concepts (for both diesel and gasoline engines) could not be realized without the support of electronics and sophisticated real-time software.

In the case-study, the project was set up in a very traditional way (illustrated in Figure 56); start with the so-called "first engine", basically the prototype engine that the carmaker could share with the supplier. This engine is still far away from the final production engine, but the main characteristics in terms of combustion are there. An innovative concept is developed by a carmaker, while the electronics and associated software functions are being developed by an automotive supplier.

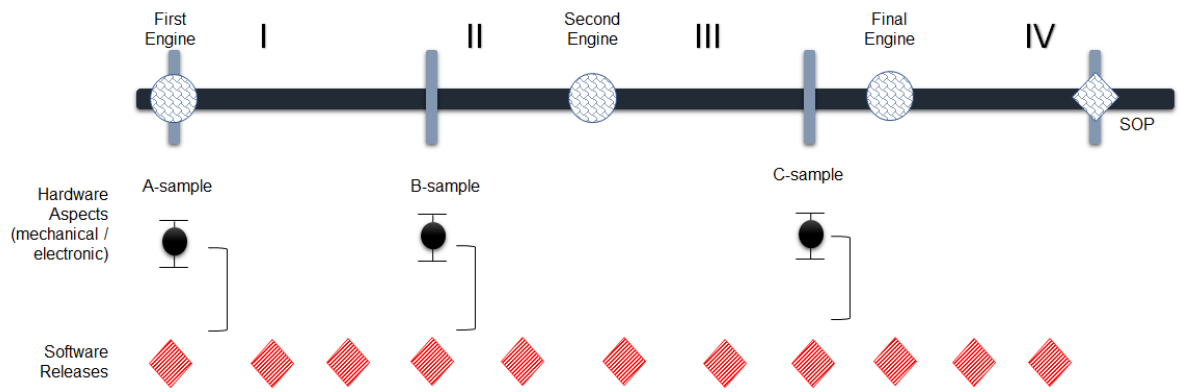

Figure 56. Original project plan until start of production

The case-study focusses on the activities of the supplier, who, in this case if high-technology innovation, had development teams working out of two countries, France and Germany. The project team itself was based in France, but the generic developments (developments that can be useful and can be applied also for other engines and other carmakers) were spread over both countries. Important to note is that the software functions allow, as much as possible, for what is referred to as "calibration", the identification of the best possible parameter values for the software to best manage the engine.

It can be noticed from Figure 56 that three samples (A-sample, B-sample, and C-sample) were proposed. The so-called A-sample, hardware level is used (mechanics and electronics) and associated software, which is still a very rudimentary version of

what will go in production at a later stage. This first engine allows for engine dyno tests and some first vehicle tests, and at the time 3 software releases were planned for this engine level. Then, a B-sample hardware level is planned, which comes closer to the real hardware. As some of the function allocations may have changed, pin-attributions for the connector, etc., the next level of SW can only be used from this hardware level onwards. This also applies for the calibration file that is intimately linked to the software level. The so-called “second engine” that comes and uses this software and hardware level. Calibration work, done on the first engine, needs to be redone to ensure the best possible parameters for this engine, and the more and more sophisticated software functions need to come as close as possible to the expected engine behavior. Over time, the final C-sample hardware level arrives, with the (more or less) final mechanical and electronic hardware. This was planned to take place just before the so-called “final engine” arrives, which engine is as good as production-ready. Again, calibration needs to be re-checked and if necessary redone, so to ensure that best performance can be obtained. Final software updates accompany the last stage of the project that often involve finalizing diagnostics, ensuring proper links to garage tools, etc.

In the original planning, the software updates were planned to be done every 3 to 4 months, with the main functionalities defined right from the start, as well as the hardware definition levels. The software functions targeted for the project were a mix of project specific solutions and generic solutions (see Figure 57; blue shaded elements were project-specific, yellow-blue generic). In Figure 56, the different releases are shown.

The governance of the project is illustrated in Figure 57. The project has a Project Leader (PL), System Project Leader, Software Project Leader, Mechanical Project Leader, and a Calibration Pilot. The PL was in charge of the project, and was the interface between the Business Responsible and the System PL, Software PL, Mechanical PL, and Calibration Pilot. The System PL handled generic and specific solutions. The Software PL was in charge of the Software team, this team handled specific project solutions and integrates engineering solutions. The Mechanical PL handled mechanical aspects of the project, and the Calibration Pilot managed the right calibration values for software.

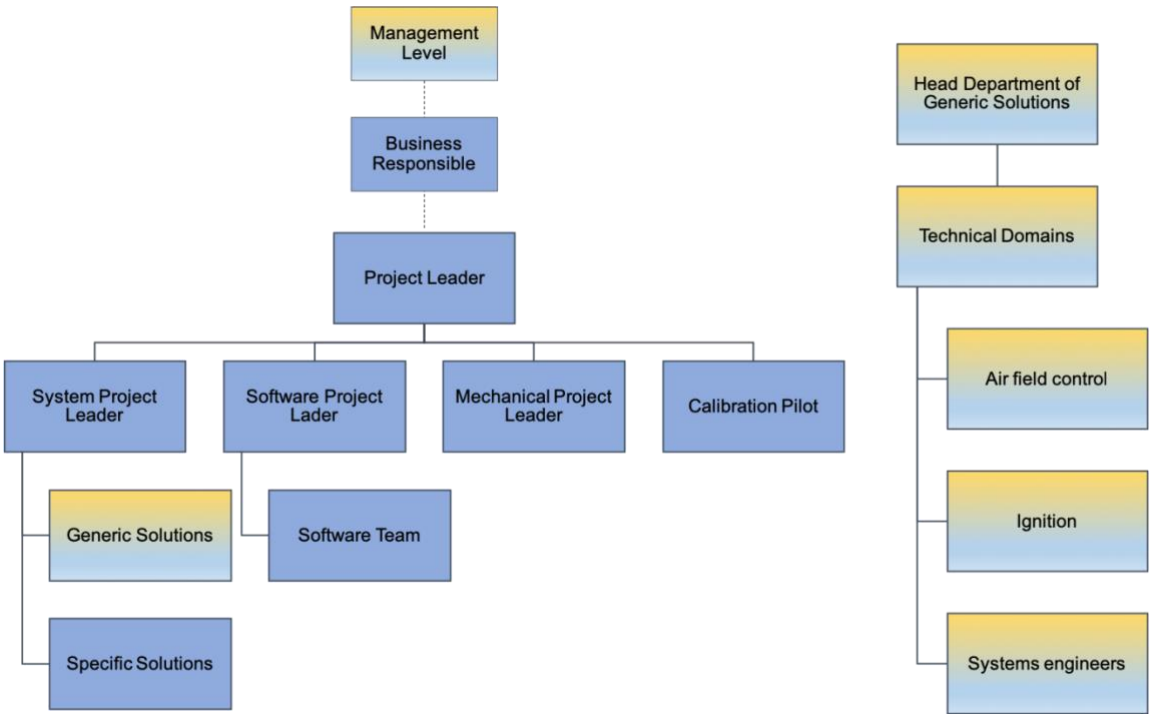

Figure 57. Governance of the project

Table 29 details the governance of the project. From Table 29, an important difference can be seen as to the number of software and system engineers working on the project, compared to the number of electronics and mechanical engineers. Indeed, the work for mechanical and electronics engineers is very focused over time and these resources are very frequently shared with other projects.

Project Duration:	Four years
Phases of the project:	Four phases
Locations of the project:	France and Germany
Release of functionality:	Every 3 to 4 months
Number of systems engineers:	20 full time + 20 part time
Number of electronical engineers:	0,4 full Time
Number of mechanical engineers:	0,3 full Time
Number of software engineers:	10-14 full time + 20 part time
Number of calibration pilots:	10 engineers

Table 29. Details of the project hypotheses from the start

Several issues happened to this project that in the end impacted very heavily the way the project took place. These issues are listed as follows:

IV.2.1.a Software release rhythm

Ten software releases were originally planned throughout the project until start of production (SOP), as mentioned roughly every 3 to 4 months. The problem for the carmaker was that the concept for the engine was so innovative that this rhythm turned out to be too slow. The carmaker started to ask for mock-ups, sort of rapid prototyping solutions based on existing software releases. Figure 58 illustrated the introduction of mock-ups in software development paths.

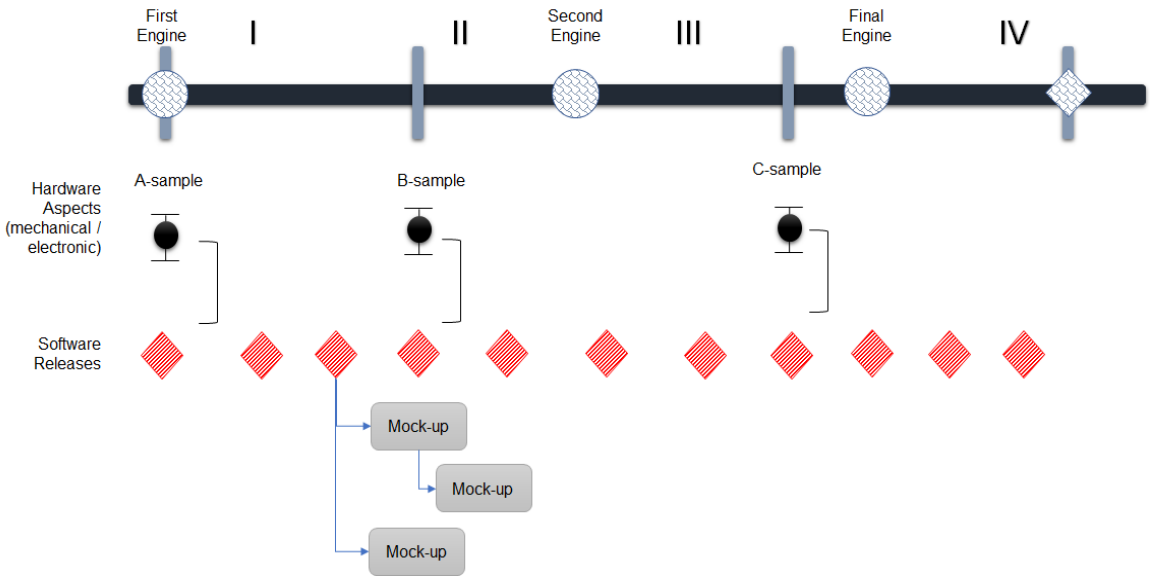

Figure 58. Introduction of mock-ups in the software development paths

Mock-ups started roughly with the third software release, still based on the A-sample. Such mock-ups could be made available in a much faster rhythm, typically once per week or even faster if necessary. The carmaker enjoyed thoroughly this way of working and very quickly a real “side-development” started to see the day of light, with mock-ups based on mock-ups, etc. The down-side of such side-developments was that on one hand this occupied pretty much the existing software development team that came under a huge pressure for the delivery of the regular software releases and on the other hand, that each of those mock-ups came with a specific calibration values file, that could not be used with other software releases. A problem arose with many different software versions in the field that were not compatible any more. The retained solution was to “align” for every next regular software release, but such integration of mock-up solutions in the regular software releases asks for special care.

IV.2.1.b Specific sensor hardware entry stage

One of the sensors to be used on this project was a brand new sensor for which there was not yet a proper hardware entry stage existed. The sensor came just out of advanced development of another supplier, and the specific functionalities necessary for the control and read-out of the measurement values could not be done with existing integrated circuits (IC). A new development was launched next to the regular electronics development, with as goal to integrate the new IC on the B-sample hardware level. Such integration requires then also updates to the basic (lower-level) software levels so to realize the communication with the sensor, and the application (higher level) software. On the A-sample, a rudimentary hardware entry stage was implemented with very basic (but limited) functionalities. The problem arose when a few months before the B-sample hardware level the new IC arrived and turned out to be not functioning as expected. Integration on the B-sample made no sense; nothing could be done with this IC.

At the same time, the design of the B-sample electronics and mechanics was already finished and the production well under way. The retained solution was to make so-called “baby-boards”, electronics boards that were made manually, soldered manually into the B-sample electronics, with a limited functionality (equal to what A-sample allowed), and with the original (basic and application) software. Figure 59 illustrates the stage of the project, where the baby-boards were introduced. The redevelopment of the IC itself was now targeting the C-sample hardware level, and the result was that all more advanced software functions that should take full benefit from the functionalities of the sensor and the IC had to be postponed until a very late stage in the project.

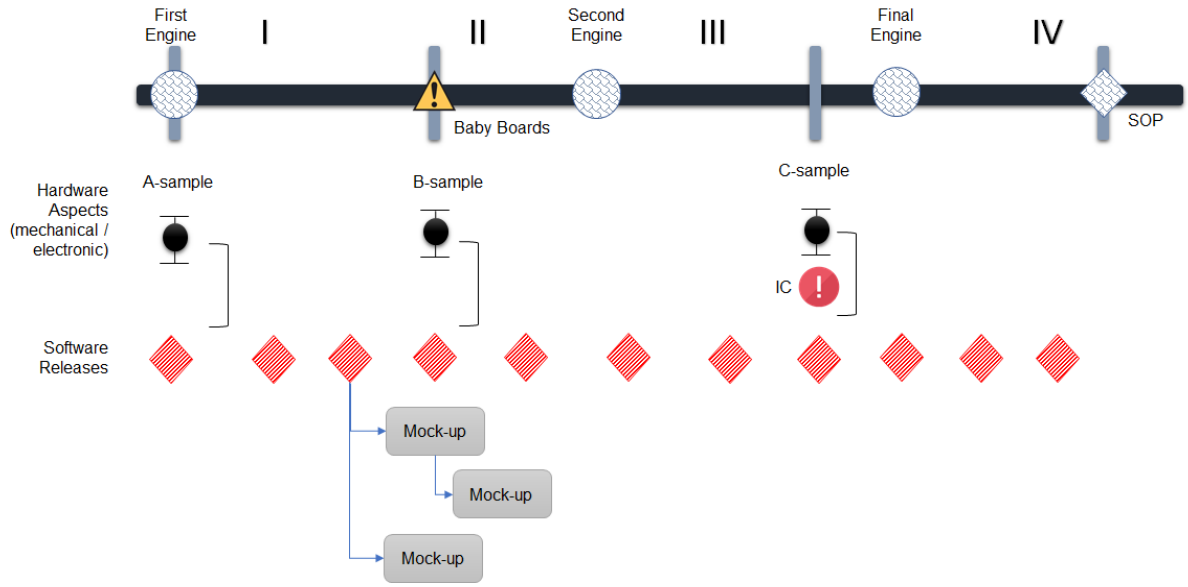

Figure 59. Impact of the not working IC

IV.2.1.c Removal of a specific sensor

The carmaker, as time went by, became more and more aware that the overall system cost (engine management system, sensors, actuators, and wiring harness) became too high to allow a reasonable price for the car. The carmaker searched for solutions to reduce the cost and in the end (around the arrival of the C-sample hardware level) focused their choice on a specific sensor that came with a high price. Their request (late into the project) was to take out this specific sensor of the project, find suitable software solutions without impacting fuel consumption or pollution, and still make the required SOP date. Figure 60 illustrates the stage where the sensor was taken out.

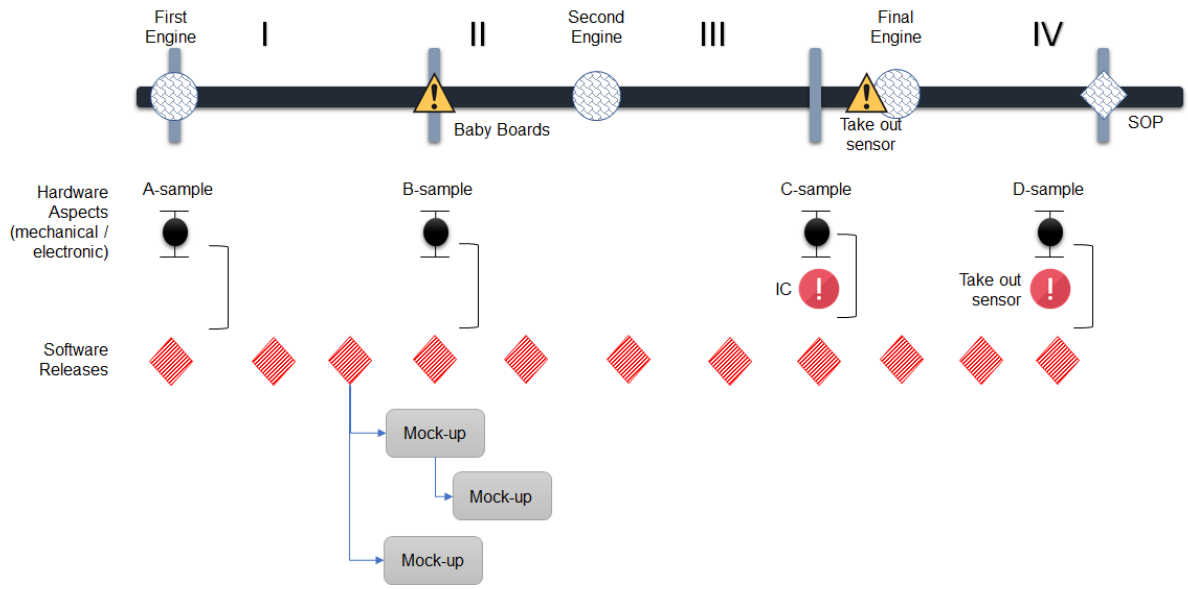

Figure 60. Impact sensor removal

This additional development in the end of the project led to a D-sample hardware level, just before SOP. The different software solutions were developed, tested and validated on the last software releases, once again based on mock-ups.

IV.2.2 Characterizing the project according to the contextual model for systems engineering development.

The project, described in the previous section, presents three global issues during its development. Before starting the Scrum Practices, to solve the issues, on this project, we will identify how the project attributes are present according to the contextual model for systems engineering development. Table 30 describes the characteristics of the project.

Size of System:	**Large**
Type of Architecture:	**Unstable**
Team Distribution:	**Large**
Rate of Change:	**High**
Criticality:	**Medium**
System Complexity:	**High**
Age of system:	**Evolution of a legacy system**
Governance:	**One Business responsible, One Project Leader, Four team leaders from different engineering disciplines. The project was carried out in two countries.**
Technology of the System:	**Creation of new elements, high innovation presence**
Operation of the system:	**It's not specified.**

Table 30. Characteristics of the project

Table 30 help to identify how the Scrum Practices are impacted according to the characteristic of this project. Considering the analysis presented in section III.4.4.c, the Scrum Practices may have limitations, or can have difficulties derived from the type of project being developed. For the difficulties, section III.4.5.c lists some alternatives to solve them. To better understand Table 30, go to the detailed analysis in section

III.4.4.c, more precisely, Figure 34 (the level of Impact of project attributes in Scrum practices).

The following section aims to introduce the Scrum Practices to try to solve the issues of this project according to its characteristics.

IV.2.3 Introducing Scrum for the development of an automotive application for engine management

This section aims to experiment the deployment of the Scrum Practices the project present previously. The use of Scrum Practices (in this project) are focused on managing the issues presented during the development of the project, that means that the issues will be addressed using the Scrum meaning. By using the number of elements involved in this project, we first, propose the distribution of the Scrum Team. Then the graphical view of the Sprint Framework will be introduced to address the issues. Finally, a global vision of the use of the Scrum Practices will be described.

IV.2.3.a Defining the Scrum Team

To better use the Scrum Practices, a new distribution of the members, involved in the development of the project, is proposed. The new distribution of the Scrum Team is illustrated in Figure 61. It can be notice from the figure, that the project leader becomes the Product Owner. The Development Team integrates the software mechanical, systems engineers, and the calibration pilots. All the engineers in charge of the generic solutions are considered as systems engineers. The new structure of the team includes the Scrum Master. The Scrum Master serves the Development Team, that means that instead having a software project leader, mechanical project leader, and so on, there is only the Scrum Master.

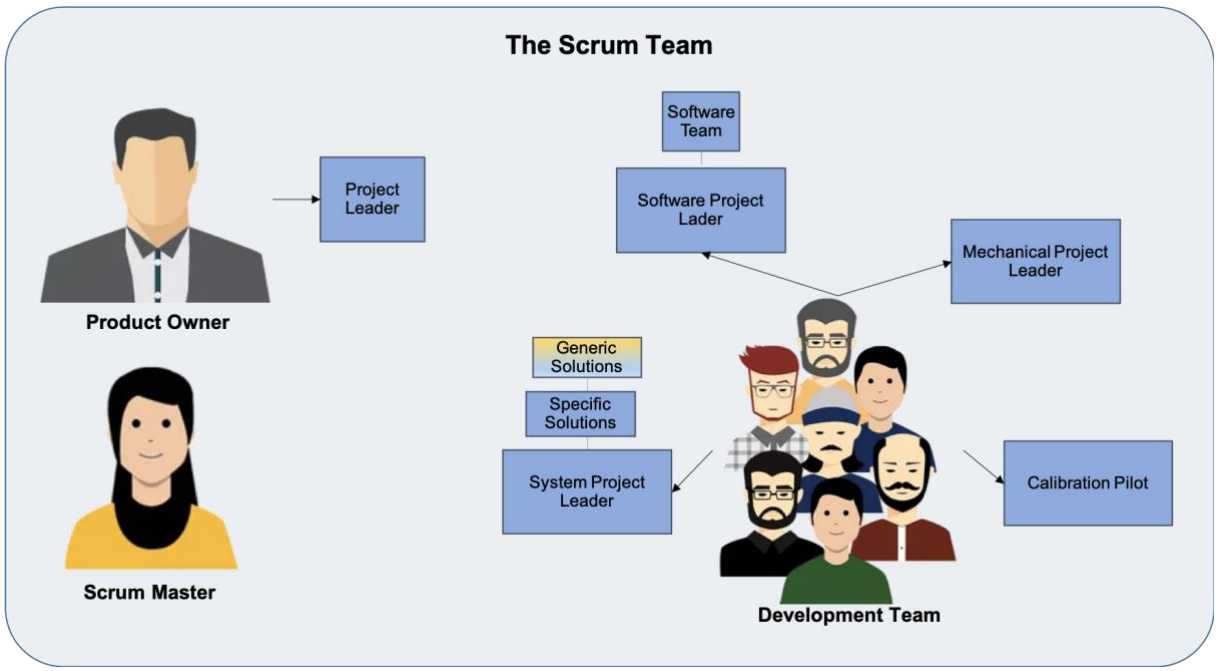

Figure 61. The distribution of the Scrum Team

In this project, the scrum team is distributed between France and Germany. Scrum Practices become complex when the distribution of the team is not centralized and there are many members (more than nine members) in the team. In that case, the definition of several Scrum Masters (one in France and another in Germany) could be an option to better coordinate the team. For this project we realized that most of team member are in France, it helps because there is no need to define different scrum masters. For the best use of the Scum Practices, the Scrum team should use different tools to communicate between them.

IV.2.3.b Using the Scrum Framework graphical view and the Scrum Practices

This project integrates hardware (mechanical/electronic) and software aspects to be integrated in parallel. By following the graphical view of the Scrum Framework, we first start with the definition of the Product Backlog (PB). The definition of the Product Backlog is in charge of the Product Owner (PO). Figure 62 illustrates the distribution of the PB. According to the project description, three samples of the automotive application for engine management, software releases (and its calibration files), and hardware aspects were the elements needed before the start of production.

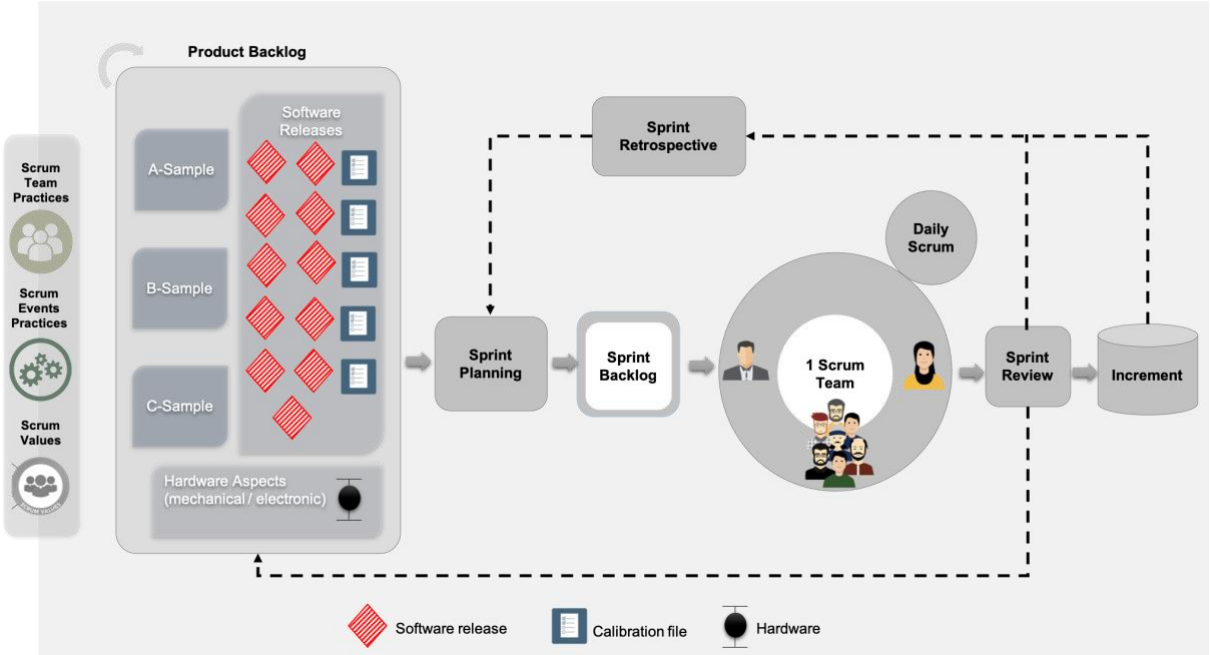

Figure 62. Product Backlog Definition

Some aspects should be considered to start the Sprint Planning:

- The project should be accomplished in four years,
- Every three or four moths the automotive supplier should deliver software updates with main functionalities (calibration files),
- The client needed to see rapid prototypes based in software versions,
- There is a presence of high technology innovation during the development of the B-sample.

These aspects will help to redefine the distribution of the project. First, we will consider the four years to develop the entire project. Every four months the Scrum Team will be deliver not only software releases but also hardware aspects (if needed), it will allow the integration of high technology innovation. Finally, to show to the client functional software versions, the planning of work will be distributed in a one-month horizon.

Figure 63 illustrates all the Product Backlog elements to be released. The sprint planning is defined in an eight-hour meeting (for one-month sprint). It can be noticed that for the A-sample and C-sample three software releases and their calibration files are planned, these samples also include hardware aspects. For the B-sample four software releases, their calibration files and hardware aspects are planned. The use

of Scrum Practices and the Scrum Values are always present in the definition of all the elements of the project. This distribution will help to introduce the Sprint Backlog.

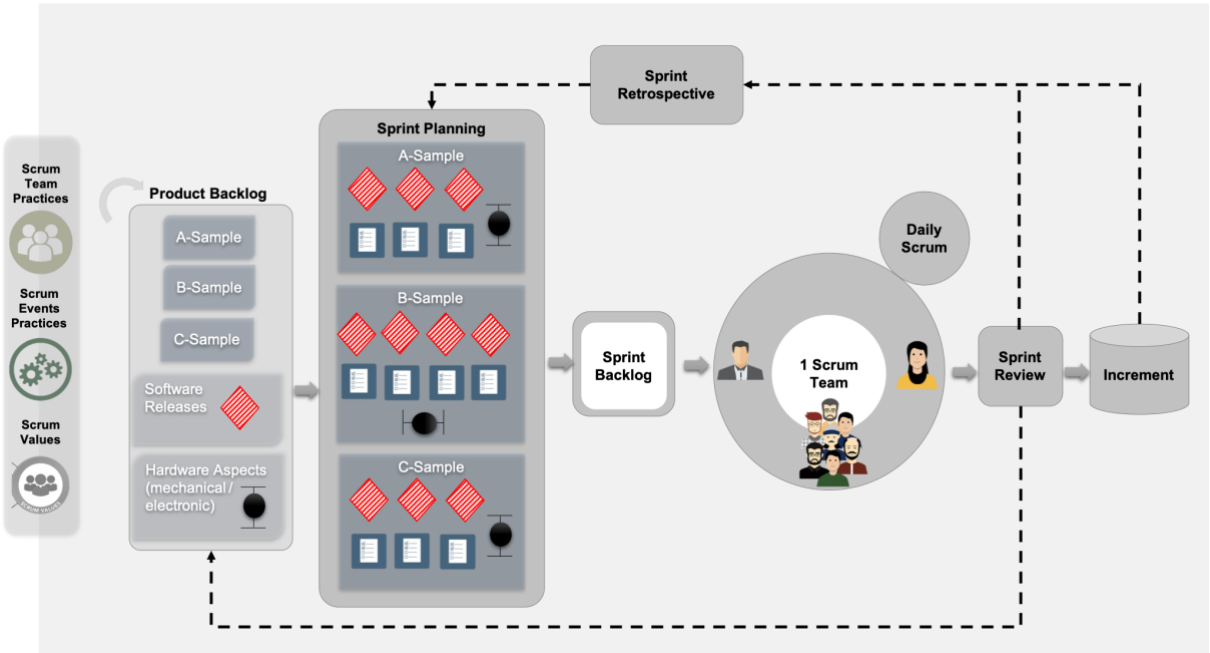

Figure 63. The distribution of the Sprint Planning

To ensure that functional parts of each sample are delivered, the Scrum Team have to estimate the hours of work of each activity (considering that each four months they have to deliver functional software versions). Figure 64 introduces the distribution of the Sprint Planning. It can be noticed that each software release includes four months stage activities. The Scrum Team should define all the activities to develop each software release. Hardware aspects are also included in each release, it will help to test the software versions in the hardware level.

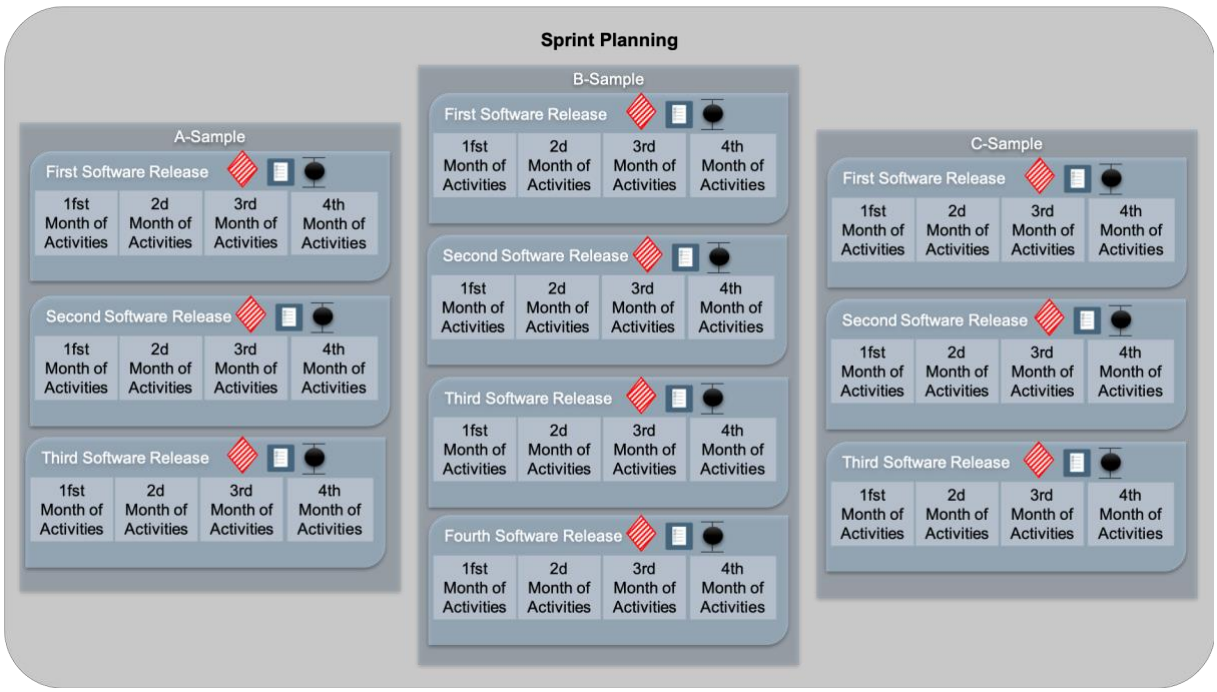

Figure 64. Distribution of the Sprint Planning

The distribution presented in Figure 64 will avoid the mock-ups and the different calibration values files. Instead of doing mock-ups we propose to adapt the software versions face to the real hardware conditions, that means that once the Scrum Team identifies hardware adaptations they can give priority to making the adaptations then to update the software version, instead of creating different software versions with different calibration files. The final distribution of the sprints is shown in Figure 65. Ten sprints are proposed to develop the entire project. Each sprint includes a four month-horizon sprints. For example (see Figure 65), to develop the A-sample three sprints are proposed: Sprint 1, 2 and 3 each sprint has the objective of deliver a software release that will allow the communication with the sensor. The Sprint 1 is composed for four sprints, each of which has a duration of one month.

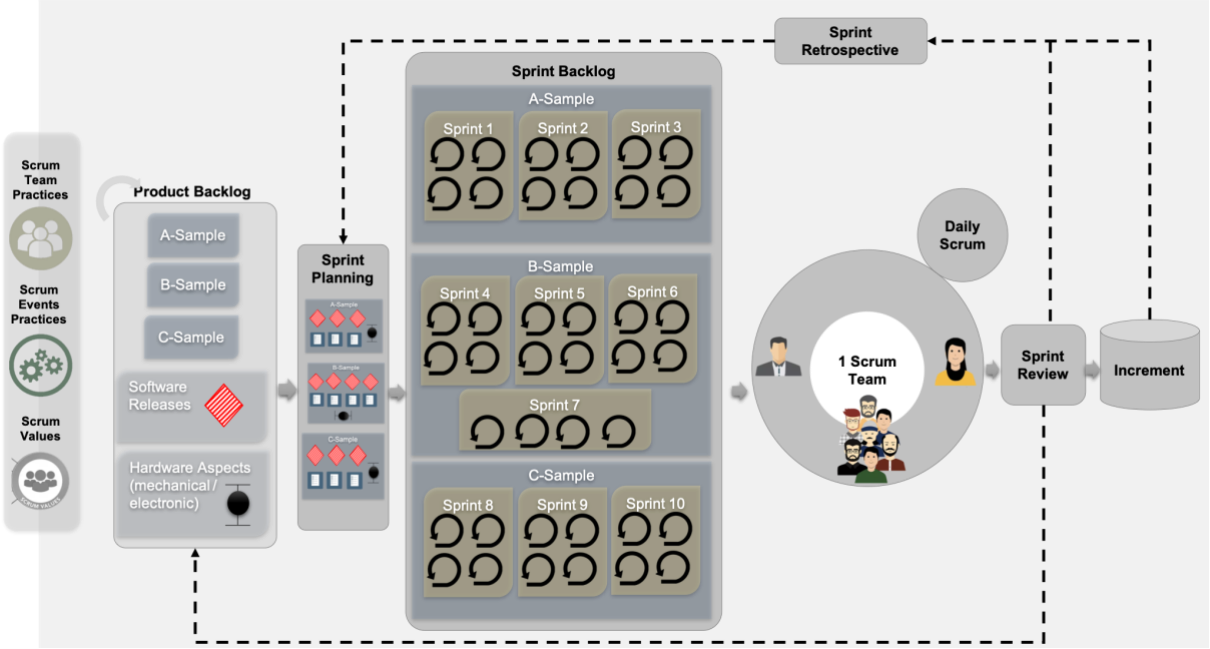

Figure 65. The final distribution of the Sprints

The distribution of one-month sprints will help to face the problems that might arise during the development of the project. For example, in the stage where the IC was not compatible, the integration of baby-boards could be done by developing them incrementally, that means delivering and testing according to the needed functionalities.

The project presented different uncertainties and evolving requirements during its development. The ten proposed sprints should follow the framework, to ensure the increments in each sprint review. The Scrum Practices and the Scrum Values have to be followed by all the members of the Scrum Team. The distribution proposed previously is only focused on the description of the project given at the beginning of this section, and it tries to face the presented problems using the scrum events (Sprint Review and Sprint Retrospective) of the scrum framework. Some difficulties could exist when trying the Scrum Practices, for example, self-management and self-organization (rather than having direct orders from superiors) could stress the members of the team. The entire team is actually held accountable for the success or failure of the project even when the Product Owner is to be held responsible. To divide the project in different sprints could become difficult according to the hardware needs of the product, that means that to prioritize activities could become a hard task for the Product Owner.

IV.3. Conclusions

This section analyses two different projects to implement the Scrum Framework as an alternative to the traditional management methods. The goal of this implementation is to identify how the projects could have benne developed in an agile way. The first project was the development of a connected robot. The implementation was proposed following three phases that the robot has to follow to accomplish a global mission. The sprint definition relied on the subsystems involved to achieve each phase of the global mission of the robot; it allows delivering software and hardware modules in parallel, during the sprint development, close communication should be effective in order to cover all the requirements defined in the robot specifications. Specific roles should be considered to manage the evolution of the robot development; these roles allow the distribution of responsibilities during the development.

For the second project evolving requirements and high technology innovation are present during the development of the project. The implementation of Scrum was proposed considering the three samples. The sprint definition relied on deliver software releases each four months. The members of the team were located in two countries, and a new distribution of the team is proposed in order to better coordinate the Scrum Team.

V. Conclusion

V.1. Contributions of the research

This thesis work aims to use agile practices for the management of engineering projects. In order to tackle the research objectives a scientific approach was followed. First, Chapter II introduced the definition of agility. The term agile is used to describe a mindset of values and principles that are supported by the Manifesto Agile for Software Development. Then, the events that gave rise to the agile movement are described until the definition of the agile manifesto. The agile manifesto states the values and principles for software development. Agile methods follow these principles and values. The most used agile methods were described. Lean and SAFe Framework were also described in this chapter. By using different criteria, the three philosophies were compared in order to identify differences between them. It has been concluded that agile methods are mainly used for software development, and the projects (in all size projects) developed in an agile way were more successful than traditional waterfall projects.

The theoretical bases bring the idea to transfer agile methods for the management of systems engineering projects (Chapter III). Chapter III addresses the contributions of this research. Frist, an exploration of the term agile was done in order to identify how this term appeared in systems engineering. Agile can be interpreted in two ways in systems engineering: the first one is focused to face the rapid change in the process of the system in an easy way (A-SE), and the second one is focused in the ability of the system to rapidly change from one estate to another in an easy way (AS-E). It has been concluded that the first one (A-SE) is the one most in line with the objective of this thesis work. Then, the notion of agility was sought in one of the different standards of systems engineering. The analysis consisted in exhaustively analyzing the tasks related to the activities of the Technical Management Processes and to check with the 12 principles from the Agile Manifesto. It has been concluded that if ever there was any notion, it would be in the Technical Management Processes group of the ISO/IEC

15288 standard. The result of this analysis led us to the first contribution of this thesis work:

First Contribution: *highlighting the introdution of agility in systems engineering process*

The first contribution helps to conclude that some of the technical management processes could be aligned with the agile principles and the introduction of agility in systems engineering could help in reducing development cycles and ensuring the control of the system. Finally, a research methodology was proposed to use an agile method for the management of engineering projects. The proposed research methodology led us to address different contributions in this thesis work:

Second Contribution: *defining the contextual model for systems engineering development*

By following different steps, a definition of a contextual model for systems engineering development was established. This model includes organizational factors (to define the environmental conditions) and project attributes (related to the process of the project). The project attributes were used to identify which agile method could be more adaptable according to the characteristics of the agile methods. The analysis led us to conclude that the Scrum Framework was the best to covers the project attributes for engineering projects. A definition of Scrum Practices was proposed based in the Scrum Guide (2017), and a deeper analysis was proposed to evaluated the impact of the project attributes in the Scrum Practices. The analysis consisted of defining different levels of impact. Three levels of impact were defined, and the results of this analysis led us to the third contribution of this work:

Third Contribution: *highlighting the difficulties while using Scrum in engineering projects and alternatives to solve them*

The possible difficulties while using Scrum in engineering projects were listed and. some alternatives were proposed in order to solve the identified difficulties.

The proposed research methodology helps us to conclude that Scrum can be used, in its natural state, for small engineering projects. However, for medium and large engineering projects, Scrum might present limitations in its use.

In Chapter IV, the analysis of two case studies were introduced. An educational project was evaluated, the project aims to develop a connected robot. Frist, the contextual model was used to identify what kind of project it was, it helps to conclude that the Scrum Practices would have no limitations because it was a small project. Then the scrum graphical view was used to plans the development of the robot. Three sprints were planned based in the three phases of the global mission of the robot. The proposition was focused in incrementally deliver functional parts of the robot (included hardware and software) instead of developing each module of the robot separately. The analysis of the second project followed the same schema. The second project aims to develop an automotive application for engine management. The contextual model showed that the Scrum Practices may have limitations or difficulties while using them because it was a large project. During the development of this project different issues were present. The graphical view of scrum was used to plan the development and to try to solve the issues during its development. A new distribution of the team was proposed, and ten global sprints was planned.

This work can help to integrate not only agile practices in systems engineering projects, but also to define a contextual model that help to identify the characteristics of engineering projects. According whit the analysis carried out in this thesis work, we can identify other aspects to be considered. These aspects are described in the perspectives and further work.

V.2. Perspectives and further work

Different questions were established to transfer agility in systems engineering. These questions led us to explore the notion of agility in systems engineering and define a schema to integrate Scrum for the management of engineering projects. In spite of that, there are still other avenues that should be considered, they are listed as follows.

a) *Conducting a deeper research for the definition of the contextual model for systems development*

The definition of a contextual model for systems development can help identify which agile method is the best to be used in engineering projects. However, the organizational factors and the project attributes should be evaluated more deeper. Different tools can be used to validate the contextual model, such as a database of different engineering projects, interviews with experts in the field, etc.

b) *Evaluating the Scrum Practices in engineering projects*

The thesis work only analysis two case studies. These case studies are projects that have been already done. The proposals are only suggestion of what may work, but the Scrum Practices should be tested in a real project to better list the limitations and difficulties while using them in engineering projects. It also will help to list all the possible alternatives to dismiss the limitations and difficulties.

c) *Considering how to measure agility in engineering projects*

Nowadays, there are not indicators that helps to measure the performance of agile projects. The definition of indicators should be proposed according to the agile practices.

Publications

Diego Armando Díaz Vargas, Claude Baron, Philippe Esteban. Systems Engineering and Agile Management. 7ème FORUM ACADEMIE- INDUSTRIE de l'AFIS, Toulouse, 7 et 8 décembre 2016.

Diego Armando Díaz Vargas, Claude Baron, Philippe Esteban, Citlalih Yollohtli Alejandra Gutierrez Estrada. IS THERE ANY AGILITY IN SYSTEMS ENGINEERING?. *INSIGHT Journal - International Council on Systems Engineering (INCOSE)*, Wiley, 2017, 20 (4), pp.11 - 14.

Diego Armando Diaz Vargas, Rui Xue, Claude Baron, Philippe Esteban, Rob Vingerhoeds, et al.. Implementing SCRUM to develop a connected robot. *12th International Conference on Modeling, Optimization and SIMulation (MOSIM 2018)*, Jun 2018, Toulouse, France. 8p., 2018.

References

Agile Alliance. (2018). Agile Alliance. Retrieved May 31, 2018, from https://www.agilealliance.org/

Agile Business Consortium. (2015, November 4). The DSDM Agile Project Framework (2014 Onwards). Retrieved May 31, 2018, from https://www.agilebusiness.org/content/introduction-0

Alain ROUSSEL. (2016, March). *Synthèse des articles de INSIGHT sur les démarches Agiles*.

Ambler, S. W. (2014). Feature Driven Development (FDD) and Agile Modeling. Retrieved May 31, 2018, from http://www.agilemodeling.com/essays/fdd.htm

Augustine, S., Payne, B., Sencindiver, F., & Woodcock, S. (2005). Agile project management: steering from the edges. *Communications of the ACM*, *48*(12), 85–89. https://doi.org/10.1145/1101779.1101781

Batra, D., Xia, W., VanderMeer, D., & Dutta, K. (2010). Balancing Agile and Structured

Development Approaches to Successfully Manage Large Distributed Software Projects: A Case Study from the Cruise Line Industry. *Communications of the Association for Information Systems*, *27*(1). Retrieved from https://aisel.aisnet.org/cais/vol27/iss1/21

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved May 31, 2018, from http://agilemanifesto.org/

Bottani, E. (2010). Profile and enablers of agile companies: An empirical investigation. *International Journal of Production Economics*, *125*(2), 251–261. https://doi.org/10.1016/j.ijpe.2010.02.016

Brocard, D. (2017, September 28). Blog | David Brocard | Consultant indépendant. Retrieved June 1, 2018, from http://www.davidbrocard.org/blog

Cambridge English Dictionary. (2017). agility Definition in the Cambridge English Dictionary. Retrieved May 31, 2018, from https://dictionary.cambridge.org/us/dictionary/english/agility

Cervone, H. F. (2011). Understanding agile project management methods using Scrum. *OCLC Systems & Services: International Digital Library Perspectives*, *27*(1), 18–22. https://doi.org/10.1108/10650751111106528

Chandra Misra, S., Kumar, V., & Kumar, U. (2010). Identifying some critical changes required in adopting agile practices in traditional software development projects. *International Journal of Quality & Reliability Management*, *27*(4), 451–474. https://doi.org/10.1108/02656711011035147

Cockburn, A. (2017). Crystal Light Methods. Retrieved May 31, 2018, from http://alistair.cockburn.us/Crystal+light+methods

Coffin, R., & Lane, D. (2016). A Practical Guide to Seven Agile Methodologies, Part 2 : Page 2. Retrieved August 21, 2018, from

http://www.devx.com/architect/Article/32836/0/page/2

Cohn, M. (2010). *Succeeding with agile: software development using Scrum*. Upper Saddle River, NJ: Addison-Wesley.

Conforto, E., Rebentisch, E., & Amaral, D. (2014). The Building Blocks of Agility as a Team's Competence in Project Management. Retrieved from http://dspace.mit.edu/handle/1721.1/88105

Crystal Methods - Wikiversity. (2017). Retrieved May 31, 2018, from https://en.wikiversity.org/wiki/Crystal_Methods

Dove, R., & LaBarge, R. (2014). 8.4.1 Fundamentals of Agile Systems Engineering - Part 1. *INCOSE International Symposium*, *24*(1), 859–875. https://doi.org/10.1002/j.2334-5837.2014.tb03186.x

Ehle, D. (2018). What is DevOps? The Ultimate Guide to DevOps. Retrieved May 31, 2018, from https://www.versionone.com/devops-101/what-is-devops/

Fairbairn, S. T. (2018). A New Muscle Memory: Training Systems Engineers in the Agile Culture of Trust. *INSIGHT Journal*, *21*(2), 17–20. https://doi.org/10.1002/inst.12198

Forsberg, K., Mooz, H., & Cotterman, H. (2005). *Visualizing project management: models and frameworks for mastering complex systems* (3rd ed). Hoboken, NJ: Wiley.
Fricke, E., Gebhard, B., Negele, H., & Igenbergs, E. (2000). Coping with changes: Causes, findings, and strategies. *Systems Engineering*, *3*(4), 169–179. https://doi.org/10.1002/1520-6858(2000)3:4<169::AID-SYS1>3.0.CO;2-W

Ganguly, A., Nilchiani, R., & Farr, J. V. (2009). Evaluating agility in corporate enterprises. *International Journal of Production Economics*, *118*(2), 410–423. https://doi.org/10.1016/j.ijpe.2008.12.009

Haberfellner, R., & de Weck, O. (2005). 10.1.3 Agile SYSTEMS ENGINEERING

versus AGILE SYSTEMS engineering. *INCOSE International Symposium*, *15*(1), 1449–1465. https://doi.org/10.1002/j.2334-5837.2005.tb00762.x

Hastie, S., & Wojewoda. (2015). Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch. Retrieved May 31, 2018, from https://www.infoq.com/articles/standish-chaos-2015

Hoda, R., & Murugesan, L. K. (2016). Multi-level agile project management challenges: A self-organizing team perspective. *Journal of Systems and Software*, *117*, 245–257. https://doi.org/10.1016/j.jss.2016.02.049

IEC/IEEE. (2015). ISO/IEC/IEEE 15288:2015 - Systems and software engineering -- System life cycle processes. Retrieved June 1, 2018, from https://www.iso.org/standard/63711.html

INCOSE. (2017). What is Systems Engineering. Retrieved September 4, 2018, from https://www.incose.org/about-systems-engineering

Jeffries, R. E. (1998). what-is-extreme-programming. Retrieved May 31, 2018, from https://ronjeffries.com/xprog/what-is-extreme-programming/]

Knaster, R., & Leffingwell, D. (2017). *SAFe 4.0 distilled: applying the Scaled Agile Framework for Lean software and systems engineering*. Retrieved from http://proquest.safaribooksonline.com/?fpi=9780134209487

Koehnemann, H. (2018). Using Agile Systems Engineering Workshops and Model-Based Systems Engineering to Drive Agile Development. *INSIGHT Journal*, *21*(2), 39–42. https://doi.org/10.1002/inst.12203

Kruchten, P. (2013). Contextualizing agile software development. *Journal of Software: Evolution and Process*, *25*(4), 351–361. https://doi.org/10.1002/smr.572

Maheshwari, A., Raz, A. K., DeLaurentis, D. A., Murphy, A., & Kolawole, O. (2018). Integrating SysML and Agent-Based Modeling for Rapid Architecture Evaluation.

INSIGHT Journal, *21*(2), 47–51. https://doi.org/10.1002/inst.12205

Markina-Khusid, A., Moulton, A., Howard, G., James, J. H., Mahoney, P. J., & Ricks, S. (2018). Agile Dynamics at Scale. *INSIGHT Journal*, *21*(2), 29–31. https://doi.org/10.1002/inst.12201

Meyer, B. (2014). *Agile! the good, the hype and the ugly*. Cham: Springer.

Mueller, E. (2017). What Is DevOps? Retrieved June 4, 2018, from https://theagileadmin.com/what-is-devops/

Oehmen, J., Oppenheim, B. W., Secor, D., Norman, E., Rebentisch, E., Sopko, J. A., … Driessnack, J. (2012). *The Guide to Lean Enablers for Managing Engineering Programs*. Joint MIT-PMI-INCOSE Community of Practice on Lean in Program Management. Retrieved from http://dspace.mit.edu/handle/1721.1/70495

PMI (Ed.). (2017a). *Agile practice guide*. Newtown Square, Pennsylvania, USA: Project Management Institute.

PMI (Ed.). (2017b). *A guide to the project management body of knowledge: PMBOK guide* (Sixth edition). Newtown Square, Pennsylvania, USA: Project Management Institute.

Poppendieck, M., & Poppendieck, T. (2010). *Lean software development: an agile toolkit* (Nachdr.). Boston: Addison-Wesley.

Pressman, R. S. (2010). *Software engineering: a practitioner's approach* (7. ed). Boston, Mass.: McGraw-Hill/Higher Education.

Rebentisch, E. (2017). *Integrating Program Management and Systems Engineering: Methods, Tools, and Organizational Systems for Improving Performance*. John Wiley & Sons.

Rebovich, G. (2006). Systems Thinking for the Enterprise: New and Emerging Perspectives. In *2006 IEEE/SMC International Conference on System of Systems*

Engineering (pp. 191–196). Los Angeles, California, USA: IEEE. https://doi.org/10.1109/SYSOSE.2006.1652297

Repenning, J., Kieffer, D., & Repenning, N. (2017). *Agile for Everyone Else: Using Triggers and Checks to Create Agility Outside of Software Development* (Working Paper). Retrieved from http://dspace.mit.edu/handle/1721.1/110325

Rosser, L., Marbach, P., Osvalds, G., & Lempia, D. (2014). 7.4.2 Systems Engineering for Software Intensive Projects Using Agile Methods. *INCOSE International Symposium*, *24*(1), 729–744. https://doi.org/10.1002/j.2334-5837.2014.tb03178.x

Schapiro, S. B., & Henry, M. H. (2012). Engineering agile systems through architectural modularity. In *2012 IEEE International Systems Conference SysCon 2012* (pp. 1–6). Vancouver, BC, Canada: IEEE. https://doi.org/10.1109/SysCon.2012.6189529

Schindel, B. (2018). Managing Awareness in a CURVE-y World: Agile Systems Engineering for Autonomous Vehicles. *INSIGHT Journal*, *21*(2), 13–16. https://doi.org/10.1002/inst.12197

Schwaber, K., & Sutherland, J. (2017). Scrum Guide | Scrum Guides. Retrieved May 31, 2018, from https://www.scrumguides.org/scrum-guide.html

Sharon, A., de Weck, O. L., & Dori, D. (2011). Project management vs. systems engineering management: A practitioners' view on integrating the project and product domains. *Systems Engineering*, *14*(4), 427–440. https://doi.org/10.1002/sys.20187

Sheard, S. A. (1996). TWELVE SYSTEMS ENGINEERING ROLES. *INCOSE International Symposium*, *6*(1), 478–485. https://doi.org/10.1002/j.2334-5837.1996.tb02042.x

Smiley, K. J., Louis, C. L., Augustine, V., Sharma, S., Nistor, I., Cordes, A., & Becker, O. (2018). Overcoming the Challenges of Agile for Globally Distributed Industrial Research. *INSIGHT Journal*, *21*(2), 34–38. https://doi.org/10.1002/inst.12202

Stark, E., & ClydeBank Business. (2016). *Agile project management QuickStart guide: the simplified beginner's guide to agile project management*.

Stettina, C. J., & Hörz, J. (2015). Agile portfolio management: An empirical perspective on the practice in use. *International Journal of Project Management*, *33*(1), 140–152. https://doi.org/10.1016/j.ijproman.2014.03.008

Tolentino, G., & Wood, J. (2018). Balancing Systems Engineering Rigor with Agile Software Development Flexibility. *INSIGHT Journal*, *21*(2), 25–28. https://doi.org/10.1002/inst.12200

Uskov, V., Krishnaiah, D. B., Kondamudi, R., & Singh, U. (2016). Innovative agile project management curriculum for engineering education (pp. 463–468). IEEE. https://doi.org/10.1109/EDUCON.2016.7474594

Walden, D. D., Roedler, G. J., Forsberg, K., Hamelin, R. D., Shortell, T. M., & International Council on Systems Engineering (Eds.). (2015). *Systems engineering handbook: a guide for system life cycle processes and activities ; INCOSE-TP-2003-002-04, 2015* (4. ed). Hoboken, NJ: Wiley.

Wikipedia. (2018). Engine. In *Wikipedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=Engine&oldid=867683334

Xue, R. (2016). *Improving Cooperation between Systems Engineers and Project Managers in Engineering Projects - Towards the alignment of Systems Engineering and Project Management standards and guides* (phdthesis). INSA de Toulouse. Retrieved from https://hal.laas.fr/tel-01376031/document

Zhu, H., Zhou, M., & Seguin, P. (2006). Supporting Software Development With Roles. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *36*(6), 1110–1123. https://doi.org/10.1109/TSMCA.2006.883170