AS3 : A method to design service-based adaptive systems for smart environments

Soufiane Faieq

Université Mohammed V
Faculté des Sciences
Rabat

UGA
Université
Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE GRENOBLE ALPES

préparée dans le cadre d'une cotutelle entre l'*Université Grenoble Alpes* et *l'Université Mohammed V de Rabat*

Spécialité : **Informatique**

Arrêté ministériel : le 6 janvier 2005 – 25 mai 2016

Présentée par

Soufiane FAIEQ

Thèse dirigée par **Agnès FRONT** et **Moulay Driss RAHMANI**
codirigée par **Rajaa SAIDI**

préparée au sein des **Laboratoires LIG (Laboratoire d'Informatique de Grenoble)** et **LRIT (Laboratoire de Recherche en Informatique et Télécommunications)**

dans **les Écoles Doctorales EDMSTII (École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique)** et **CEDESTR (Centre d'Etudes Doctorales en Sciences et Technologies de Rabat)**

AS3: Une méthode de conception des systèmes adaptatifs basés sur les services pour les environnements intelligents

AS3: A method to design Adaptive Service-based Systems for Smart environments

Thèse soutenue publiquement le **01/12/2021**, devant le jury composé de :

Monsieur Ahmed HAMMOUCH
Professeur, UNIVERSITÉ MOHAMMED V de RABAT, Président
Madame Frédérique BIENNIER
Professeur, INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON, Rapportrice
Monsieur Amine BAÏNA
Professeur, INSTITUT NATIONAL DES POSTES ET TELECOMMUNICATIONS, Rapporteur
Madame Manuele KIRSCH-PINHEIRO
Maitre de Conférences HDR, UNIVERSITE PARIS 1 - PANTHEON SORBONNE, Examinatrice
Monsieur Ahmed LBATH
Professeur, UNIVERSITE GRENOBLE ALPES, Examinateur
Madame Agnès FRONT
Professeur, UNIVERSITE GRENOBLE ALPES, Directrice de thèse
Monsieur Moulay Driss RAHMANI
Professeur, UNIVERSITÉ MOHAMMED V de RABAT, Directeur de thèse
Madame Rajaa SAIDI
Professeur, INSTITUT NATIONAL DE STATISTIQUE ET D'ÉCONOMIE APPLIQUÉE, Codirectrice de thèse
Monsieur Hamid EL GHAZI
Professeur Assistant HDR, INSTITUT NATIONAL DES POSTES ET TELECOMMUNICATIONS, Invité

To my parents.
To my family.
To my friends.
To my professors.
To my colleagues.

Acknowledgements

As the curtains close on this beautiful journey, I cannot help but reflect on the hurdles and difficult times that have accompanied it and the people whom helped me through them. This section is but a small way to acknowledge the impact these people had on my life and the journey to becoming a PhD.

I would like to start by thanking the coordinators of PHC TOUBKAL for the opportunity provided by the program that allowed to establish a joint PhD between the LRIT (Laboratoire de Recherche en Informatique et Télécommunication) and LIG (Laboratoire d'Informatique de Grenoble) laboratories. This project was financially supported by CAMPUS FRANCE (PHC TOUBKAL 2017 (French-Morocco bilateral program) Grant Number: 36804YH). Their financial support allowed me to experience different research contexts both scientifically and culturally.

I would like express my deep gratitude to my advisors and supervisors Prof. SAIDI Rajaa, Prof. FRONT Agnès, Prof. EL GHAZI Hamid and Prof. RAHMANI Moulay Driss for their invaluable guidance, advice and contribution to the development of this PhD project. Their presence throughout the project has propelled me forward at each step and the quality of our interactions has made me grow as a researcher and as a person.

I would like to extend my gratitude to Prof. HAMMOUCH Ahmed for taking the time off his busy schedule and accept to report on this thesis and preside over its defense, as well as Prof. BIENNIER Frédérique and Prof. BAÏNA Amine for the time they have invested in reading and reviewing this manuscript. Their comments and suggestions were of the utmost pertinence and helped me improve the quality of my thesis. I would also like to thank my other thesis jury members, Prof. KIRSCH-PINHEIRO Manuele and Prof. LBATH Ahmed for accepting to examine this thesis.

Many thanks to my dear colleagues and friends with whom I have shared this journey both in Morocco and France. The experiences and talks we shared have always had a tremendous impact on my moral and knowledge. Around a coffee table or just standing near the lab entrance, the conversations we had were always a breeze of fresh air in what mostly was a mind-bending endeavor.

To my friends, I would like to express my sincere gratefulness and appreciation for your invaluable friendship and emotional support, and for helping me get through the tough times. Social life was always important to me and thanks to you it wasn't hard to maintain the ties and keep up even when the geographical distance wasn't favorable.

Lastly but most importantly, my deepest gratitude and acknowledgment go to my parents who have raised me to be inquisitive and free to pursue my dreams and goals at my own pace and in my own way. Their continuous and unlimited support have always given me the peace of mind and tranquility to keep going forward when the times were tough. THANK YOU !

Abstract

Smart systems are systems that rely on technological advancements to continuously adapt and improve in order to provide added-value to their users. Designing these systems in a coherent and methodical way is important to set the stage for highly interoperable and collaborative systems with the potential to propel the software community into an era of Systems of Systems. However, the different and numerous concepts, technologies and techniques that have been linked and used recently to develop these systems made this task a challenging endeavor. Indeed, the existing literature on the subject is focused on the technical aspects of developing smart systems with very little effort and thought to how these systems should be designed.

To tackle this gap, this thesis proposes and develops a method, called AS3, to analyze and design smart systems. The method starts from a broad definition of the smart system and builds on it to define a smart system loop that provides an integrated view of the main entities that are present in a smart system and their interactions. This smart system loop builds on the adaptability loop as well as the main concepts from context-awareness and service orientation to cover the life cycle of the smart system. Supported by a product metamodel and a process model, the method then provides the intentions and strategies that can be followed in order to design context-aware service-based smart systems. To insure the continuous improvement of the system, the method supports recommendation to allow easy automation of the improvement while keeping the method user in the loop. To showcase the relevance and the efficacy of the AS3 method, this thesis includes a complete rundown of the method to design a system that deals with road security called SMARTROAD.

Keywords: Smart system, system design, service-orientation, context-awareness, system improvement, recommendation system.

Résumé

Les systèmes smart sont des systèmes qui s'appuient sur les progrès technologiques pour s'adapter et s'améliorer de manière continue afin d'apporter une valeur ajoutée à leurs utilisateurs. Il est donc important de concevoir ces systèmes de manière cohérente et méthodique pour préparer le terrain à des systèmes hautement interopérables et collaboratifs, susceptibles de propulser la communauté logicielle dans l'ère des systèmes de systèmes. Cependant, les différents et nombreux concepts, technologies et techniques qui ont été liés et utilisés récemment pour développer ces systèmes ont fait de cette tâche un véritable défi. En effet, la littérature existante sur le sujet se concentre sur les aspects techniques du développement de systèmes smart avec très peu d'effort et de réflexion sur la façon dont ces systèmes devraient être conçus.

Pour combler cette lacune, cette thèse propose et développe une méthode, appelée AS3, pour analyser et concevoir des systèmes smart. La méthode part d'une définition abstraite du système smart et s'appuie sur celle-ci pour définir une boucle de système smart (smart system loop) qui fournit une vue intégrée des principales entités présentes dans un système smart et de leurs interactions. Cette boucle de système smart s'appuie sur la boucle d'adaptabilité ainsi que sur les principaux concepts de la sensibilité au contexte et de l'orientation vers les services pour couvrir le cycle de vie du système smart. Soutenue par un métamodèle de produit et un modèle de processus, la méthode fournit ensuite les intentions et les stratégies qui peuvent être suivies afin de concevoir des systèmes smart basés sur des services et sensibles au contexte. Pour assurer l'amélioration continue du système, la méthode prend en charge la recommandation comme un mécanisme permettant une automatisation facile de l'amélioration tout en gardant l'utilisateur de la méthode au courant de toute modification du système. Afin de démontrer la pertinence et l'efficacité de la méthode AS3, cette thèse inclut une application de la méthode pour concevoir un système smart qui traite la sécurité routière appelé SMARTROAD.

Mots-clés : Système smart, conception de système, orientation service, sensibilité au contexte, amélioration de système, système de recommandation.

Contents

List of Figures

List of Tables

Acronyms

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
AI	Artificial Intelligence
API	Application Programming Interface
BDaaS	Big Data as a Service
BPEL	Business Process Execution Language
BPM	Business Process Management
CAC	Context-Aware Computing
CAS	Complex Adaptive System
CBD	Component-based Design
CBR	Case-Based Reasoning
CC	Cloud Computing
CoAP	Constrained Application Protocol
CPS	Cyber-Physical System
DBMS	DataBase Management System
DL	Deep Learning
DM	Data Mining
DS	Data Science
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
ICT	Information and Communications Technology
IoT	Internet of Things
IP	Internet Protocol

ISO	International Organization for Standardization
ITS	Intelligent Transport System
JSON	JavaScript Object Notation
MAC	Media Access Control
MDE	Model-Driven Engineering
ML	Machine Learning
NoSQL	Not Only Structured Query Language (SQL)
NSF	National Science Foundation
PaaS	Platform as a Service
QoS	Quality of Service
REST	REpresentational State Transfer
RFID	Radio Frequency IDentification
RQ	Research Question
RS	Recommender System
SaaS	Software as a Service
SBS	Service-based System
SC	Service Computing
SE	Smart Environment
SLA	Service-Level Agreement
SmartSyLo	Smart System Loop
SNS	Sensor Network System
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SoS	System of Systems
SQL	Structured Query Language
SS	Smart System
SSH	Secure SHell
SSS	Service-based Smart System
SVM	Support Vector Machine
TaaS	Things as a Service

TCP	Transmission Control Protocol	
UbiComp	Ubiquitous Computing	
UDP	User Datagram Protocol	
UML	Unified Modeling Language	
UN	United Nations	
UX	User eXperience	*
WADL	Web Application Description Language	
WSDL	Web Service Definition Language	
WSN	Wireless Sensor Network	
XML	eXtensible Markup Language	

Chapter 1

Introduction

Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning

Albert Einstein

This chapter presents the general scope of the research work carried throughout this thesis project. First, we introduce the motivation behind this work and the relevant topics that are discussed along its contextualization. Afterwards, we specify the research questions that were identified as the basis of this thesis before presenting the objectives and contributions of our research work. Then, we present the *SMARTROAD* case study through which we aim to illustrate the usefulness and pertinence of our contributions. Finally, we describe the structure of this thesis document in chapters and briefly state the content of each chapter.

1.1 Research Context

In 2011, Marc Andreessen coined his now famous phrase "Software is eating the world" to shortly describe the proliferation of software systems in different industries and everyday life (Andreesseen, 2011). This proliferation and the clear benefits of software for businesses and end-users alike, created a certain dependence on what they offer and raised little by little what is expected of them. As a result of this phenomenon, the software community was affected mainly at two levels. First, it led to an increasing demand on software products owing mostly to the competitive advantage it provides businesses. Second, it made software systems ever more complex due to complex and excessive requirements.

To deal with the growing demand for and complexity of software products, the software engineering community had to come up with new paradigms, concepts, methods

and frameworks to think about how to build quality software quickly, efficiently and effectively. To be successful in this endeavor, these artifacts would need to be (i) flexible, to allow customization to the user's preferences and adaptation to his needs and changes in his environment, (ii) modular, to allow frequent component reuse and easy integration, and (iii) generic, to allow continuous improvement and consistent interoperability.

One of such paradigms that has known immense acceptability and success is Service Computing (SC), also known as Service Oriented Computing (SOC). It is defined as *"the discipline that seeks to develop computational abstractions, architectures, techniques, and tools to support services broadly."* (Bouguettaya et al., 2017). Services, as the basic components of SC, allow the abstraction from low level technical details, while focusing on high level functional ones. SC design approaches, based on different derivatives of SOA, are considered nowadays as *de facto* standard for building large scale distributed applications.

Using SOA as a guideline allows the creation of distributed software systems that are loosely-coupled, flexible and platform agnostic; mostly based on the principles of reuse and composition of several services that are modular, autonomous and self-describing, and that can be offered internally or through third parties (Bieberstein et al., 2005). Consequently, several types of services are nowadays provided by different companies in numerous domains and at different scales (e.g., smart home, industry 4.0, e-commerce platforms, sharing platforms, crowd-sourcing platforms, etc.).

Almost a decade after Andreessen's famous phrase was coined, software continues to invade new markets and industries. Most recently, the excessive use and integration of software into different facets of everyday life and its embedding in what was once purely mechanical, electrical, hydraulic or pneumatic systems gave rise to what we call "*SE*s". We define SEs as the following:

> *"Smart Environments are virtual extensions of the physical world, that are capable of providing customized services to their users when required and adapting to their changing needs"*

The idea is to deal with the issues related to both the supporting physical environment (e.g., management of limited resources) and the people evolving inside them (e.g., facilitate and optimize complex operations) through the use of computing paradigms. Hence, SEs exhibit five major characteristics:

1. *being dynamic, heterogeneous and mobile.* As SEs are formed by the integration of human and software agents. There are numerous possible configurations for each SE. This makes SEs: (i) dynamic, as their concrete components can change at any time, (ii) heterogeneous, as the components are of different natures and use different protocols and implementation details, (iii) mobile, as their geographical scope can be constantly moving.

2. *being highly collaborative.* As we move towards a highly connected world, the borders and boundaries between different organizations and domains become blurry. This is

the result of realizing that major value co-creation can be achieved through intensive collaboration.

3. *having a user-centered vision*. As SEs are supposed to provide their users with tailored services, they are built with a special attention to the needs and preferences of the user. Hence, they should be adaptable enough to serve different users that may be interested in the offered services for different reasons.

4. *providing a myriad of services*. As SEs are open ecosystems and can span different domains, numerous business services are provided to the users by multiple entities due to competitiveness. While some of the offered services fulfill the same functionality, they do so at different levels of non-functional properties.

5. *being dependent on the advances of technology*. As SEs are the product of different Information and Communications Technology (ICT) disciplines coming together, we can safely assume that future technological advances will play a big role in the evolution of SEs.

We designate the type of systems that operate on the aforementioned environments as a "*SS*". So far, the design and operationalization of these systems is based on data (Lim and Maglio, 2019). In addition to dealing with challenges related to the characteristics of SEs, these systems should exhibit two features:

1. They are supposed to be *context-aware*. They must exploit the wealth of data available in SEs to fully or in part automate the tasks of the users or help them make decisions by providing relevant information or services.

2. They must also be *(self-)adaptive*. These systems are required to work in dynamic environments and thus should be able to learn from their past to adapt and/or improve their functionalities and non-functional properties.

Several SS projects have spored in the last decade having varying scopes and operating on different environments including manufacturing (H. Lee and J. Lee, 2018), domotics (Palanca et al., 2018), transport (J. Chang et al., 2017), agriculture (Sivamani et al., 2013), etc. However, most of the existing SSs remain application-specific and are developed in an *ad hoc* manner from scratch. This manner of system building not only breaks the reuse and composition principles of software engineering, but also leads to duplicate solutions that are not interoperable and use too much resources to achieve their purpose (Santana et al., 2017). Moreover, the erratic use of enabling technologies, such as virtualization techniques and communication protocols, makes the developed solutions hard to integrate with each other, presenting a fragmented landscape (Truong and Dustdar, 2015).

Meanwhile, Data Science (DS) and its related disciplines (e.g., Machine Learning (ML), Data Mining (DM), etc.) are expected to play a major role in the "smartification"

movement (Hashem, V. Chang, et al., 2016). It is used to develop systems that are capable of delivering actionable information to other systems or to decision makers, through the analysis of the huge amount of data transiting on the network nowadays. Recommender System (RS)s are a popular implementation of such systems, as they produce relevant items for the active users based on the analysis of previously recorded actions performed by those same users or previous ones. RSs have shown their effectiveness and have been used for several application domains such as, tourism (Luberg et al., 2011), e-commerce (Sarwar et al., 2000), news (Garcin et al., 2013) and entertainment (Christensen and Schiaffino, 2011).

The use of RSs in the development of service-based SSs can be beneficial in several aspects. On the one hand, they are used to tackle run-time challenges related to the system's functionality where the goal is to optimize or improve its performance. For instance, there are challenges where there is a need to get actionable information from incomplete and sparse data (e.g., recommending services based on incomplete Quality of Service (QoS) data) (Zheng, Y. Zhang, et al., 2014). On the other hand, they can also be used to address design-time challenges. The goal of using recommender systems at this level is to improve the design of the system by making its engineers and domain experts aware of the run-time trends and patterns in the usage of their services by the users and recommend actions based on these patterns to improve the initial design.

As figure 1.1 shows, our research context is positioned in the middle of these previously mentioned disciplines. Our aim in this dissertation is to provide a clear and consistent method for the design and development of SSs. A method that not only helps in tackling the intrinsic challenges related to SEs, but also incorporates models that capture the concepts at play in SSs and guide the engineers in their making. To that end, we ground our approach on well-established abstractions, practices and techniques from the SC and DS communities. Particularly, we consider *services* as the basic component of any SS and *recommendation* as its continuous improvement process.

1.2 Research Questions and Rationale

Based on the characteristics of SSs mentioned in section 1.1 and our primary review of the literature, we realized that the design and development of SSs differs from traditional software systems. Indeed, SSs present themselves as highly complex, collaborative and evolving ecosystems. This realization led us to elaborate the following general Research Question (RQ).

> *How can we design and build systems and software solutions that are suitable for smart environments ?*

This research question sets the context of the present dissertation. Starting from this guiding research question and upon more in-depth analysis of the literature, we identified

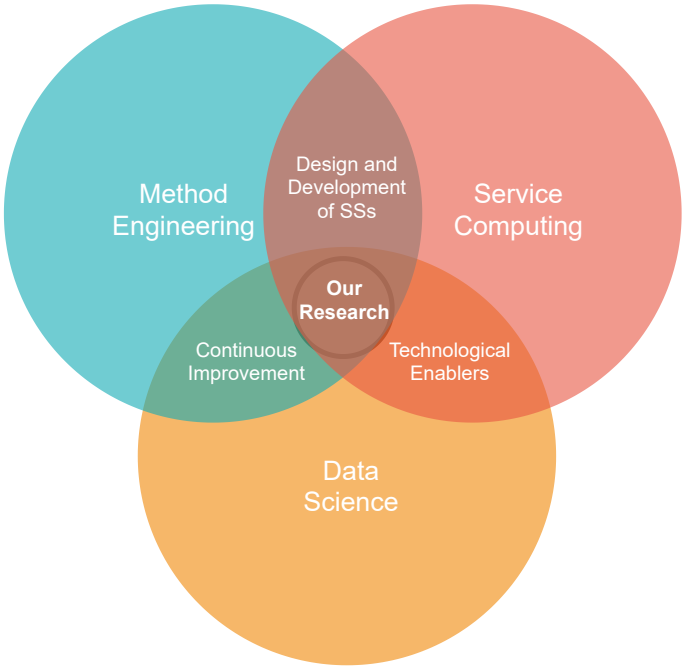


Figure 1.1 – The context of the research work in this thesis.

more specific and fine-grained research questions that form our main concerns. Each of these research questions develops on a specific set of challenges that face the design and development of SSs. In the following, we state these specific research questions and describe the rationale behind their elaboration.

> ***RQ1:*** *What are the main concepts that should be taken into account when building a smart system ?*

This research question stems from the fact that existing approaches to the design and development of SSs in literature seem to focus on different concepts and aspects. This lack of consensus on first class concepts introduces several challenges. First, it creates a fragmented landscape where the approaches cannot integrate and interoperate with each other. Second, it breaks the reuse principle as the previously identified concept instances cannot be discovered and reused in new or complementary smart systems. Third, it creates a confusion for the engineers trying to design their SS as the concepts are numerous and sometimes ambiguous across different domains. These challenges lead to an increase in development time, effort and cost of SS projects and also make the endeavor error-prone. In addition, they prevent the achievement of a benchmark regarding the best practices of SS design and development.

> ***RQ2:*** *How do the previously mentioned concepts relate to each other and what aspects of the smart system do they represent and/or influence ?*

Understanding the relationships between the concepts of SSs helps system engineers identify concept instances from their application domain. Though it is tightly related

to the identified concepts, we introduce this question as a separate research question from *RQ1* as the relationships between the main concepts should be defined regardless of what these concepts are. This allows system engineers to understand the nature of the interactions between the concepts. Also, it allows the extensibility of the approach as the introduction of new and emerging concepts (due to technological advances) would compel practitioners and academics to describe how these new concepts relate to the existing ones. These elements are essential to keep a coherent understanding of the concepts at play in designing and developing SSs.

> ***RQ3:*** *What enabling technologies can be used to implement and operationalize smart systems ?*

Existing technology stacks (e.g. Internet of Things, Cloud Computing and Big Data) offer several advantages for the implementation and operationalization of different aspects related to applications in different domains. However, for an SS these technologies can be more of a liability than an asset for three reasons. First, existing SSs in literature show an erratic use of these technologies. This makes the interoperability and integration between the different SSs an additional endeavor to the design and development of the SSs. Second, these technologies and their offerings are separated from the design of the SS. This makes them limited to the support of the SS's functions, ignoring how they fit into the system's operations and how they can benefit from the system's usage to optimize their performance. Third, there is a lack of collaboration between the enabling technologies as each technology stack provider focuses of his offerings. To fully benefit from the force of these technologies, they need to be able to collaborate and communicate information.

> ***RQ4:*** *How can the smart system and its design be adapted easily by its administrators (i.e., system engineers and business experts) ?*

As discussed above in section 1.1, an SS is supposed to be able to adapt to its environment and improve its performance. Recent approaches to system adaptation and improvement have been focused on self-* properties, meaning that the system should be able to self-adapt and self-improve. While this can be beneficial in most cases, in other cases this can be dangerous, especially in sensitive application domains like transport security, health-care or manufacturing. Hence, there is a need for a way that allows the system to self-adapt and self-improve while at the same time keeping its designers and stakeholders in the loop and in control. Moreover, the system's adaptation and improvement possibilities are tightly tied to its initial design. Therefore, the system's design itself should be flexible and adaptable while keeping consistency and compliance with previous designs.

1.3 Objectives and Contributions

This thesis presents our efforts to address the research questions described above. The main goal of this thesis is to **allow system engineers to methodically design SSs that are reproducible, adaptive, reusable and composable**. This entails addressing the concerns related to these systems at design-time and run-time. To achieve this objective, we develop an approach derived from our reflections on what constitutes an SS and grounded by concepts and techniques in both the SC and DS communities. The following items point out the objectives of this thesis and the contributions made to achieve them in more detail.

- **Define a general model describing the capabilities of a typical SS** To allow software engineers and architects to develop efficient and effective SSs, it is imperative to understand and define how an SS works. This entails defining how an SS captures information, processes it and acts on it to deliver customized services to its users. To that end, we model the capabilities of SSs using a loop we call "Smart System Loop". This loop defines the capabilities of SSs to perceive and respond to their environment using known concepts from context-aware computing, service computing and adaptability.

- **Propose a uniform way to design adaptable and service-based SSs.** In order to build software products that are reproducible, reusable and composable, there is a need for a uniform and holistic way to think and design these products. To that end, we propose a method to design SSs that is based on the Smart System Loop defined above. In this method, we identify the key general concepts in these products (i.e., SSs) and propose an intentional approach based on the MAP formalism (Rolland, 2007) to instantiate these concepts to build a target SS.

- **Provide tools and techniques to allow the continuous improvement of SSs.** Software products are now subject to frequent changes in requirements and environment. They need to be adaptive in order to face these challenges. However, software engineers/architects and domains experts can not keep up with the frequency of changes and the improvements they entail within the SS. In the proposed method, we think about adaptability early on in the design of SSs and we provide some tools to facilitate the task for these users and guide them in the adaptation process. The provided tools and techniques stem from ML techniques that we use to recommend actions to the method users.

The research work carried out in the course of this thesis has led to the publication of several articles in international peer-reviewed journals and conference proceedings. The following listing presents these publications in reverse chronological order:

- Faieq, S., Saidi, R., El Ghazi, H., Front, A., & Rahmani, M. D. (2021). Building adaptive context-aware service-based smart systems. Service Oriented Computing and Applications, 15(1), 21-42.

- Dongo, J., Faieq, S., Panta, F. J., & Polacsek, T. (2020). Vers un framework de composition de services sensible au contexte pour les environnements intelligents. INFORSID 2018 Forum Jeunes Chercheuses Jeunes Chercheurs. Open Journal in Information Systems Engineering, 1(4).

- Faieq, S., Front, A., Saidi, R., El Ghazi, H., & Rahmani, M. D. (2019). A context-aware recommendation-based system for service composition in smart environments. Service Oriented Computing and Applications, 13(4), 341-355.

- Faieq, S. (2018, May). Vers un framework de composition de services sensible au contexte pour les environnements intelligents. In INFORSID-Forum des Jeunes Chercheurs Jeunes Chercheuses.

- Faieq, S., Saidi, R., Elghazi, H., & Rahmani, M. D. (2017). C2IoT: A framework for Cloud-based Context-aware Internet of Things services for smart cities. Procedia Computer Science, 110, 151-158.

- Faieq, S., Saidi, R., Elghazi, H., & Rahmani, M. D. (2016, May). A conceptual architecture for a cloud-based context-aware service composition. In International Symposium on Ubiquitous Networking (pp. 235-246). Springer, Singapore.

1.4 Case Study

The present case study and the underlying motivating scenario are derived from the *SMARTROAD* projet, which is a cooperation project funded by CAMPUS FRANCE (PHC TOUBKAL 2017 (French-Morocco bilateral program) Grant Number: 36804YH), aiming to create a platform, a method and a set of design tools to develop dynamically composed services in the context of smart roads.

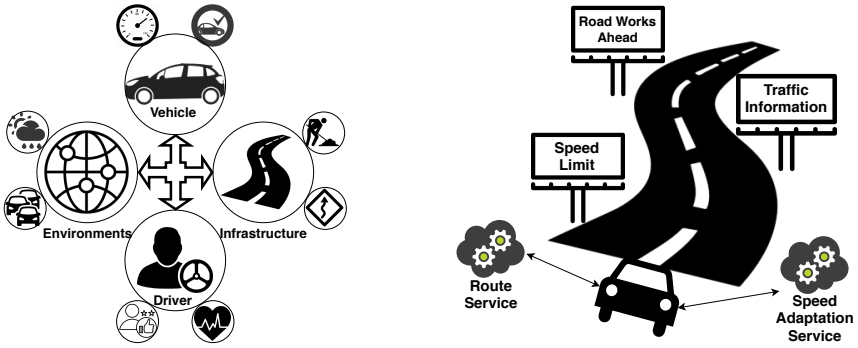
There are 1.35 million human deaths related to traffic accidents worldwide. According to the World Health Organization, the majority of this number is registered in Africa (WHO, 2018). In Morocco for example, the road safety situation has been described as a “road war", with close to 82 000 accidents in 2016, an increase of 3.8% when compared to 2015, which is mainly due to the rise of the number in vehicles and automobile industry in the country (Prevention of Traffic Accidents (Morocco), 2016). This number of accidents has effects on the economy, environment, society and mobility and a similar situation is also noticed in the rest of the Arab Maghreb Union and developing countries in general. In this regard, Morocco had launched the national strategy for road safety 2016-2025, aiming at halving the number of deaths in road accidents.

Road safety, however, is only one of the issues related to the surface transport landscape. In fact, transport is at the heart of any nation's continuous and sustainable development. From an organizational perspective, different stakeholders are involved and each one has a particular perspective in the planning, building , managing, using and analyzing of the transport landscape (e.g., Policy-makers, vehicle suppliers, energy providers, service providers, end customers, etc.). Any effort to reduce the number of accidents and deaths on the road has to deal with the pre-crash phase and the post-crash phase with respect to the different actors in the transport ecosystem. In this thesis, we focus on the pre-crash phase, where the goal is to manage and minimize the risk factors related to each actor using the services provided by the different stakeholders. We are interested in the services provided to the End-customers by the service providers and city authorities. Table 1.1 presents some examples of the services that can be offered to the smart city inhabitants, their service provider and their supporting entity (J. Chang et al., 2017), (Ait-Cheik-Bihi et al., 2012).

Acronym	Service Name	Stakeholder
AEVW	Approaching Emergency Vehicle Warning	Emergency Services
CBW	Car Breakdown Warning	Automobile Industry
RMS	Road Monitoring Service	Road Operators
VSMS	Vehicle State Monitoring Service	Automobile Industry
ISA	Intelligent Speed Adaptation	Automobile Industry
IVS	In-Vehicle Signage	Automobile Industry
TJAW	Traffic Jam Ahead Warning	Road Operators
ERP	Electronic Road Panel Service	Road Operators
RWW	Road Works Warning	Road Operators
WWS	Weather Warning Service	Weather Services
SMSD	Short Messaging Service for Driver	Automobile Industry
RSS	Route Selection Service	Automobile Industry
ERS	Emergency Rescue Service	Emergency Services
NAAS	Nearby Area Alarming Service	Road Operators
TSS	Towing Selection Service	Navigation Services
OCS	Oil Calculation Service	Automobile Industry
GSS	Garage Selection Service	Navigation Services
GSSS	Gas Station Selection Service	Navigation Services
HSS	Hospital Selection Service	Health-care Services

Table 1.1 – Examples of provided services

To effectively deal with the potentially problematic situations (e.g., traffic jam, bad weather, etc.) that may arise on the roads and their potential causes (In this thesis, we call them *Risk Factors*), several of the services presented in table 1.1 have to be composed. We focus on road safety by defining situations that can be considered dangerous and the risk factors that can contribute to the occurrence of these dangerous situations. We consider that each risk factor can cause a dangerous situation. These risk factors are related to the different entities involved in the road transport ecosystem (i.e., vehicle, driver, etc.) and can generally be monitored through sensors that can be accessed via exposed services (e.g., speed, driver health and weather). Figure 1.2a shows these entities and their related risk



(a) *SMARTROAD*: Actors and Risk Factors

(b) *SMARTROAD*: Example services offered to the driver

Figure 1.2 – *SMARTROAD*: Actors, Risk Factors and Services

factors, as defined in this case study, while figure 1.2b shows a vision of the *SMARTROAD* ecosystem.

Motivating scenario

Consider a smart system that needs to operate on a road transport environment to improve security $ss_{smartroad}$. Let u_{driver} be a human user driving his smart car u_{car} on an open road u_{road}. To identify the situation $st_{riksySpeed}$ where u_{car} is exceeding the speed limit set on u_{road}, the system needs to know the speed limit currently applicable in u_{road} and the current speed of u_{car}. We define these bits of information as context and identify $c_{roadSpeedLimit}$ and $c_{vehicleSpeed}$ as the context dimensions representing respectively the speed limit on u_{road} and the vehicle speed of u_{car}.

If and when the situation is detected, the goal then becomes that of responding to the situation $st_{riksySpeed}$. Let $g_{riskySpeedResponse}$ be that goal, which aims to inform the driver about his situation to try and mitigate its risk and to inform the neighboring vehicles about the potential risk so they are more vigilant. To that end, the system uses available business services and coordinates their execution so as to accomplish the prescribed goal. For instance, the system uses the ISA (Intelligent Speed Adaptation) service sr_{isa} at the infrastructure level (i.e., u_{road}) to compute an optimal speed for the neighboring vehicles so as to minimize any potential risk of collision and the DNA (Dangerous Nearby Area) service sr_{dna} to compute a safe zone around the risky vehicle. Then, the system uses the IVS (In-Vehicle Signage) service sr_{ivs} provided by the risky vehicle to inform the driver of his situation and eventual steps to take to resolve the situation (e.g., reduce the speed of the vehicle).

After some time, while doing a report on road security, the stakeholders notice that most of the vehicles involved in accidents have been flagged by the previously described system. Mostly because the speeding vehicles did not slow down. They decide to work with the police force to intervene in cases where the $c_{vehicleSpeed}$ does not slow down after some time. To make this happen, the system needs to add the service sr_{pp} to inform the police for them to intervene in the $g_{riskySpeedResponse}$. Furthermore, some analysts suggest that the speed limits on several roads should be further decreased due to the high number of accidents witnessed on these roads.

1.5 Dissertation Outline

This dissertation is organized in seven chapters. The current chapter represents Chapter 1 and describes the context, motivation, methodology, objectives and contributions of this research. The rest of this document is structured in two main parts, namely the state of the art and the contributions. Part I provides a foundation to read and understand Part II while an epilogue concludes the dissertation.

Part I presents a review of the literature and sets the scene for our contributions. It is composed of two chapters:

- In Chapter 2, we cover and present the main concepts and definitions necessary for the reader to understand the topic. At first, we present various paradigms and methods that influence the design and development of SSs. Then, we present the technological landscape surrounding the rise of the SS paradigm. Last, we explore the improvement techniques that were used in literature to make software systems adaptable in both design and technological aspects.

- In Chapter 3, selected related works that have reported on the design and development of smart systems have been analyzed according to the three aspects and dimensions covered in the previous chapter, namely the design concepts used, the technological enablers leveraged and the improvement approaches adopted or developped in each approach.

Part II presents the contributions of this thesis and is composed of three chapters:

- Chapter 4 describes our approach for designing adaptive service-based SSs. In the proposed approach, we present the Smart System Loop as a functional

model for SSs. Then, we present a general view of AS3 as an intentional method to design SSs which covers their capabilities, enabling technologies and improvement aspects.

- Chapter 5 discusses the elements representing the capabilities of SSs and their enabling technologies through a product metamodel and proposes a manner through which system engineers can proceed to define these elements in a targeted SS via process models. A rundown of defining the elements of the *SMARTROAD* case study is also provided to help illustrate the workings of AS3.
- Chapter 6 discusses the process of improving SSs through recommendation techniques. We propose and discuss some techniques and algorithms allowing the improvement of SS at different levels according to (i) the elements constituting the SS, (ii) the relationships between the elements and (iii) the technological resources used by the SS. A rundown of discovering the improvements for the *SMARTROAD* case study is also provided to help illustrate how recommendation techniques can be used in AS3.

Chapter 7 presents an epilogue for this dissertation where we summarize and discuss the conclusions of this thesis and anticipate several future work directions.

Part I

State of the Art

Chapter 2

Background and context

History is merely a list of surprises. It can only prepare us to be surprised yet again

Kurt Vonnegut

In this chapter, we aim to present the concepts and principles related to Smart System (SS)s, to their design and to their development. We start the chapter by introducing key concepts and definitions related to the topic in 2.1. We, then, present some of the prominent approaches taken by researchers and practitioners to design and develop software systems in general and smart systems in particular in section 2.2. Afterward, in section 2.3, we present the technological landscape that paved the way to the rise of Smart Environment (SE)s as we discuss the role of technologies like Cloud Computing (CC), Internet of Things (IoT) and Big Data in SSs. Last but not least, we present how improvement techniques and approaches were used to make SSs adaptable, both from a design perspective and from a technological and operational one in 2.4.1 and 2.4.2 respectively. The goal of this chapter is to build a knowledge base that defines and introduces the concepts used throughout this thesis.

2.1 Smart Systems: Definitions and Background

Before venturing in the analysis of the literature regarding SSs, it is important to define what an SS is and by extension what is an SE. As mentioned in the section 1.1, SEs are complex ecosystems where several components interact to the betterment of the quality of their existence. SSs, as the systems that operate on

SEs, have attracted the attention of researchers and practitioners from various backgrounds and interests. This is mainly due to the fact that they provide a wide range of real world applications and implications but also to the various technologies and approaches that need to be put in place to successfully set them up. Indeed, SSs cover a large spectrum of areas from sensing technologies (e.g., Radio Frequency IDentification (RFID) tags and readers, Sensor Network System (SNS), actuators, etc.) and communication protocols (e.g., Infrared, Bluetooth, WiFi, 3/4/5G, etc.), passing by Human-Machine Interaction (e.g., voice command, user interfaces, etc.) to Artificial Intelligence (e.g., face recognition, fall detection, driver drowsiness, etc.). Due to this interdisciplinarity, SEs and thus SSs, have varying definitions depending on the background and interests of the researchers and practitioners, and the goals of their work. Hence, there is no consensus over a standard definition of what is an SS or what is an SE.

Table 2.1 summarizes various definitions and descriptions found in the literature regarding SSs and SEs. The table also presents the influences (i.e., background) of the authors and the source's venue (i.e., scope of the journal or conference). The selected definitions describe smart systems and environments in general without focusing on a particular domain application. Note that an SE in this manuscript is not to be confused with "Smart Environment" as a Smart City dimension that focuses on environmental issues like air and water pollution, park management and waste management (Khatoun and Zeadally, 2016). So far, from a software system engineering perspective, the most complete definition was given by the National Science Foundation (NSF) (Foundation, 2014, p. 5). Although, the definition concerns and describes Service-based Smart System (SSS)s, we argue that the definition remains accurate for all SSs. The definitions states that:

> *A "smart" service system is a system capable of learning, dynamic adaptation, and decision making based upon data received, transmitted, and/or processed to improve its response to a future situation. The system does so through self-detection, self-diagnosing, self-correcting, self-monitoring, self-organizing, self-replicating, or self-controlled functions. These capabilities are the result of the incorporation of technologies for sensing, actuation, coordination, communication, control, etc. The system may exhibit a sequence of features such as detection, classification, and localization that lead to an outcome occurring within a reasonable time.(Foundation, 2014, p. 5)*

However a key missing component in this definition, although present in the definitions influenced by Ubiquitous Computing (UbiComp), is the value of the

Sources	Definitions	Influences
(Cook and Das, 2007)	*A smart environment is one that is able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience in that environment*	Pervasive Computing, Artificial Intelligence
(X. Chen, L. Wang, et al., 2016)	*The smart environment can infer action from people's context and then influence collective behaviour of individuals in the environment*	Heath Informatics, Optimization and Scheduling
(Machado et al., 2014)	*The term ubiquitous smart system is utilized to characterize applications that are able to perceive the user context and properly react, according to the occurrence of specific events*	Conceptual Modeling, Ubiquitous Computing
(Durães et al., 2018)	*a smart environment is a digitally augmented physical world where sensor-enabled and networked devices work continuously and collaboratively to make the lives of the inhabitants more comfortable*	Artificial Intelligence, Distributed Computing
(Oltean et al., 2013)	*Smart service systems may be intended as service systems designed for a wise and interacting management of their assets and goals and capable of self-reconfiguration in order to perform enduring behavior capable of satisfying all the involved participants in time*	Service Science, Management
(Ribino et al., 2016)	*Smart systems aim at augmenting real environments to create smart spaces where users are provided with pervasive electronic devices. Usually each device can provide a set of services and functionalities. A smart system connects such electronic devices into a network and control them by using advanced Information and Communications Technology (ICT) technologies in such away the devices satisfy user requirements*	Human-Machine Interaction, Software Engineering
(Cicirelli et al., 2018)	*Smart Environments (SEs) are open and dynamic systems typically extending over a wide area and including a huge number of interacting devices with a heterogeneous nature*	Internet of Things, Edge Computing
(Jara et al., 2015)	*An integrated smart system is just like a human who has his own sensing systems, nervous system, store system, and his own brain for decision-making*	Internet of Things, Cyber-Physical Systems

Table 2.1 – Definitions and descriptions of SS and SE in literature.

system for its users. Indeed, to be smart, we argue that an SE needs to be capable of considering both its own state and behavior as well as those of its users. For this reason, we formulate our own definition of SE, taking a more abstract view over their characteristics. The definition is the following:

> *"Smart Environments are virtual extensions of the physical world, that are capable of providing customized services to their users when required and adapting to their changing needs"*

2.1.1 Smart Systems vs Intelligent Systems

In most of the literature, Smart Systems are used analogously and almost interchangeably with *Intelligent Systems*. This confusion isn't exclusive to computing paradigms but also extends to common linguistics as most dictionaries use both adjectives synonymously. Although some researchers tried to distinguish between the two paradigms, we are still far from a complete discriminatory distinction. For example, while the authors in (Augusto et al., 2013) claim that *Intelligent Environments* are in fact SEs integrating Ambient Intelligence and based on ubiquitous availability of services, the authors in (Wolter and Kirsch, 2017) claim that smart, intelligent and cognitive represent different facets of a system's capabilities. Others insinuate that *smartness* is a higher form of *intelligence* (David, 2011).

In this thesis, we decided to use *smart* instead of *intelligent* as we consider that smartness is more related to the objectives of our research work. Table 2.2 summarizes the arguments for the former decision. As depicted in the table, the decision is based on four criteria which are:

1. The cognitive functions supported by the system category. This criterion denotes the capabilities to simulate the cognitive functions of human beings in sensing, reasoning and acting on their environment.

2. The intelligence types integrated by the system category. There are two types of intelligence, (1) Computing Intelligence indicates the ability of computers to reason and learn from their experiences, and (2) Human Intelligence indicates the ability of humans to take advantage of the functions offered by computers and make informed and educated decisions.

3. The main characteristic of the system category. While there are many common and overlapping characteristics related to both system categories, this criterion represents the forefront characteristic that each category strives to achieve.

4. The scope or view of the system category. The scope or the view represents the focal point (i.e. actor) taken by each category. A system-centered view is more concerned with the system's performance while a user-centered view is more concerned with the satisfaction of the user.

*SS*s are more cognitive than *Intelligent Systems* as the former address the whole cognitive cycle (i.e., sensing, reasoning and acting) while the latter focus on the reasoning function. Moreover, while *Intelligent Systems* only integrates Computing intelligence, SSs integrate both Human Intelligence and Computing Intelligence in their operating functions to perform their tasks. These system categories

Criterion	Intelligent Systems	Smart Systems
Cognitive function	Reasoning	Holistic
Intelligence types	Computing Intelligence	Human Intelligence And Computing Intelligence
Main characteristic	Autonomy	Adaptability
Scope	System-centered	User-centered

Table 2.2 – Characteristics of Smart Systems vs Intelligent Systems

also have different priorities in term of their forefront characteristic. While *Intelligent Systems* focus on Autonomy as their defining characteristic, SSs focus on adaptability in being capable of detecting and responding to emergent behavior. Finally, the views taken by both system categories also vary. Generally, SSs take a user-centered view in comparison to a more system-centered view in *Intelligent Systems*.

2.1.2 Smart Systems as a Complex Adaptive System

Holland, in (Holland, 2006), defined Complex Adaptive Systems as "systems that have a large numbers of components, often called agents, that interact and adapt or learn". There are numerous examples of Complex Adaptive System (CAS)s. They can be natural (e.g., the climate, the brain, the immune system, etc.) or man made (e.g., social networks, markets, the Internet, etc.). However, even though they differ greatly in their states and behaviors, Holland argues that all CASs share four (4) major features (Holland, 2006).

1. *Parallelism*. The author claims that the components of every *CAS* are capable of sending and/or receiving signals (i.e., data) thus producing a big amount of signals as they interact simultaneously with each other. Hence, these interactions need to be coordinated at both the component level and the system's level to insure the smooth operation of the system.

2. *Conditional actions*. Each component in a CAS can reason and act on the signals receives. The component thus develops a set of complex conditional rules that dictates its behavior regarding the events that occur in the system's state and by extension the state of its other components. The action taken by a component can be another signal that would affect other components receiving it.

3. *Modularity*. To be able to react to different situations, a component tends to rely on the enactment of a sequence of conditional rules. The combination of these rules leads to the construction of different compositions that may perform differently depending on the situation. These compositions and their underlying rules provide modular solutions to respond to future and emerging situations.

4. *Adaptation and evolution*. The components of a CAS change overtime to address the problems that arise along the process of the system's evolution. According to the author, those changes are usually adaptations to current state of the components that improve the performance of the component and/or the system, rather than random variations.

The features discussed above are clearly aligned with the characteristic of SSs and their underlying SEs that we discussed in section 1.1. Thus, in this manuscript, we consider SSs to be a type of CASs. This categorization constitutes the basis of our approach to building SSs. It allows us later on to extract the capabilities of SSs and to select the paradigms and concepts that would guarantee the fulfillment of their features.

2.2 Software Design and Development in Smart Systems

Several techniques and paradigms have been proposed and used by software engineers to build software systems. Some of these techniques and paradigms have known more success and adoption than others depending on the different communities. In this section, a brief introduction into some of these techniques is given. The hereafter mentioned techniques and paradigms have been selected due to their relevance to different facets of SSs discussed in the previous section (see section 2.1). Hence, we focus the discussion of these approaches on the characteristics of SSs rather than a general discussion of their use.

2.2.1 User-centered Software Design

For a long time, traditional software design and engineering have been focused on meeting the functional requirements of their systems (Kling, 1977). Little by little, as software was mainly being developed to help workers achieve their daily tasks, researchers and practitioners began to notice that software systems evolve in complex ecosystems that also involve human factors as well as external (i.e., environmental) factors. This observation sparked a new line of thought into the role and impact of human factors in the usability of software systems. As a

result, several approaches were born promoting the human user as the centerpiece around which software systems are built.

Later on, researchers started to think about human factors as part of broader notion, which is that of *context*. In this perspective, context was considered to be the environment that the software system will fit in and under which it will operate (Karat, 1997). However, most of the efforts on the integration of human factors in system design and development was focused on the experience the user has interacting with the system, and this interaction was mostly visual (i.e., through some kind of Graphical User Interface (GUI)) in nature (Card et al., 1983). This has ignited the research on software usability which later on marked the interplay between Software Engineering and Human-Computer Interaction (HCI) under the umbrella word of User eXperience (UX).

With the advances in technology, new interaction modalities have been put forth (e.g., microphones/speakers, touchscreens, etc.). Moreover, due to the rise of novel computing paradigms such as Artificial Intelligence (AI), software system users nowadays are not just humans but can also be other autonomous systems. These paradigm changes laid the foundation for the creation of more complex software ecosystem. Thus, classical approaches to software engineering were rendered almost useless, if not to being a stepping stone towards approaches that are suitable for these new and complex ecosystems.

In this thesis, we focus on the bigger picture of user involvement in software. We consider both humans and autonomous systems as potential users of an SSs. This reflects directly on the way systems are conceived and built. Referring back to our motivating scenario in section 1.4, the *car* is considered as a user of the *SMARTROAD* system as well as the *driver* of that car. In this scenario, two types of interaction are happening. First, a human-machine type of interaction between the *driver* and the *car*, and between the *driver* and the *SMARTROAD* system. Second, a machine-machine interaction type between the *car* and the *SMARTROAD* system.

The previous distinction between machine users and human users made us explore a new line of thought and it implies that design approaches need to and should consider both types of users and interactions at the same time and as first class citizens. Hence, the design process should include thinking about the data regarding their states and behaviors. While the states and behaviors of software systems are quite predictable at some point in time, it is much more challenging to pin down or accurately predict those of human users. Hence, the need for a conceptual framework that facilitates the capture of their requirements and preferences at all time.

Another important factor to consider when thinking about SS users is how the interactions between the users, of both human and machine types, help the SS in offering functions that are more tailored to their needs. Indeed, in our motivating scenario (see section 1.4), the *SMARTROAD* system can access the interactions between the *driver* and the *car* through the information recorded by the *car* on these interactions. The *SMARTROAD* system then is able to know what information and services should be handled autonomously and automatically by the *car* and what information and services should be passed to the *driver*. In time, these interactions help the *SMARTROAD* system to tailor the information to a more fine grained level, differentiating not only between human and machine users but among human users themselves and among machine users themselves.

Being highly complex, it is hard to define clear boundaries where SSs operate (Lewis et al., 2007). Indeed, SSs usually involve several stakeholders and span across multiple domains, as is the case in Intelligent Transport System (ITS)s. The blurry boundaries make developing a coherent view of what the system is and what it should do very challenging. This results in a fragmented and subjective landscape surrounding the system where each participant has his own view of the system depending on his role within that system. This subjectivity was highlighted in Lyytinen's framework (Lyytinen, 1987) in showing how different Development Groups perceive and act upon different Object Systems (facets) of the same target system, as can be seen in figure 2.1 and this problem only got more pronounced as systems grew more complex. Each stakeholder (or representative of a stakeholder), as he perceives the system, interacts with the SS in a unique manner and expects the SS to behave a certain way without considering the ramifications on the SS as a whole. Considering these interactions is essential for the system to be 'usable' by the different stakeholders and for the system to offer them the added value they expect.

2.2.2 Context-aware Engineering

What is context ? Ironically enough, the answer to this question would depend on the context surrounding the individual trying to answer it (e.g., his background, knowledge, field, expertise, etc.). Indeed, *Context* is a very broad concept. As humans, we are trained to capture context information through our cognitive system and interpret it to the best of our ability to extract some knowledge or meaning and/or grasp the situation in which we find ourselves (Barsalou, 1982). Linguistically, a phrase that is taken out of its context can lose or change its intended meaning completely. A fact that malicious individuals often exploit to

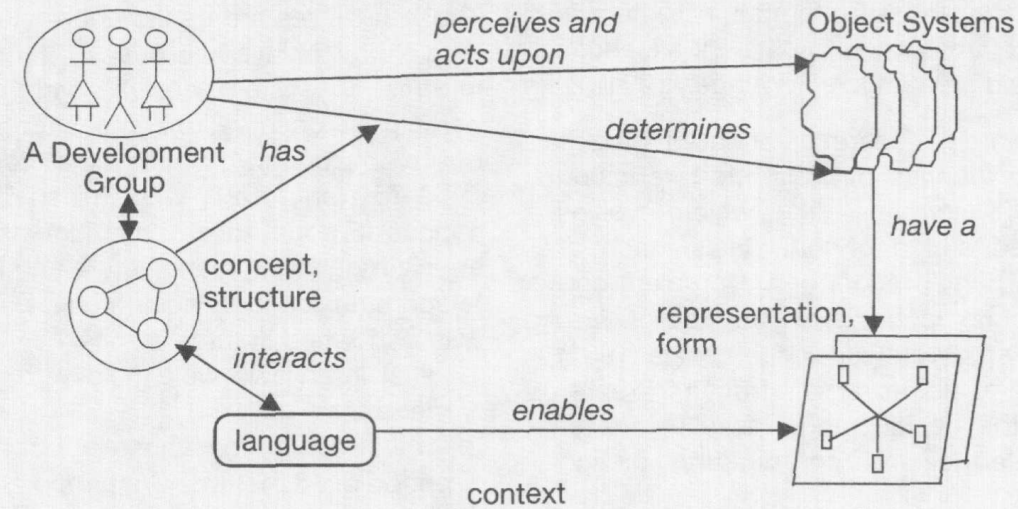


Figure 2.1 – Target Object Systems in the development process (Gasson, 1999; Lyytinen, 1987).

spread false information on social media and other platforms [1].

In computing, context is as important as it is to humans and language since most if not all of computing paradigms are modeled after these two. Though researchers have recognized its importance, especially in UbiComp (Sundar et al., 2013), they still could not agree upon one definition for context (Coutaz et al., 2005). One of the most accepted definitions is the one formulated by (Abowd et al., 1999), and it states that:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

This definition has the advantage of being generic and does not try to enumerate the dimensions that constitute the context; a trend that was quite common in the early days of Context-Aware Computing (CAC) in different fields like mobile networks (Schilit and Theimer, 1994) and archaeology (Ryan et al., 1998). Instead, it proposes a general view of what context is and allows engineers to think about context information that is relevant to the target application and domain. However, this definition is still limited to the interaction between humans and machines (i.e., applications) and does not include machine-machine interactions. Context-awareness is thus the ability of a system to capture, interpret and act upon context information to enhance its capabilities to better assist and serve its users.

[1] https://en.wikipedia.org/wiki/Quoting_out_of_context . Accessed on the 21st of March 2021

Engineering context-aware systems is a process that usually includes a context data life cycle. In literature, this life cycle is composed of three to six circular steps where the goal is to explain how context information is managed in the system. In their paper on the role of context-awareness in the IoT, (Perera et al., 2014) discussed and nicely summed up several of these life cycles. The activities involved in these life cycles include the capturing, sensing, transmitting, disseminating, managing, storing, processing, reasoning, using, maintaining, etc. of context information. The authors in (Perera et al., 2014) reassembled all of these activities, extracted four essential phases (i.e., steps) and put them together in what they called "the simplest form of a context life cycle" as illustrated in figure 2.2. *Context Acquisition* phase deals with the problems related to the channel or medium through which the data is transmitted between the source and the destination. *Context Modeling* includes the processes of how the data is represented and stored. *Context Reasoning* represents the different ways through which the context information is processed to extract higher level information and eventually knowledge. *Context Dissemination* explores how context data and the subsequently inferred information and knowledge can enhance or assist the system users in their tasks.

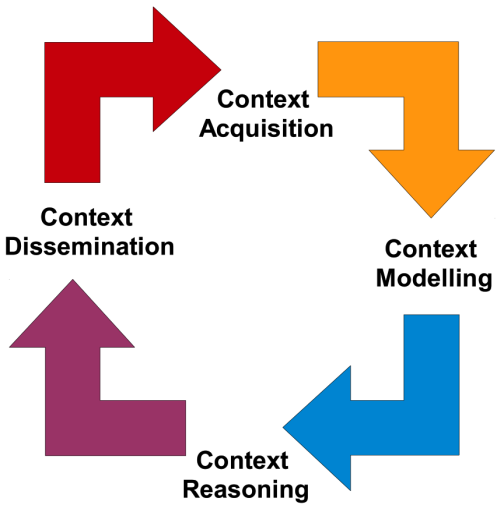


Figure 2.2 – The simplest form of a context life cycle (Perera et al., 2014).

An important aspect that has been mostly ignored in literature related to context-aware systems is how systems can help in optimizing the context life cycle. Indeed, traditionally context information is used to improve or/and optimize the functionality of systems, whether to filter unnecessary content, understand user intention or propose relevant content and service. But, the manner in which the systems can be built proving an environment where the context life cycle can be optimized is virtually unexplored. Thus, the need for such a contribution is essential because some types of context information can have more impact and

significance than others, depending on the application. Hence, it is necessary to analyze the impact of each type of contextual information on the performance of the systems to be able to leverage contextual awareness effectively and efficiently (Pinheiro and Souveyet, 2018). Focusing on the context information that brings the most value also reduces the amount of data to be stored and processed which reflects positively on the performance of context-aware systems.

2.2.3 Service Oriented Architecture

Software system development has known many methodologies over the course of almost 70 years. From structured programming in the early 1950s to aspect-oriented development, passing by Object-Oriented Design (OOD), Business Process Management (BPM) and Component-based Design (CBD); all of which impacted the software industry and academia in various ways and came to address specific concerns in software development. The most recent methodology is that of service-orientation and microservices as part of the Service Oriented Computing (SOC) paradigm (Newman, 2015; Papazoglou and Georgakopoulos, 2003). Though, unlike past development methodologies, service-orientation came as a part of a whole computing paradigm, it depicts in great detail the architectural and design considerations to be taken when developing service-oriented systems.

Since its inception, SOC has proven itself as a paradigm for improving the agility of organizations, as well as facilitating intra- and inter-organizational cooperation through the use of design principles such as reuse, modularity, virtualization and composition (Legner and Heutschi, 2007). Organizations (private and public) are increasingly looking for ways to capitalize on the wealth of available data through the advances in technology to improve their service offerings (Varadan et al., 2008). At the same time, user expectations regarding the relevance and quality of the offered services are getting higher and higher. This is particularly true when they're performing complex and context-dependent tasks, such as travel planning which involves many activities (e.g. finding and booking accommodation, transport, interesting events, etc.) and where the user has to sift through a large amount of information (offers) and make many decisions to achieve his or her goal.

As the name suggest, *services* are the main unit in SOC. Services are modular, autonomous, loosely coupled and self-describing, and they encapsulate the business logic serving one or several particular tasks. The complexity and operational scope of a service can be fine or coerce grained depending on the design choices made by system engineers and domain experts. In a conventional architecture built on services, there are usually three roles that interact with these services.

Like any offer-demand policy, there are *service providers* and *service consumers*. *Service providers* are responsible for managing the service (e.g., it's availability, efficiency, visibility, etc.), while *service consumers* search for appropriate services to achieve the task they need. The third role is that of the *Service Broker* responsible for collecting the available services and structuring them in a way that facilitates finding them by the consumers. Figure 2.3 shows the roles, connections and operations in a typical Service Oriented Architecture (SOA).

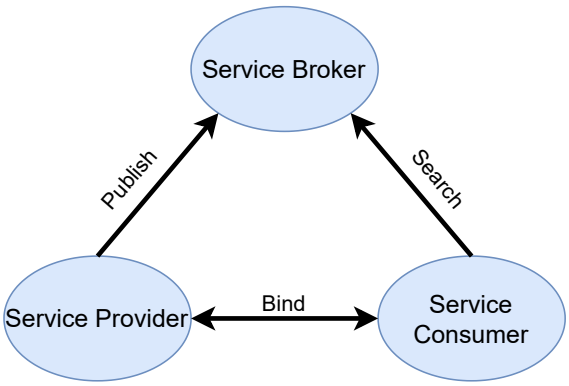


Figure 2.3 – Roles, connections and operations of a typical SOA

The functional scenario in SOA can be summed up in four (4) steps. First, each *service provider* publishes the set of operations it offers via sharing its interface, which is a document containing its functional and non-functional details and how and where to communicate with it. Second, the *service broker* collects the service interfaces and indexes them based on some agreed upon criteria (i.e., domain, functionality, quality of service, etc.) in a convenient structure, such as a database, a directory or an inventory. The *Service Consumer* is then able to search the database for the service he requires through keywords. Once the *Service Consumer* has found the service he needs, he can bind to the particular service by following the access protocol specified on the interface of the service, and then invokes any particular operation that the service allows him to make.

The power of service-orientation, as a design paradigm, lies mainly within its design principles. Thomas Erl (Erl, 2009), argues that there are eight (8) key design principles that need to be addressed to achieve "real" service-oriented solutions. He defines these design principles as follows:

- *Standardized Service Contract:* Services within the same service inventory are in compliance with the same contract design standards.
- *Service Loose Coupling:* Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment.

- *Service Abstraction:* Service contracts only contain essential information and information about services is limited to what is published in service contracts.
- *Service Reusability:* Services contain and express agnostic logic and can be positioned as reusable enterprise resources.
- *Service Autonomy:* Services exercise a high level of control over their underlying runtime execution environment.
- *Service Statelessness:* Services minimize resource consumption by deferring the management of state information when necessary.
- *Service Discoverability:* Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted.
- *Service Composability:* Services are effective composition participants, regardless of the size and complexity of the composition.

Over the years, SOA has had a big impact on software development as several technology stacks implemented its principals, whether it be fully or partially. Among these technology stacks, Web services have gained the most traction and are nowadays widely used in distributed systems. Web services rely on web communication and access protocols such as HyperText Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST) to receive requests and send responses in the form of eXtensible Markup Language (XML) or JavaScript Object Notation (JSON) documents. The reliance on Web standards and protocols allows for a platform and language independent access and exploitation of the offered services. The use of standard interface description documents (i.e., contracts), expressed mainly either through Web Service Definition Language (WSDL) or Web Application Description Language (WADL), allows to ensure the functional, technology, logic and sometimes quality abstractions of the service. Composability is ensured through business process-like orchestrations using Business Process Execution Language (BPEL) or coded manually in the form of service or Application Programming Interface (API) mashups.

2.3 Technological enablers of Smart Systems

The unprecedented level of urbanization and the human's incessant pursuit of a better quality of life has stretched the limited resources of the planet too thin.

According to United Nations (UN)'s report on the world's urbanization prospects, published in 2014, more than 70% of the world's population will be living in cities by 2050 (Nations, 2014). The concept of SEs came about as a possible approach to optimize resource utilization to cater to their scarceness, while at the same time, keeping the inhabitant of the users happy by improving their quality of life within these environments. To that end, the concept of SEs argues that the integration of ICT into the user's everyday life could achieve the aforementioned goals. The role of these SEs is to automate when and if possible some activities that their users were traditionally forced to perform, and help them make decisions and/or assist in regard to performing the activities and tasks that can't be automated.

The integration of ICT in all aspects of everyday life assumes the existence of a technological infrastructure that is equipped to handle the different facets involved in the data-based decision making process. From the computing perspective, this process can be broken down into five (5) major computing tasks revolving around data as follows.

1. *Creation*: The creation of data delineates the process by which the data is either produced, transformed or acquired in a machine readable format. This process involves the handling of uncertainties, trust, security and other quality attributes related to the sources of the data (e.g., sensors, mobile devices or system databases).

2. *Processing*: The processing of data describes the algorithmic operations that are performed on the data to derive higher level information, and/or discover and learn patterns in the data. These operations mainly include the fusion, extraction and transformation of raw data (Bourque and Clark, 1992). Processing is thus at the heart of every digitized information and/or software system.

3. *Storage*: Data storage is an important aspect when dealing with data in general and more so with digitized data. This is mainly owing to the huge amount of data transiting nowadays in intranets and the internet (estimated by IDC[2] at thirty-three (33) ZB in 2018 (Reinsel et al., 2018)). Data storage has to deal with structuring the data to make the retrieval, writing, modification and removal of data fast enough to handle the requirements of real-time applications.

[2]https://www.idc.com/about

4. *Transmission*: SEs assume the existence of highly, if not fully, connected infrastructure where the different components of the environment can communicate data. To that end, the integration and interoperation between wired or wireless connections using different communication protocols at different frequencies and speeds, while ensuring the integrity and confidentiality of sensitive information need to be achieved.

5. *Dissemination*: Dissemination represents the manner in which information and data are offered to interested parties (i.e., users). Indeed, the offered information can take many forms and is usually specific to the user's device and/or available devices in his vicinity. Chat-bots, GUIs or digital files (i.e., multimedia content), among others, can all be employed for such an endeavor

The successful realization of SEs relies thus on technologies, processes and paradigms that can enable the execution of the previously mentioned tasks. However, in the last two decades, it was hard to keep track of the huge number of technological trends and innovations, especially in the past decade. Each one with its own set of promises and assurances, it was also hard to distinguish facts from illusions surrounding them. To address this issue, Gartner[3], a global research and advisory company, annually publishes a hype cycle to position recent and emerging technologies, as in the example of the 2015's hype cycle provided in figure 2.4.

Recurring technologies on Gartner's hype cycle throughout its many versions were ones that have had great impact on the IT industry such as CC, IoT and Big Data. These technologies constitute the focus of this technological overview relating to SEs. Indeed, several research and industrial solutions have used these technologies to develop SSs (Santana et al., 2017). In their Smart City report, the International Organization for Standardization (ISO)[4] have listed the IoT, CC and Big Data as important technological enablers to the implementation of the smartification trend (JTC-1, 2014). The adoption of these technologies, although somewhat erratically, was also supported by different research projects and industrial applications in the context of SEs in general and Smart Cities specifically (Hashem, V. Chang, et al., 2016; Jara et al., 2015; Khan et al., 2013; Petrolo et al., 2017; Santofimia et al., 2018). In the following, we look at these three (3) technologies from an SE point of view to understand their underpinnings and analyze their individual advantages, and study their collective synergies.

[3]https://www.gartner.com/en
[4]https://www.iso.org/about-us.html

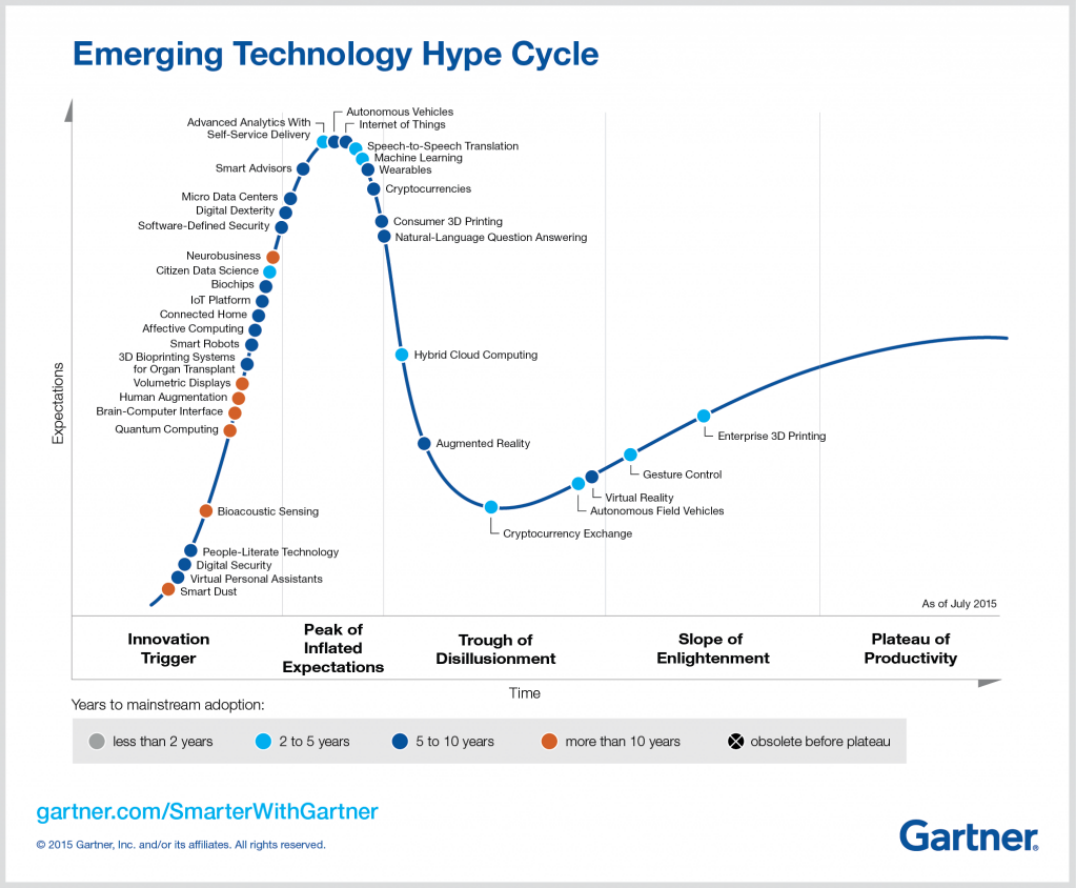


Figure 2.4 – Gartner's hype cycle for emerging technologies in 2015 (Walker and Burton, 2015).

2.3.1 Internet of Things

Since its inception and proliferation into our lives, the Internet has sparked numerous innovations and paradigm shifts (e.g., the world wide web (Hall and Tiropanis, 2012)) to get to where it is today and continue its evolution towards the Internet of the future. One of the driving forces of this evolution is what is called the IoT. As a matter of fact, the IoT has attracted the attention of business managers and engineers from several subareas in computer science, telecommunications, as well as service science and management. These differences in backgrounds and thus in perspectives are translated in the variety of definitions that exist today for this technology, and that tend to showcase the specific characteristics of the IoT that are of interest to the authors (S. Li et al., 2015). The most comprehensive definition of the IoT and the one adopted throughout this manuscript came from the authors in (Minerva et al., 2015) and states the following.

> *"Internet of Things envisions a self-configuring, adaptive, complex network that interconnects 'things' to the Internet through the use of standard communication protocols. The interconnected things have physical or virtual representation in the digital world, sensing/actuation capability, a programmability feature and are uniquely identifiable. The representation contains information including the thing's identity, status, location or any other business, social or privately relevant information. The things offer services, with or without human intervention, through the exploitation of unique identification, data capture and communication, and actuation capability. The service is exploited through the use of intelligent interfaces and is made available anywhere, anytime, and for anything taking security into consideration."*

This definition is generic and goes beyond singular perspectives regarding IoT. It clearly states the characteristics of IoT, its underlying infrastructure, communication paradigm, the offered services and the way applications are built using these services. In the following, we dive deeper into these concepts to construct a complete view of what the IoT represents to us.

2.3.1.1 Sensing and actuation

Sensing and actuation constitute the basis of any cognitive system. Much like humans, software systems rely on data to perform their functionality, whether these data come from manual entries performed by human users or from automatic readings from sensors. The word 'sensor' here is an umbrella word to designate any physical or virtual element that can measure some metric or indicator. Hence, sensing represents the capability of a system to measure and collect data upon which the system's logic is going to be applied. In the same way, actuation represents the capability of a system to effect change within its action range (e.g., changing the state of the system itself, changing a physical property like temperature within a space, etc.).

Depending on the type of interactions they support and the environment they interact with, sensors and actuators are generally divided into two categories (Armando et al., 2018).

1. *Physical:* Sensors and actuators that fall into this category are usually electronic-based (e.g., RFID readers and tags, SNS devices, etc.) and interact with the physical world to measure physical phenomenons or with human users to receive and/or generate useful information and can vary in range and complexity. These include but are not limited to, temperature sensors, microphones,

cameras, light switches, body sensors and pacemakers among others.

2. *Virtual:* Virtual, also called soft or smart, sensors and actuators are software-based. They rely either on data that is provided by physical sensors or on defined data processing models to extract the desired information. They effect changes directly on virtual environments such as software systems but require physical actuators to effect change on physical ones. Examples of virtual sensors and actuators include, safety level sensors, virtual proximity sensors (relying on physical light sensors (Madria et al., 2014)), load sensor in a virtual machine, among others.

2.3.1.2 Networking and identification

The success of the Internet is mainly attributed to its identification and networking protocols and communication models (e.g., Media Access Control (MAC), Internet Protocol (IP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), etc.). These protocols allowed the interconnection of billions of computers and mobile devices quite seamlessly creating a virtual fabric where machine across the planet can exchange data. The IoT seeks to integrate 'anything' within this fabric. Vehicles, bulbs, fridges, video cameras, fitbits and many more are examples of innovations where the integration internet into otherwise mechanical or electronic devices has proven powerful and potentially life changing. However, some of the components of the IoT suffer from resource limitations due to the nature of the environments where they are deployed (e.g., a Wireless Sensor Network (WSN) used to monitor a war zone, sensors embedded in a road to monitor pavement condition, etc.). Moreover, the sheer number of new devices that populate the IoT make IPv4 unsustainable as it makes it impossible to individually address and identify each device on the internet, even with the use of proxy techniques; not to mention the huge overhead generated by all these devices over the network (Atzori et al., 2010).

To mitigate these issues, researchers and practitioners have been moving to using more lightweight networking stacks and addressing schemes that are able to sustain the rapid growth in IoT devices connected to the Internet. The IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) (Montenegro et al., 2007) standard's protocol stack is a prominent example that is adapted specifically for the requirements of IoT applications in terms of low power consumption, low network overhead and high addressing capacity (Atzori et al., 2010). It relies on the lightweight nature of the IEEE 802.15.4 standard frames to deliver packets at the link layer (also called the network access layer) of the TCP/IP model

(Hennebert and Santos, 2014). At the network (internet) layer, it also uses IPv6 through an adaptation layer for addressing the nodes on the network, which uses 128-bits addresses represented in hexadecimal format, allowing 2^{128} possible addresses. The standard mainly relies on UDP at the transport layer to transmit asynchronous datagrams and thus avoid the overhead generated by TCP-based communications. Constrained Application Protocol (CoAP) is the last piece of the puzzle and is used at the application layer to encapsulate application payload. Similar to HTTP, it relies on a client/server architecture to deliver content between nodes predominantly using a REST architecture style. Figure 2.5 illustrates how WSN-based networks can communicate with 'classic' computer-centric network over the Internet.

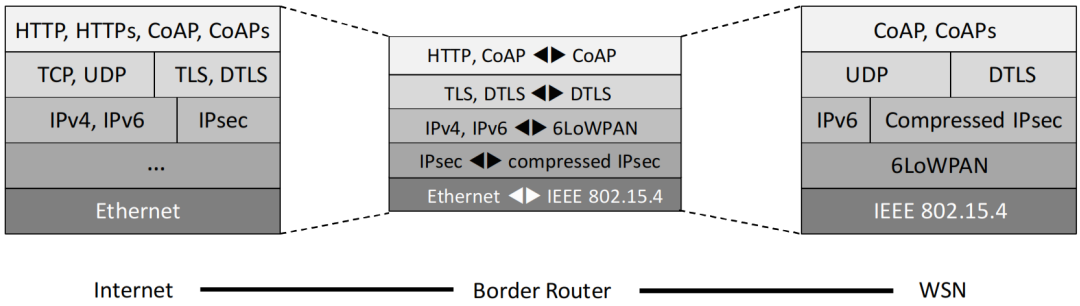


Figure 2.5 – Comparison and communication between computer-centric Internet and WSN-based Internet. From (Hennebert and Santos, 2014)

2.3.1.3 Services and applications

Undoubtedly, the IoT is a powerful tool for innovation within several application domains. Indeed, the ability to implant sensors and actuators almost anywhere, opens up the possibility to monitor the conditions surrounding business activities at the finest levels and to act based on the insights gained from the resulting data. Transportation (Levina et al., 2017), logistics (Nettsträter et al., 2010), healthcare (Laplante et al., 2017) and manufacturing (Arnold et al., 2016) are just some of the application domains that have been touched by the power of the IoT. In their paper, (Mohanty et al., 2016) discuss the role of IoT in providing services and supporting operations across multiple application domains. They divide the innovation brought by the IoT into 9 (nine) 'components', namely, Infrastructure, Buildings, Transportation, Energy, Health Care, Technology, Governance, Education and Citizens. We adopt this taxonomy to classify some examples of the services and applications in academia and industry. Table 2.3 shows examples of services and applications in each component.

Infrastructure	Buildings	Transportation
- Smart grid - Pavement monitoring	- Energy efficient buildings - Smart office	- Electronic toll collection - VANets
Energy	**Health Care**	**Technology**
- Smart metering - Energy trading	- Remote health monitoring - Fall detection	- Self-healing systems - Green IT
Governance	**Education**	**Citizens**
- Smart ID cards - Electronic voting	- Smart attendance - Intelligent campus	- Intrusion detection - Smart home

Table 2.3 – Examples of applications and services in different domains based on the IoT.

2.3.2 Cloud Computing

Due to the growing complexity of today's computing-based systems and the proliferation of mobile devices, the need for more powerful computing infrastructure and a way to deliver computing-heavy systems to otherwise constraint devices grew more and more persistent. As a potential solution, CC received great attention as a way to deliver ICT services over the Internet. In addition to the Internet, the key technology behind the genesis of CC is *virtualization* (Uhlig et al., 2005) which represents the ability to use the same physical infrastructure (i.e., hardware) to run multiple user spaces over different operating systems.

CC is defined by the National Institute of Standards and Technology (NIST)[5] as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models" (Mell and Grance, 2011). The five essential characteristics are : (1) *on-demand self-service* for the on-demand provisioning of resources with minimal interaction with the provider, (2) *Broad network access* for ubiquitous access through different types of clients (e.g. mobile phones, tablets, desktops, wearables, etc.), (3) *Resource pooling* to serve multiple consumer demands using a multi-tenant model (4) *Rapid elasticity* enables automatic scaling of the resources according to the user's need and (5) *Measured service* allows the user to pay for the computing resources in pay-per-use model. While, the four deployment models include : (1) *Private cloud*, where the cloud resources are used by a single organization, (2) *Community cloud*, in which a community of organizations with similar

[5]http://www.nist.gov/

activities share the cloud resources, (3) *Public cloud*, where the resources are open for public use and (4) *Hybrid cloud*, which is a combination of two or more of the deployment models mentioned above. Figure 2.6 gives a picture representing the actors involved in CC, its characteristics, service types and deployment models. In the following, we discuss the service types provided by cloud providers in more detail.

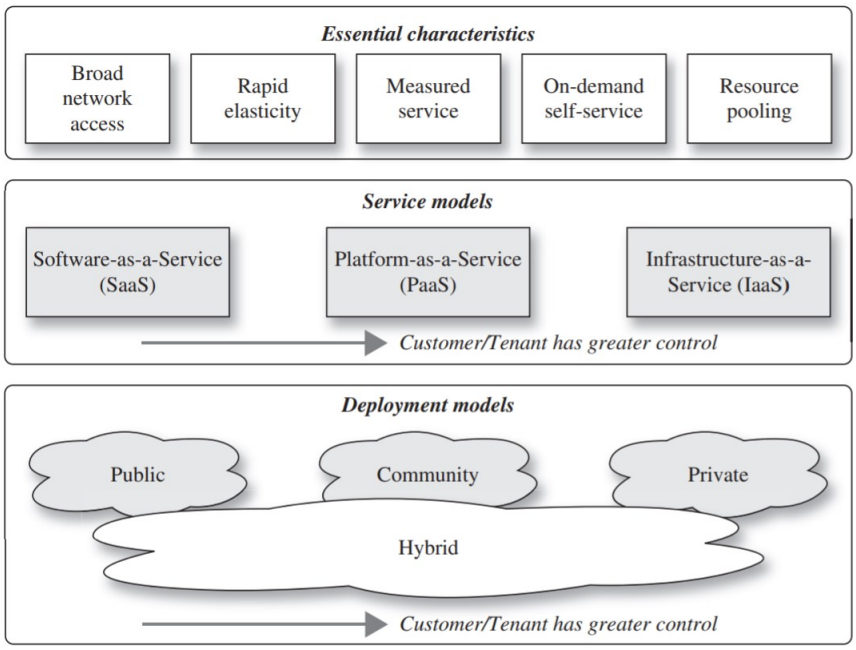


Figure 2.6 – Cloud computing, users, characteristics, service types and deployment models. From (Winkler, 2011)

2.3.2.1 Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) represents the offering of virtual hardware as services that are accessible via specific protocols like HTTP or Secure SHell (SSH) making the providers responsible for managing the physical hardware. This allows cloud users (i.e., clients) to benefit from a virtually infinite amount of computing resources, while keeping a high level of control on the platforms and software to be installed on the provisioned virtual hardware. IaaS providers usually provide mechanisms to automatically scale up if the provisioned virtual hardware is unable to process the workload or scale down if some of the provisioned resources are wasted (i.e., not used) on the workload. This scaling can be horizontal or vertical. Vertical scaling means modifying the configuration of the virtual machines in the fleet by increasing or decreasing their memory, storage, network and/or processing power. Horizontal scaling means adding or removing entire virtual machines from the fleet. Major actors in the IaaS market include

Google with Google Compute Engine [6], Microsoft with Azure [7] and Amazon with EC2 [8].

2.3.2.2 Platform as a Service (PaaS)

Platform as a Service (PaaS) cloud solutions are usually intended for application and software developers. They offer a set of frameworks and management systems to allow fast development, storage, packaging, testing and deployment of software within the virtual infrastructure. Again, it is up to the provider to manage the provided platforms they support in their offers. The development environment offered by the PaaS paradigm allows for highly distributed and available applications. However, PaaS users surrender the control over their data to the providers, which can be an issue in sensitive applications. Moreover, if there's a problem on the provider's side that affects the software running on their platform (e.g., problems with database management system or the operating system), the client cannot do anything about it. PaaS are commonly used to build and deploy web applications using a web server (e.g., apache, nginx), a programming language (e.g., python, php) and/or frameworks (e.g., Django, symphony, etc.). Major actors in the PaaS market include Google App Engine [9], IBM's Cloud Foundry [10] and Amazon's S3 [11].

2.3.2.3 Software as a Service (SaaS)

The Software as a Service (SaaS) service type is the most common type of cloud services around. It is basically any cloud-based application that the client can only use via the Internet using either a web browser or a mobile app or an API. The providers are responsible for managing the whole technological stack from the physical and virtual hardware to the application deployment and execution. To the clients of these services, this brings the advantages of easy usage, deployment and configuration, especially for services that are considered as a commodity such as email. However, they have no control over the data inputted to the software service nor can they intervene if some problem arises on the provider's side. Popular examples of the SaaS paradigm include Google's G-Suite(e.g., Gmail, Drive, Sheets, Docs,etc.), SalesForce, Dropbox and Microsoft's Office 365.

[6] https://cloud.google.com/compute. Last accessed on March 24, 2021
[7] https://azure.microsoft.com/. Last accessed on March 24, 2021
[8] https://aws.amazon.com/ec2/. Last accessed on March 24, 2021
[9] https://cloud.google.com/appengine. Last accessed on March 24, 2021
[10] https://www.ibm.com/cloud/cloud-foundry. Last accessed on March 24, 2021
[11] https://aws.amazon.com/s3/. Last accessed on March 24, 2021

Figure 2.7 illustrates how each service type is positioned with regard to the different components of an IT infrastructure when compared to an on-premise solution. The figure summarizes the key differences between the different service types according to nine components, namely, networking, storage, servers, virtualization, operating system, middleware, runtime, data and applications.

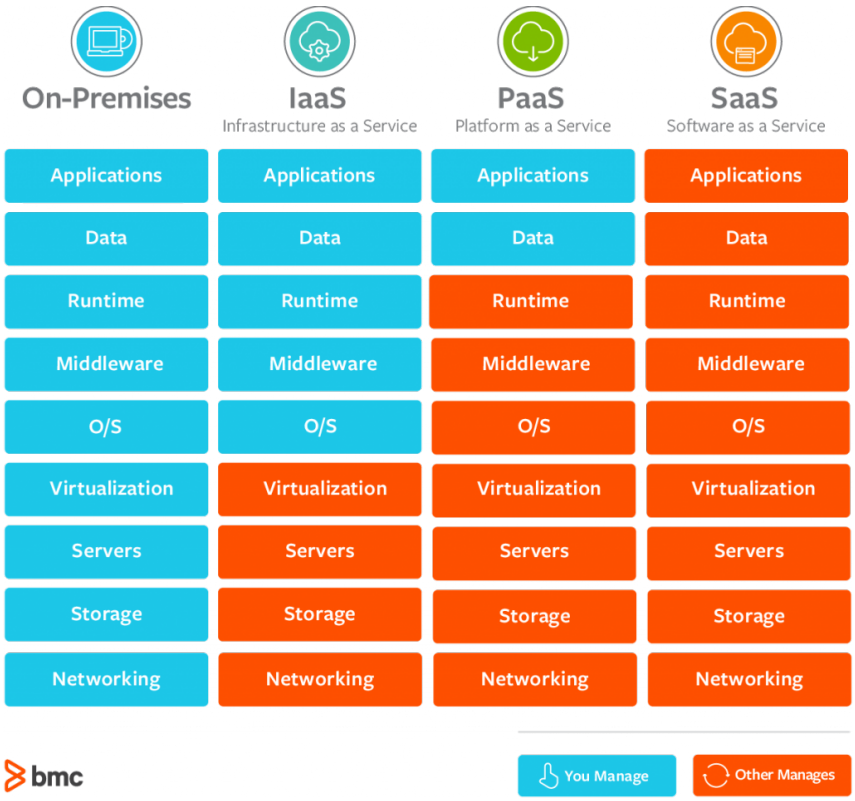


Figure 2.7 – Summary of key differences between the different cloud service types and on-premises solutions [12]

2.3.3 Big Data

Although the debate regarding the definition of Big Data and its important aspects knows much controversy, there is no doubt that we face several data-related issues in today's digital landscape. These issues have either been there since the dawn of computing or have been brought about by the new advances in communications and electronics. Nowadays, Big Data is an umbrella word that covers any and all challenges and practices that deal with data from their capture and storage to their processing and analysis. The word "Big" in Big Data suggests a volume dimension, meaning that the amount of data is too big to manage efficiently, but it also suggests the complexity related to the behavior and

[12] https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/. Last accessed on March 25, 2021

structure of the data. Initially, META's (now Gartner) definition of big data [13] stated below, defined three dimensions of Big Data, namely, Volume (i.e., the size and magnitude of the dataset), Velocity (i.e., the rate at which data is generated and/or has to be processed) and Variety (i.e., the different types of data structures in a dataset).

> *"Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation."*

However, as the subject gained traction, more and more V's were thrown in the mix as we went from 3 V's to 42 V's (Shafer, 2017). Putting aside the obsessiveness of using words beginning with 'V' to discuss Big Data dimensions, there is actually no quantifiable manner to distinguish what Big Data is (Ward and Barker, 2013). Consequently, in the following, we discuss Big Data as the sum of the different phases (i.e., steps) that allow the transformation from raw data to knowledge of value, while presenting some of the prominent techniques, technologies and issues faced at each phase.

2.3.3.1 Collection

Data collection is the basis of every data-based activity. Without data, scientists cannot confirm and/or evaluate their contributions and managers cannot adapt or monitor their business goals and strategies. This constitutes a major challenge in several research areas and business domains. For instance, machine learning applications usually require big datasets that are clean, uniform and accurately labeled or tagged. There are a lot of techniques that are used to generate such data. The implementations of these techniques are specific to the type of the collected data (e.g., textual descriptions, ECG signals, images or videos). However, acquired data is rarely complete as data gets corrupted and/or missing due to faulty hardware or mishandling of data at any stage of the acquisition process.

As we perceive it, Big Data collection deals with challenges at two levels. First, at the source level, to reduce the amount of data collected that have no particular value for the targeted application. This is usually done through filtering. The goal here is to check, at the source, if the data conform to a set of rules for them to be collected. This is especially useful in applications where the source of data are sensors (physical or virtual) generating data at an important rate. For instance, if a camera feed is blocked by some object, there is no reason to transmit the data

[13] https://www.gartner.com/en/information-technology/glossary/big-data. Last accessed on March 25, 2021

over the network as data have no particular intrinsic value. To do this, some logic needs to be implemented either at the same device operating the camera or on the same local network. Second, in-memory level, to preprocess the collected data in order to prepare it for a specific use and facilitate their exploration. At this level, several techniques are used, such as missing values imputation (Ryu et al., 2020), noise treatment (Zhong et al., 2014), discretization, normalization, feature selection (Y. Zhang and Cheung, 2014) and aggregation (Obinikpo and Kantarci, 2019). Novel frameworks such as Hadoop MapReduce[14] and Spark[15] offer efficient built-in modules to perform these tasks, either for stream or batch processing.

2.3.3.2 Management

Data management describes how big data is encoded, stored and retrieved. The rate at which data are generated nowadays and their heterogeneity requires DataBase Management System (DBMS)s to be scalable in size, efficient in read-/write operations and highly available to client applications. These issues have sparked new innovations on big data management in both academia and industry. In this sense, the most significant advances have been made in what now falls under "NoSQL" DBMSs. The name goes to show that these DBMS are not based on Structured Query Language (SQL) or any of its extensions. However, the differences between SQL and Not Only SQL (NoSQL) DBMSs are not limited to the querying language they use. Indeed, NoSQL databases also have flexible data models, are distributed and easier to scale, and have no investment to design models as conceptual models are highly coupled to the application logic (Gudivada et al., 2014).

NoSQL DBMSs are usually classified based on their data representation schemes under one or more of the following four types. (1) Key-value, use keys to index the corresponding value for each 'column' of a data record (e.g., Redis[16], aerospike[17]). (2) column-oriented, use columns to store data separately as opposite to classic row-oriented relational DBMSs (e.g., Cassandra [18]). (3) Document-oriented, are similar to Key-value DBMSs but support more complex queries and hierarchical relationships (e.g., MongoDB[19]). (4) Graph-oriented, use graph-like structures to store data along with relationships between data records (e.g., Neo4J [20]). Each

[14] http://hadoop.apache.org/. Last accessed on March 25, 2021
[15] https://spark.apache.org/. Last accessed on March 25, 2021
[16] https://redis.io/documentation. Last accessed on March 25, 2021
[17] https://www.aerospike.com/docs/. Last accessed on March 25, 2021
[18] https://cassandra.apache.org/doc/latest/. Last accessed on March 25, 2021
[19] https://docs.mongodb.com/. Last accessed on March 25, 2021
[20] https://neo4j.com/docs/. Last accessed on March 25, 2021

DBMS uses a somewhat unique query tool to query and retrieve the data of interest. These tools can be put forward as APIs, frameworks (e.g., MapReduce) or languages (e.g., AQL for Aerospike and Cypher for Neo4J). Techniques such as clustering and/or sampling, to enable storing somewhat large volumes of data in relatively small and limited storage resources; replication, to allow high availability and low latency in responding to client requests, and indexing, to improve the performance of the DBMS in executing queries on the database, are used at this level to mitigate some issues related to the use of NoSQL databases.

2.3.3.3 Analytics

Analytics describes the set of methods and techniques that are used to derive knowledge from collected and stored data over a period of time. However, the characteristics of Big Data discussed above make this task challenging as efforts in data analysis have been mostly directed toward single machine algorithms with complete data. As a result, recent years have known many efforts to develop distributable algorithms that are able to perform analytic tasks over data either in stream and batch forms. These algorithms can be divided into three major classes based on their analytic goal.

- *Descriptive:* Descriptive analytics have the common goal of trying to answer the question of 'what happened?' in the past. They generally rely on the statistical analysis of past data to infer some insights into events or trends with these data. To help interpret these trends and events, the results of the processing algorithms are generally presented as visual artifacts like charts, graphs and dashboards. A popular example of descriptive analytics is using frequent pattern mining algorithms like FP-tree (Han et al., 2000b) to extract the items that are usually bought together in a supermarket. Another example is analyzing the number of accidents on different roads to get a picture of the dangerous roads on the infrastructure.

- *Predictive:* Predictive analytics try to answer the question of 'what is most likely to happen?' in the future. The algorithms that fall in this class rely on the analysis of past data to infer some "laws" or functions that govern how the data changes over time. These functions are then used to guide the decision-making process of managers to improve some aspects of their business. For example, machine learning based predictive analytics can be used to predict the number of accidents that can happen in the near future on every road (Narasimhan et al., 2017) to set up more road signs or maybe police patrols.

- *Prescriptive:* Prescriptive analytics goes one step further. Algorithms in this class try to answer the questions of 'what to do when it happens?'. They rely on past data and simulation and optimization techniques to understand how each parameter plays into the function and the way the outcome changes when these parameters change with the goal of maximizing profit and/or minimizing risk and loss. For example, optimization techniques can be used to analyze traffic data and redirect some traffic to have optimal traffic flows over a large road area (Jin et al., 2017).

2.4 On the improvement of Smart Systems

2.4.1 Approaches for the improvement of architectural aspects in Smart Systems

In the development process of software systems, architecture is an important artifact that guides the technical choices while offering a high abstraction over technical details. A Software architecture illustrates, usually in a visual way, the structure of the system through the definition of its artifacts (entities, components, policies and processes) as well the relationships between these artifacts and their relationship to their external environment. Although the architectural choices made by software systems engineers tend to ease development and define responsibilities, it also makes them more rigid later on due to the top-down nature of the process. This has prompted a heated discussion on the importance of software architecture between 'purists' who stress the importance of good quality software and agile methodology followers who defend the importance of having working software in the shortest time possible. This debate led to the rise of research on adaptation and evolution in software architecture in effort to make software systems more adaptable. In the following, architectural adaptation and improvement is discussed using two popular architectural styles, namely, component-based architecture and service-oriented architectures.

2.4.1.1 Component-based adaptation and improvement

It has been more than fifty years since the idea of developing software systems through components was formulated (McIlroy et al., 1968). The idea promotes reuse through the development of connectable software components rather than monolithic software. Early on, researchers in component-based design often assumed that components are reused "as-is". This was later refuted by practitioners as they found that reusing software component with no adaptation is very rare.

Indeed, software components needed to be modified in order to be reused to interact with different software components in different software systems. To cater to this issue, copy-paste, inheritance and wrapping were commonly used but presented their own shortcomings (e.g., efficiency, implementation, etc.). However, the type of adaptations envisioned stayed mainly at the code level and was hard to maintain in the long term. Since then, several approaches have been proposed to deal with the adaptation of component-based software system to achieve long term evolution.

Superimposition is a concept that was proposed by (Bosch, 1999) to tackle the problems mentioned above. *Superimposition* is presented as an adaptation technique allowing one to impose predefined, but configurable types of functionality, called *adaptation types*, on a reusable component. This is possible through the decoupling of the component and the adaptation functionality so they can be treated separately. However, their integration and composition is crucial to achieve complex adaptations on component-based software systems. The author also identifies three types of adaptation levels to which adequate adaptation types are proposed. First, adaptations related to the interface of the component to address situations where a component could potentially be reused in a new system whereas its interface does not match the interface expected by the target system. Second, adaptations related to component composition to address the problems where no one component can satisfy a system's desired functionality. Third, adaptation related to component monitoring to enable the detection of unwanted behavior at the component level and the triggering of remediation strategies at the system level.

To tackle the issues of heterogeneity in multimedia-intensive software systems, the authors in (Derdour et al., 2010) propose a model-based approach called MMSA (Meta-model Multimedia Software Architecture). The main idea behind MMSA is the use of connectors as adaptation entities to allow the component transformations that are necessary to cater to the different types of components used in multimedia systems. At its core, MMSA mainly relies on two models, namely, the data flow model and an architecture model. The dataflow model, depicted in figure 2.8a, allows the description of the various types of media types that are used by multimedia application as well as their relationships. The architecture model, presented in figure 2.8b, depicts the elements of multimedia systems as a combination (i.e., configuration) of software components and connectors. Describing multimedia systems at high levels of abstraction allows for early separation between functional aspects and non-functional aspects, thus facilitating the adaptation of the system's functional behavior separately from

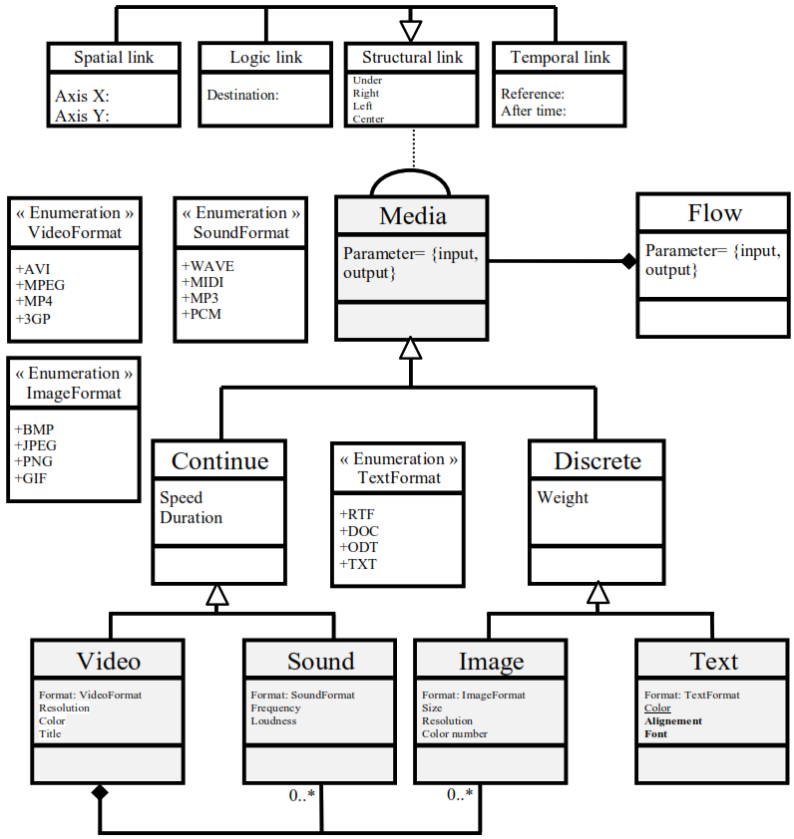
non-functional constraints.

Learning techniques have also had some applications in the context of adaptation in component-based software systems. A representative work is presented in (Esfahani et al., 2016), where the authors applied mining techniques to automatically and dynamically create the software's *component interaction model* describing what components interact with each other and represented in a set of rules called *interaction rules*. The goal is to mitigate some of the drawbacks associated with the *a priori* knowledge about the behavior of the targeted software, which is required to design adaptation strategies. The inferred *interaction rules* are then used to power up several applications that assure the self-management properties of the targeted software system. The authors provide three applications that potentially benefit from the inferred interaction rules. First, the authors present the details of how the interaction rules are used to safely applying dynamic changes to a running software system without creating inconsistencies. Second, they discuss how these interaction rules can be applied to identify potentially malicious (abnormal) behavior for self-protection. Third, they use the interaction rules to improve the deployment of software components in a distributed setting for performance self-optimization.

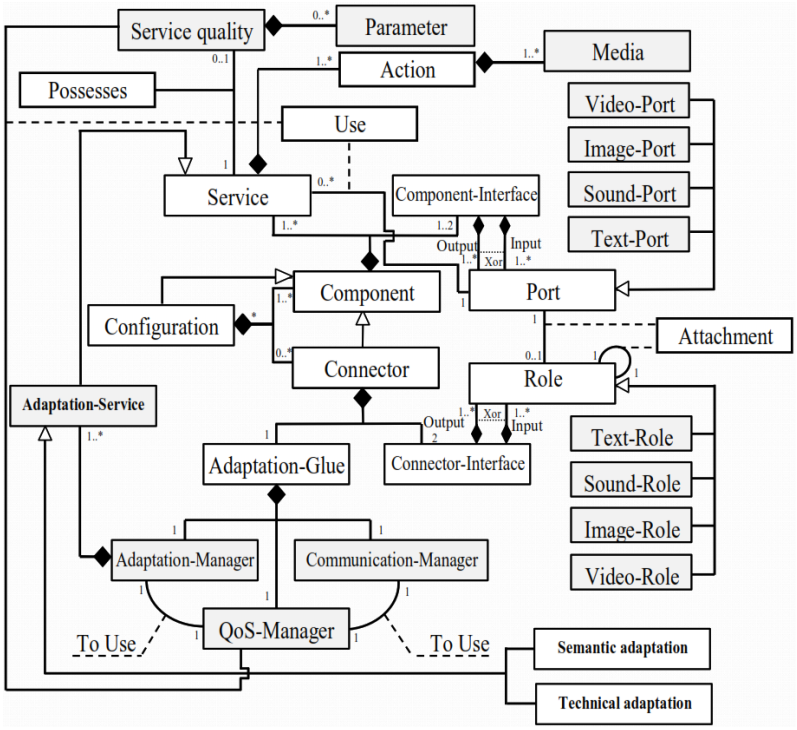
2.4.1.2 Service-based adaptation and improvement

Compared to component-based software, Service-based software is much more recent paradigm as we discussed in section 2.2.3. It seeks to separate software units by aligning them with business goals rather than functionalities. However, it still encountered some of the same issues faced by component-based ones, especially related to rigidity and need for adaptation. To cater to these issues, some approaches from component-based software engineering were adopted and adapted to deal with adaptation, while others proposed techniques to deal with specific issue related to Service-based System (SBS) (e.g., dynamic service binding, dynamic service composition, service replacement, etc) (Kazhamiakin et al., 2010). Other approaches rely on novel techniques and paradigms such as those belonging to Machine Learning (ML) research to cope with the advances in the technological landscape and leverage the deluge of data generated in the span of the software execution life cycle. In the following, we present and discuss representative works of the development on adaptation in service-based software systems.

To enable SBSs to satisfy non-functional constraints (e.g., cost, reliability, etc.) in open and dynamic service environments, the authors in (Cardellini et al., 2011) propose a tool supported methodology called MOdel-based SElf-adaptation of



(a) Multimedia flow model for MMSA.



(b) Class diagram of software architecture MMSA.

Figure 2.8 – Core models in the MMSA approach (Derdour et al., 2010).

SOA systems (MOSES). The authors rely on a detailed taxonomy of the problem space on the adaptation of SBS, with a special focus on Quality of Service (QoS) constraints, to position their approach at the intersection of both adaptation (or self-adaptation) and SBS. MOSES uses IBM's control loop MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) as an overarching design to the approach. The *knowledge* component holds the data describing the services and the characteristics of their operating environments (i.e., QoS parameters). The *monitor* and *analyze* components serve as a perception layer to detect changes in the services or their operating environment in order to trigger the action of adaptation. Once a request for adaptation is issued, the *plan* component tries to compute the best adaptation plan either in the behavior of the composite services (e.g., replacement of concrete services, changing composition workflow, etc.) or in the selection of the best candidate concrete services to construct the optimal composition for each composition request. The *execute* component is then responsible for carrying out the changes required by the adaptation at runtime while ensuring the system's consistency and then deploying and running the resulting compositions while ensuring the link with the *monitor* component is valid.

Feature models, a common technique used in Software Product Line (SPL) engineering (K. Lee et al., 2002), have also been used to deal with the challenges of adaptation in SBSs. In (Cubo, Gamez, et al., 2013), the authors rely on feature models to address the problem of adaptation of services in the DAMASCo framework (Cubo and Pimentel, 2011). The approach considers every valid configuration of features (i.e., a configuration that satisfies the tree and cross-tree constraints) to be a potential service but the deployed services are only a subset of the possible service space. Hence, a feature model is associated to each service with points of variability known at design time. That way, adding a new feature to satisfy the user's requested service becomes an easy task. To integrate this adaptation into the DAMASCo framework, the authors perform a mapping between the feature models and the artifacts produced by the transition system that is part of the DAMASCo framework called Context-Aware Symbolic Transition Systems (CA-STS). The authors discuss the promise of the approach on a Proof-of-Concept in the transport domain which demonstrates the usefulness of the adaptation mechanism. The resulting architecture of the feature model extended DAMASCo framework is presented in figure 2.9.

A common issue in adaptation research in general and SBS in particular is that adaptation are only triggered in reactive manner. Recently however, the advances in ML have made it possible to make fairly accurate predictions in the near future from the execution data, prompting their use to make adaptation proactive. An

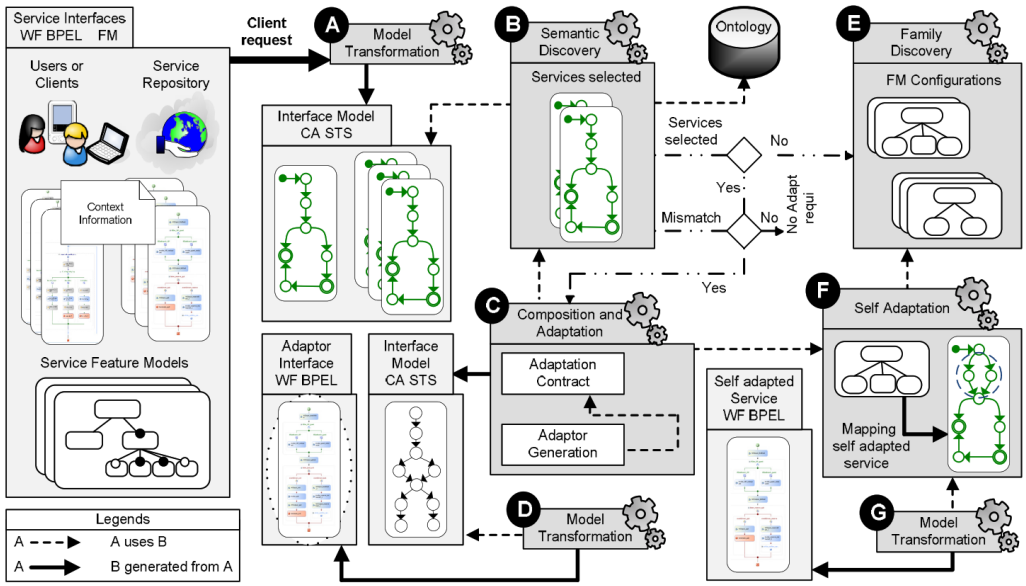


Figure 2.9 – Feature model extended DAMASCo framework architecture. From (Cubo, Gamez, et al., 2013)

exemplar work is presented in (H. Wang et al., 2018), where the authors propose a combination of an efficient prediction model to predict the future reliability as a QoS parameter of SBSs, and a decision-making approach to select of adequate adaptation strategy to be adopted. Specifically, the authors leverage Dynamic Bayesian Networks (DBN)s to make predictions on the reliability of each service in the SBS. These predictions are time-based, where the time parameter can be configured based on the sensitiveness of the application (i.e., shorter time requirement for sensitive applications). Moreover, each prediction made for $t+1$ can be leveraged to make a new prediction for $t+2$ creating a multi-step prediction which the authors called *multi_DBN*. For the adaptation strategies, once the adaptation is triggered by the online prediction module, the paper proposes the replacement of the 'faulty' or unreliable service. To proceed, the services that cannot satisfy the QoS requirements of the user are dropped first. Then, the remaining candidate services for the replacement are ranked based on their predicted reliability. Lastly, following the ranking order, the services go through a usability test using a Recovery Block mechanism to insure the usability of the service in the targeted SBS.

2.4.2 Approaches for the improvement of technological aspects in Smart Systems

Adaptation has been an important subject in the field of software engineering since the researchers noticed that software systems keep getting more complex

and their execution environments ever more heterogeneous and dynamic. In this regard, new technological advances are no exception. As a matter of fact, adaptation can have several applications within the same technology or paradigm. In this section, we present some adaptation applications with the identified technological enablers related to SSs, namely, CC, IoT and Big Data. It is by no means an exhaustive list of the possible adaptation applications nor that of possible approaches to adaptation. However, the goal of this section is to establish a common ground where the integration of different technologies in the engineering of SSs becomes not only possible but of an added value.

2.4.2.1 Internet of Things adaptation and improvement

As mentioned early on, in section 2.3.1, the IoT paradigm comprises of heterogeneous technologies that integrate together to provide users with highly personalized services in different domains. Hence, one way to present the possible adaptation applications in IoT is to follow the same categorization that we followed in section 2.3.1 by separating the three silos of hardware, communication and software.

Sensing and actuation In literature, most of the adaptation techniques used to improve sensing and actuation seek the objective of energy-efficiency. This is because most of the devices in IoT tend to be mobile and hence lack a constant source of power and their batteries remain viable for a limited time period. In this sense, adaptation techniques are used to alter the defined behavior of IoT devices to optimize their resource (e.g., energy) consumption. Adaptations can be implemented either at the hardware level (e.g., using programmable hardware like Field Programmable Gate Array (FPGA)) or at the software level (e.g., offloading and load-balancing). In the following, we focus our attention to the latter set of solutions.

Since IoT devices mainly use energy to receive and send wireless signals, most of the literature is focused on how to minimize the execution of costly operations. One technique where adaptation has proven useful is *sleep scheduling*. For example, the authors in (Q. Li et al., 2017) propose EnergIoT, a technique where they leverage the strength of clustered structures to design hierarchical network models where battery preservation is a priority. In their approach, the authors give devices adaptive time periods where they switch between sleep and active states according to their distance from the sink. Moreover, they also take into account the energy consumed during the network construction phase and the

data processing phase. To evaluate their approach, they demonstrate its efficiency at maximizing the lifetime of the network via simulation-based experiments.

Networking and identification Improving networking and identification capabilities is not a problem that is exclusive to the IoT. However, due to the heterogeneous, dynamic and mobile nature of the devices in IoT, the issue is more pronounced and pressing. Adaptation has played a big role to make physical network topology more efficient, especially in the applications of Mobile Cloud Computing (MCC), where most of the network devices are mobile (S. Wang and Dey, 2013). For the IoT however, adaptation techniques have mostly been applied to networking problems in the context of WSNs. In this context, in a software-like manner, (Oteafy and Hassanein, 2016) propose *adaptive components* as an approach to build resilient IoT architectures over dynamic WSNs. The rationale is to decouple the component's functionality from the physical device by abstracting IoT things as dynamic interfacing components. The authors argue that this allows for building architectures that are more resilient, resource-efficient and dynamic. The dynamic aspect comes from the adaptive association between functional components to build the network.

Since IoT is expected to weave more than 50 billions connected devices within its fabric, a challenging task concerns the deployment of such a big number of devices in a seemingly *ad hoc* network to allow smooth communication. To that end, network protocols need to implement adaptive techniques to allow devices to emit signals at varying distances and at varying speeds, taking into account resource constraints and deployment schema. LoRa (Slabicki et al., 2018), has a built-in mechanism called Adaptive Data Rate (ADR) that serves to dynamically manage link parameters for scalable and efficient network operations. However, this mainly designed at stationary networks, which doesn't cover several applications of the IoT. To cater to this limitation, the authors of (Benkahla et al., 2019) propose to enhance the ADR mechanism by making it adaptive to the node's mobility through quick reconfiguration of the mode (combination of link parameters). Their conducted experiments show that E-ADR (Enhanced-ADR) improves the quality of service (QoS) of the overall network.

Services and applications Adapting and improving services and applications in the IoT context depend on the paradigms that are adopted to develop these services and applications and hence follow the same patterns that we previously mentioned in section 2.4.1. We focus on service-based IoT applications in this section and evoke some adaptation applications in that sense. For instance, service

discovery is an important step in building service-based applications. At different scopes, it allows engineers or end users to discover services that are of interest to them at a given moment. However, as services grow in number and complexity in the IoT context, it becomes harder and harder to organize them efficiently in the services registries. To address this problem, (Cabrera and Clarke, 2019) propose a self-adaptive service model for smart cities (i.e., an IoT application) to support service discovery. To make it possible, the authors propose a smart city knowledge model in the form of an ontology serving as a way to organize the events withing the city and to organize and classify the available services. The self-adaptive service model is capable of organizing the services information according to both scheduled and unforeseen city events. A self-adaptive architecture is then proposed to keep track of the discovery metrics and moves information about services between registries within the city to improve the efficiency of service discovery.

2.4.2.2 Cloud Computing adaptation and improvement

As one of the most influential technological advances of the 3rd millennia, CC has also known several improvements that are directly linked to its capability to adapt to the user's requirements. As a matter of fact, in its inception, one of the most attractive characteristics of CC is the adaptive payment model, which is implemented in the pay-as-you-go model. Since then, several technical aspects have been made adaptable to improve the performance of clouds, either for specific cases (e.g., natural disasters) or to minimize downtime and resource allocation. In the following, we present some of the works that leveraged adaptation techniques to improve cloud environments.

On the Infrastructure as a Service level Undoubtedly the prime focus of researchers in the improvement of CC environments, IaaS, has seen many proposals that seek to enhance different aspects (e.g., resource provisioning, load-balancing, energy consumption, etc.). This is not surprising as 'Infrastructure' is considered the most important part of CC, on which everything else is built. The range and scope of its operations also make it challenging to develop effective adaptation strategies since it needs to balance between cost, performance, resources and energy. A seminal work in this context is proposed by (Jung et al., 2010) called Mistral. To make the aforementioned balance possible and effective, Mistral uses optimization techniques to make adaptive decision making on resource allocation and application migration in order to save on cost and energy with minimum loss in performance. Mistral also takes into account the time and cost that the

controller (i.e., the component responsible for making adaptation plans and decisions) takes to favor fast but temporarily more performance-costing operations instead of taking longer to take optimal decisions.

One drawback of the classic CC paradigm, is the assumption that resources are always available and stationary (e.g., in big data centers), which is not always the case. The Mobile Cloud Computing paradigm was born to address this problem and at the same time alleviate some of the resource-related problems of mobile computing. The objective here is to allow resource-constrained devices (e.g., smartphones) to offload some of the costly operations -from a resource perspective- more appropriate (i.e., powerful) 'devices' (i.e., server, or a data center). From an operational standpoint, the biggest issue is that of the offloading method. In this regard, adaptation has proven to be an efficient approach. For instance, (Nakahara and Beder, 2018) propose CoSMOS (Context-Sensitive Model for Offloading System), a context-aware and self-adaptive offloading decision support model for mobile cloud computing systems. CoSMOS follows a self-adaptive and self-expressive architecture to balance both execution time and energy consumption at the method level to decide what components need to be offloaded to the central cloud.

On the Platform as a Service level Since cloud platforms usually target specific development environments, the possibilities for adaptation are very limited in scope. Hence, we notice that most adaptation possibilities, and thus works, lie in the interplay between IaaS and PaaS. This is because PaaS are built upon IaaS and thus most of issues of IaaS are inherited by PaaS. To add more dimensions to the energy-consumption problem, (Djemame et al., 2017) argue that energy gains can also be obtained through the optimization of IaaS and PaaS operations. To that end, the authors propose a detailed self-adaptive architecture covering each layer of the cloud software stack and that is argued to facilitate energy-aware and efficient cloud operations. Adaptation rules are used to implement adaptation strategies based on the monitoring of Service-Level Agreement (SLA) agreements (i.e., through KPIs) at both the IaaS and the PaaS levels. The detection of breaches at the SLA level or adaptations at one layer (i.e., PaaS or IaaS) triggers the adaptation on the other level to further minimize energy consumption while keeping the cost and performance of the operating services within the SLA.

On the Software as a Service level Software sits on the top of the CC stack. Performing any adaptation on this level influences all of its underlying layers (i.e.,

platform, middleware and infrastructure), while only being influenced by adaptations on its level (Marquezan et al., 2014). Since, software is the direct link between the end user and the cloud, most of the adaptations triggered at this level serves to accommodate the needs and/or preferences of the user by changing either the application business logic or the service selection process. This is especially true for SEs where the user's state and his context change quickly and unpredictably. (Tzafilkou et al., 2017) discussed the particular issue for the need for continuous user-centered cloud service adaptation in SSs. To bridge the gap between software engineers and the potential users of the SS at design time, the authors propose a framework that enables self-adaptation of cloud services based on users' distinct needs and requirements. They proceed by analyzing user implicit and explicit feedback to construct user profiles that encompass their behavioral patterns and preferences. These profiles are later exploited to implement adaptation rules to create application stores that are personalized to the user's context.

2.4.2.3 Big Data adaptation and improvement

Big Data, as a computing paradigm, has also had its fair share of adaptation techniques applied to its various capabilities. Although, Big Data mostly deals with data related issues, the techniques and algorithms involved usually have to deal with application- and infrastructure-specific requirements. Overall, adaptation was mostly used in this sense, to make big data techniques more appropriate to the context of its use. To discuss how adaptation was used to improve Big Data techniques, we stick to the same categorization followed back in section 2.3.3.

Collection Data collection in the context of Big Data applications can be challenging depending on the sources of the data. In SEs, these data usually come from different sources (i.e., IoT sources). Hence, it is important to take into account the characteristics of data production (e.g., heterogeneity, rate, size, etc.) in the techniques used for data collection in big data. This requires an adaptation step to move from classic techniques used mainly to exploit data in more or less stable computing environments. Authors in (Yan et al., 2019) focus on Big Location Data and discuss the shortcomings of classical policies. They argue that the release of collected data at fixed intervals of time does not bode well with the big data analytics at real time. To tackle this issue, part of their paper focuses on deploying an adaptive sampling strategy to achieve a dynamic release of big location data. Their approach, which is based on a control loop implementing the proportional-integral-derivative (PID) controller, allows to track the dynamic

trends within the data over time. The authors argue that their approach remains extendable to other big data applications that deal with the issue of frequent data release.

Management Managing big data implies dealing with a lot of the issues related to their effective storage in a way that makes exploiting them faster and cleaner. Techniques like caching have been quite useful in this endeavor and have been used for almost all types of systems (e.g., DNS servers, Web servers, search engines, etc.). However, with the growing trend of collaboration on Big Data projects and research, it also has to deal with a varying data demand and external data access. To that end several adaptation strategies were adopted to make caching more effective in dealing with the characteristics of different applications. Among these efforts, (R. Gu et al., 2019) have proposed a framework to enable adaptive cache policy scheduling with the goal of improving I/O operations in Big Data applications. More specifically, they leverage data from hit ratio statistics to develop a prediction model that can infer the behavioral pattern related to data access. The prediction model is then used to select the best caching policies in different application scenarios. The experiments conducted in the paper show the relevance and effectiveness of using adaptation techniques to improve the management of big data.

Analytics Due to the characteristics of Big Data in term of size, heterogeneity, rate and incompleteness among others, performing effective and accurate analytics on Big Data is a challenging task. Furthermore, this type of analytics are usually performed by users having different roles and different technical expertise from technical experts to system engineers to business analysts to final users. This complex landscape prompted researchers and practitioners to propose different levels of abstraction to deal with Big Data problems. To facilitate the use of big data pipelines and encourage their reuse, (Kantere and Filatov, 2015) propose a workflow model that enables adaptive analytics on Big Data. They do so by separating the task's functionality from its dependencies and decoupling the application logic from its implementation in a similar way to service orientation. In this context, they discuss adaptation at two levels. First, at the user level, the paper argues big data analytics should be adaptive to the user's expertise, role and interest. Furthermore, since analytics are application-specific, they should be adaptive to the context of the application. Second, at the system level, the paper mentions multi-tenancy and online adaptations as techniques that need to be implemented to optimize the execution of analytics.

2.5 Summary

The goal of this chapter was to introduce the scientific and technological context in which our thesis work takes place. We defined the concepts and background that we deemed necessary for the reader to understand this thesis. Specifically, we focused on three different dimensions that are of interest when building SSs, namely, design, technology and improvement.

We started by investigating how SSs were defined in the literature and noticed how the definitions differ as the backgrounds and interests of the authors change. Afterwards, we tried to provide a clear distinction between SSs and **Intelligent Systems** and noticed that SSs are more holistic and user-centered than **Intelligent Systems**. We also introduced and investigated the features of CASs as we drew some interesting parallels to SSs.

In this thesis, we are interested in the design of SSs. Hence, we investigated some of the prominent paradigms in the design and development of software systems, especially those that have motivated our contributions later. We gave an overview of software engineering paradigms, namely, user-centered, context-aware and service-oriented software design. The different concepts that we introduced in these paradigm, as well as the definitions and analogies introduced earlier, made us aware of some possible relationship that will be useful in the design of SSs. These relationships will be explored more in depth in the followed chapter in a new methodological framework depicting the design principles and capabilities of SSs.

Technological aspects are also of interest in this thesis as the genesis of SEs is largely attributed to technological advances. We focused on three major technologies, namely, CC, IoT and Big Data and introduced the different components of each technology. This has led us to notice some interesting synergies between the different technologies as well as with the engineering paradigms that we introduced earlier in this chapter. These synergies will be explored later on in the context of a methodological framework that will be presented in the next chapter.

One of the main characteristics of SSs is their ability to improve and adapt to their users and environments. Hence, we introduce some of the adaptation and improvement techniques that were introduced in literature to allow software and cyber physical systems to adapt and improve. We focus and divide these techniques on both the design and technological aspects.

In the next chapter, we will introduce the previously mentioned methodological frameworks that we developed based on the observations and conclusions

from this chapter. These methodological frameworks will serve a twofold purpose. First, they will serve as a comparison framework to compare selected related works approaches that have dealt with building SSs in the next chapter. Second, in the second part of this thesis, they will serve as the basis for the development of our method to design SSs.

Chapter 3

A multi-dimensional analysis of related smart systems design, development and improvement

Science is supposedly the method by which we stand on the shoulders of those who came before us. In computer science, we all are standing on each others' feet

Gerald J. Popek

This chapter presents an analysis and a discussion of some of the approaches related to designing and developing Smart System (SS)s. Our analysis is based on the three aspects presented in the previous chapter, namely, the concepts adopted in the design of the SSs, the enabling technologies that are leveraged towards their development and the improvement techniques and strategies that are adopted towards their adaptation. To that end, we first present two methodological frameworks that we developed to describe the capabilities and the technological landscape of SSs, respectively called PeRMI and C2IoT. We then summarize and discuss the selected related works according to three mentioned aspects, before positioning the overarching objectives of our own research effort within this same global framework.

3.1 The PeRMI framework

In this section, we present our *PeRMI framework* as a tool allowing us to paint a holistic view of SSs and where we can position the different solutions and research

works on the design, conception and development of SSs. Inspired by the analogy between SSs and Complex Adaptive System (CAS)s, this framework structures the different facets of SSs into three (3) different major capabilities. The first capability is *Perception* (**Pe**), it translates the ability of the system to acquire a comprehensive view and model of its own state (i.e., the states of the parts constituting the system) and of its surrounding environment. The second capability is *Response* (**R**), it represents the ability of the system to act effectively and efficiently upon the system's internal and external boundaries when the need arises in an autonomous way. The third capability is *Manual Intervention* (**MI**), it typifies the ability of the system to accommodate and integrate the preferences of its users in the autonomous *Perception* and *Response* cycles. Figure 3.1 illustrates how the three capabilities of PeRMI framework relate to each other. The *Perception* and the *Response* capabilities are continuous in nature. Combined, they allow the SS to autonomously deal with the different user needs and situations that arise in the system's internal and external boundaries. However the user can keep control of the SS's autonomous Perceive-Respond cycles through *Manual Intervention*.

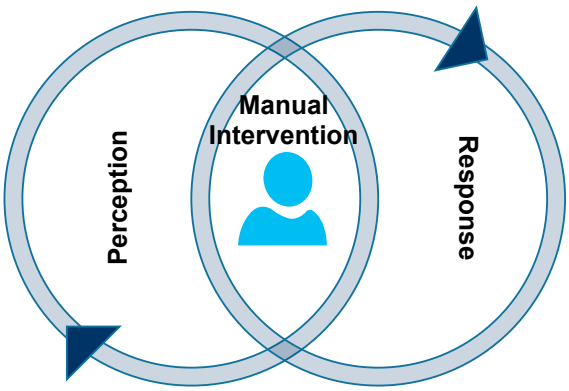


Figure 3.1 – The relations between the capabilities of the PeRMI framework. The Perception process iteratively triggers responses that in turn change the perception of the world. Manual intervention can either be in the form of a response or a perception which produces a new cycle of influence.

The rationale behind the instigation and conception of this framework is based on our analysis of SSs early on in section 2.1, especially the definitions in table 2.1. Indeed, most of the definitions evoke the capabilities of the framework in one sense or another. For instance, the concept of *Perception* is eluded to through various notions like sensing and acquiring, while the concept of *Response* is put forth through notions like applying, influencing or reacting. *Manual Intervention* was put forth to regroup the concerns regarding the users of SSs like User eXperience (UX), interaction or user behavior. This framework is also motivated and

inspired by the works of (Haeckel, 1999; Nechkoska, 2020) on providing managerial and informational adaptability in tactical management through the S/R (Sense-Respond) framework as they also consider the organisation to be a case of CAS. However, while their framework is mostly meant for human actors (e.g., managers, decision makers, etc.), our framework is devised to cater to the particularities and peculiarities of an ecosystem comprised of software and human agents.

3.1.1 Perception capabilities in smart systems

To understand and objectively evaluate prior approaches and solutions relating to SSs, a set of criteria is put forth. In this subsection, we discuss the criteria related to the perception capabilities that the SS needs to exhibit in order to effectively and efficiently capture the states of its internal and external environments. To better illustrate the rationale and motivation behind the choice of these criteria, we reflect back on the motivating scenario presented in 1.4 as a starting point for their adoption in SSs in general.

In the following, we use the term *situation* to refer to the result of the perception capability. Situations are basically snapshots of the environment surrounding the SS that can be discreetly and distinctively interpreted. These situations form the basis of any SS as they allow to focus the response efforts on particular events. Hence, the system should be able to detect the situation of the user while ensuring:

1. *a minimal to no user intervention*. Referring back to the $ss_{smartroad}$ scenario presented earlier in section 1.4, the speed of the vehicle $c_{vehicleSpeed}$ as well as the speed limit of the road $c_{roadSpeelLimit}$ both represent essential data to detect the $st_{riskySpeed}$ situation. They can be acquired with minimal to no intervention from the respective users. Hence, the system should detect the situation of the user based only on available data. In the best case, this is just more convenient for the user. In the worst case, the user might be unable to intervene (e.g., being unconscious due to fatigue or an accident).

2. *a good level of trust in the detected situation*. In the $ss_{smartroad}$ scenario, the $st_{riskySpeed}$ situation is detected based on the data representing the speed of the vehicle $c_{vehicleSpeed}$ and the speed limit of the road $c_{roadSpeelLimit}$. A faulty speed sensor or a mistake in entering the speed limit by the traffic authorities (or in the program managing traffic speed in the case of automatic speed management) can have bad repercussions either on user trust towards the system or in wasting resources on responding to the situation. Hence,

given the sensitivity in different application domains, the system needs to be precise when detecting the situations (e.g., the system should not invoke the Dangerous Nearby Area Service sr_{dna} because of a faulty speed sensor).

3. *the spatiotemporal validity of the detected situation*. Being dynamic, it is important that any SS be able to discern the spatiotemporal characteristics of a detected situation. In our motivating scenario, it is important to know the exact road where the $st_{riskySpeed}$ situation has been detected and the time at which it happened. This allows to limit the scope of operation of the response later on. For example, triggering the Intelligent Speed Adaptation service sr_{ivs} on one road would have a lesser impact on the traffic in the general area than triggering the same service on a bigger geographical range. Thus, the system should be able to determine the space within which a situation is valid (e.g., what routes are influenced by a traffic congestion) and the time estimate for which it will be valid (e.g., how long until scheduled road works are finished).

3.1.2 Response capabilities in smart systems

In this subsection, we discuss the criteria related to the response capabilities that the SS needs to implement so it can efficiently react to any events or situations that occur in its environment. We use the motivating scenario presented in 1.4 to introduce the rational behind the adoption of each criterion.

The term *response* is used in the following to refer to the different tasks that need to be performed in order to react to a particular situation. Responses reflect the ability of the system to wisely and efficiently use the resources at its disposal to address the situations that arise in the environment. Hence, the system should be able to respond to detected situations by:

1. *combining different existing tasks*. To promote the reuse and composition principles of Service Oriented Computing (Baresi et al., 2007), the domain expert should be able to combine existing, clearly defined and usable tasks to respond to detected situations (e.g., combining the Intelligent Speed Adaptation (ISA) and the Dangerous Nearby Area (DNA) can mitigate the effects of a traffic congestion situation and reduce the risk of the vehicle triggering the $st_{riskySpeed}$ situation getting into an accident).

2. *using control-flow constructs where applicable to orchestrate the tasks*. Organizing the execution of the tasks with control-flow constructs allows the reduction of dependencies between the tasks and a faster and more optimized execution (e.g., the In-Vehicle Signage (IVS) and the Electronic Road

Panel (ERP) can both be used to display information and their execution can hence be paralleled).

3. *selecting the best services to satisfy each task*. In smart environments, there are several services that can satisfy a particular task. The system should be able to select the service that best satisfies the required task for each user according to certain Quality of Service Properties (e.g., selecting the best Route Selection Service (RS) can be done based on response time).

3.1.3 Manual intervention capabilities in smart systems

As we mentioned above, Manual Intervention is a crucial capability that guarantees an added level of control and security over the system to mitigate the issues of any unwanted behavior. This capability allows the interaction between the user and the autonomous Perception-Response cycles of the SS. Like the previous subsections, we use the motivating scenario in 1.4 to explain the rational behind the adoption of each criterion.

The term 'Manual Intervention' does not only relate to the final users of the system but also to the managers and decision-makers behind the system. It represents the capability of the system to tolerate and integrate expert, personal and unwanted knowledge. Hence, the system should be able to enable Manual Intervention by:

1. *allowing the addition, modification or deletion of situations and how they are detected*. Though the Perception capability of the system is carried out autonomously, it is based on the input of the users. Hence, it should always be possible for the user to manage the situations that are not supported by the system. For instance, in the $st_{riskySpeed}$ situation, instead of using a radar to get the speed of the vehicle, the engineers decide to use the already present cameras on the road to derive that information.

2. *allowing the addition, modification or deletion of the way the system responds to detected situations*. Like the Perception, the Response capability of the system is also carried out autonomously. System engineers specify the actions to be taken once a situation has been detected in a process-like manner. Thus, the users of the system should be able to manage the responses to the situations that are known to it. For example, in our motivating scenario, adding the Police Pursuit sr_{pp} service to the $g_{riskySpeedResponse}$ response should be easily feasible by domain experts.

3. *managing authorizations to access and modify the behavior of the system or particular subsystems within the system*. As mentioned before, SSs can be seen differently depending on each user's interest in the system. The interest in the system delineates the logical (i.e., virtual) boundaries for each user and limits his ability to operate to those boundaries only. The system should, thus, be able to manage the authorization and access rights that each user has specifying which operations are possible and on which parts of the system. For instance, the driver u_{driver} cannot interfere in the way the $g_{riskySpeedResponse}$ goal is achieved but can invoke additional services that are made accessible to him.

4. *checking the improvements resulting from the adaptation algorithms*. SSs are characterized by their ability to continuously adapt to their environment though learning the habits and discovering particular trends and patterns within that environment. However, this ability means that the system is learning autonomously without the supervision of its users, which can lead to unwanted if not dangerous behaviors. This is why, the users should be appraised and kept in the loop in any change of the way the system detects and responds to particular situations.

The improvement of Smart Systems based on the PeRMI capabilities

Earlier in this chapter (in section 3.1), we discussed the capabilities of SSs under the PeRMI framework. Improving the capabilities of an SS means the ability to manage them (i.e., integrating new instances of the capabilities identified in the PeRMI framework, removing old deprecated ones and changing existing ones, etc.). This adaptation can be either manual (i.e., performed by the engineering team) or automatic (i.e., performed by the system itself). The goal is to improve the quality of the system via continuous monitoring, analysis of its functional and non-functional properties and how the user interacts with it. Hence, in this section, we present some of the techniques used to adapt and/or improve the perception, response and manual intervention of SSs.

Adaptation and improvement in perception Perception elements deal with how the system stays aware of its internal state, its users, as well as that of the environment around it (e.g., execution environment). Hence, the adaptation and improvement at this stage covers techniques and approaches aiming to enhance the perception of the system. This can be achieved for example by making the system sensitive to new stimuli or able to detect changes and events with higher

levels of certainty. Major techniques and contributions in this regard stem from 3 (three) different backgrounds as introduced below.

Learning-based approaches seek to leverage past data to establish the best policies to incorporate in order to get the best results. Results, in this sense, can be any set of parameters that serves as an objective metric to measure the perception of the system. In (Brdiczka et al., 2008), the authors present a framework for continuous situation learning in a smart home setting. The framework uses different learning techniques. First, Support Vector Machine (SVM) is used to detect the role of each entity in the smart home environment. Then, to detect and extract new situations, the authors use the Jeffrey Divergence (Puzicha et al., 1999) between sliding histograms. Last but not least, user feedback is used to improve the quality of the learning model and check valid new situations.

Model-based approaches define a set of structures and functions (or metrics) that enable engineers or software systems to detect particular abstractions in the environments. Note, this can be a design-time task where these structures change rarely, or a run-time endeavor where the elements constructing or influencing these abstractions and their importance change as the software system is running. For instance, (Souabni et al., 2018) define the *situation* as an abstraction of the way systems in ubiquitous learning environments perceive their entourage. These *situations* are inferred from smaller elements called *context* elements that reflect some attributes of the system's users. Based on the assumption that not all *context* elements contribute in the same way to the detection of a particular *situation*, the authors use Dempster-Shafer theory of evidence to propose an adaptive context weighting approach to situation detection.

Knowledge-based approaches focus on the optimization of data structures and quality parameters to improve and adapt their underlying systems. The improvement can be aimed toward one or multiple quality parameters depending on the targeted system. The authors of (Baumgartner et al., 2010) present several dimensions to assess data quality in a traffic management system (e.g., completeness, accuracy, timeliness, coverage, confidence, etc.). These dimensions form the bases of the improvement strategies that they consider in their work. To improve data quality, the authors proceed through data fusion by following a three-step process as presented in (Bleiholder and Naumann, 2009). The process starts with *schema matching* to transform data from different sources into a uniform (or common) representation. Then, using *duplicate detection* techniques, multiple and possibly inconsistent representations of the same objects are found. Finally, in the *data fusion* step, the detected duplicates are combined together into a single representation that is more complete.

Adaptation and improvement in response Response elements focus on how SSs deal with detected changes in their internal or external environment. At this stage, the improvements and adaptations are directed at the core structure and functions of the system, be it at the application level, the communication and network levels or the technical resources in use by the system. The goal here is to improve the system's performance by adapting its behavior in response to some stimuli (e.g., user preferences, resource sharing, Service-Level Agreement (SLA)s, etc.). For instance, making a smartphone switch to silent mode when its user is asleep or in a meeting is considered adaptive response behavior. Major techniques and approaches dealing with the improvement of the response capabilities of software systems come from three backgrounds same as those dealing with perception capabilities.

As mentioned above, learning-based approaches aim to leverage past data collected by the managed software system to extract patterns through which the adaptation of a system's component becomes possible and yields the needed results. In terms of response capabilities, results can mean different things. It can be parameters to be achieved like Quality of Service (QoS), adding components or services, switching between goals to be achieved for different users among others depending on how the system is built. For instance, the authors in (Zhao et al., 2017) propose a reinforcement learning-based framework to the generation and evolution of adaptation rules to deal with the shortcomings of in self-adaptive software systems that are based on adaptation rules (i.e., no guarantee for optimal adaptations and weak support for dynamic environments). The framework improves the flexibility of the adaptation logic and the quality of adaptation through combining Reinforcement Learning for online evolution of adaptation rules from real-time information about the environment and user goals, and Case-Based Reasoning (CBR) techniques for offline learning of adaptation rules from different goal settings.

Model-based approaches seek the improvement and adaptation of system responses to detected changes in the environment through the definition of structures, policies, rules and functions that constitute the elements necessary for the response and it's improvement. In this sense, there are several types of system models that were used in literature (e.g., architecture models, feature models, behavioral models). Generally, several types of models are used together to cover all the aspects relating to the response capability of a system and its possible configurations. In (Ballagny et al., 2009), the authors present an approach, called MOCAS (Model Of Components for Adaptive Systems), as a generic state-based

component model which enables the self-adaptation of software components together with their coordination. MOCAS adopts a behavioral adaptation to adapt the components of the target system using Unified Modeling Language (UML). Each component embeds a UML state machine to realize its behavior at runtime. It is installed in a container managing the adaptation process and ensuring its consistency. Adaptation is triggered when invariants related to the component's business properties are violated. The component supports updates of its specification while it is running.

Control-based approaches leverage advances in control theory to build adaptive systems. The goal here is to define control structures that reflect how the system works. The most commonly known example of control structures is the MAPE (Monitor - Analyze - Plan - Execute) and its knowledge-based extension the MAPE-K loops (Kephart and Chess, 2003a), which are closed feedback loops as they follow the general structure presented in figure 3.2. To be adaptive, a second control loop is necessary atop the existing one. The top loop is responsible for adjusting the controller of the bottom control loop to respond to changes of the controlled process. By linking the overall satisfaction towards the system to a business value indicator as feedback, the authors in (Peng et al., 2012) propose a control theoretic self-tuning method that can dynamically adjust the trade-off decisions among different quality requirements. A preference-based reasoning algorithm is also proposed to configure hard goals accordingly to guide the subsequent architecture reconfiguration.

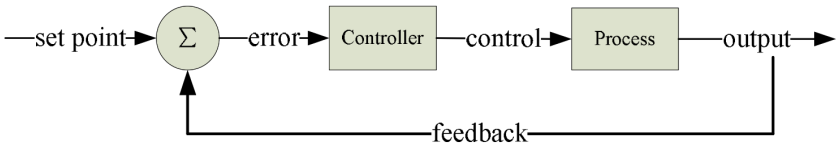


Figure 3.2 – Block diagram of a feedback control loop. Adapted from (Peng et al., 2012).

Adaptation and improvement in manual intervention Manual intervention elements aim to define the different ways that human users interact with target SSs. Like we stated earlier in chapter 2, these interactions can take several forms depending on the way systems are designed and the 'nature' of supported users. In this sense, adaptation and improvements seek to find the best way that each individual user or class of users can interact with the system for the benefit of both the user and the system. Since Graphical User Interface (GUI)s are the most common form of Human-Computer Interaction (HCI), it is not surprising that most of the literature in this direction is focused on the adaptation of GUIs, mainly to improve

user satisfaction (López-Jaquero et al., 2007; Schwartze et al., 2010). However, other researchers have worked on adaptations of multi-modal interaction (Dumas et al., 2012) and adaptations that are aimed towards impaired users (Biswas et al., 2014).

Model-based approaches to the adaptation and improvement of user-system interactions have been mainly focused on the Model-Driven Engineering (MDE) methodology. This entails the definition of models (or metamodels) and transformations that enable the monitoring of the user's and system's states, reasoning on the best course of interaction and effectively acting on the elements of the chosen interaction medium. For instance, in (Khaddam et al., 2015), the authors present *Adapt-First* which is an MDE transformation approach for supporting User Interface Adaptation that seeks to enhance user satisfaction through the consideration of his context of use. To reduce the complexity related to the amount of context elements and information that influence the adaptation, the authors propose a process of five steps. First, a classification of the adaptation rules at the desired level of application is performed. Second, an analysis of the adaptation rules to extract the inputs needed from the source and context model is performed. Third, an intermediate model is constructed, consisting of an enriched version of the source model by exploiting the elements within the context model. The fourth step is to design translation policies where the goal is the adaptation to the context of use and implementing adaptation rules at the same level of abstraction. Last, the fifth step, is to design the concretization policies, where the goal is to set the manner in which the adaptations go from the abstract level to the instance level.

Learning-based approaches to the adaptation and improvement of user-system interaction keep changing as the research keeps advancing mainly in Machine Learning (ML). One of the highly efficient set of techniques in ML in the past decade has been Deep Learning (DL), which goes a step further from prior techniques in ML by taking care of the feature extraction process. Interface adaptation has been one application of DL techniques. To enhance system usability and user satisfaction, real-time contextual adaptation of user interfaces that is tailored to each user is required. To that end, the authors in (Soh et al., 2017) propose an architecture for personalized AUIs (Adaptive User Interfaces) that incorporates a deep sequential recommendation model leveraging upon developments in (1) deep learning, particularly gated recurrent units, to efficiently learn user interaction patterns, (2) collaborative filtering techniques that enable sharing of data among users, and (3) fast approximate nearest-neighbor methods in Euclidean spaces for quick UI (User Interface) control and/or content recommendations.

Specifically, their proposal leverages Deep Recurrent Neural Networks (DRNN) to learn a latent embedding space that permits data-sharing in a collaborative filtering manner. By querying this shared latent space, the AUI is able to exploit similar usage patterns from across the user base to make personalized adaptations for relatively new users and novel scenarios.

3.2 C2IoT Framework

Current software solutions that are developed for Smart Environment (SE)s present a fragmented landscape (Truong and Dustdar, 2015). This is due on one hand to the way the solutions are built (i.e., application-specific, in an *ad hoc* manner and from scratch) and on the other hand to the erratic definition and use of enabling technologies. To study and deal with the second issue, we briefly introduce in this section the C2IoT framework as the reference architecture for our approach (Faieq, Saidi, et al., 2017b). C2IoT stands for Cloud-based Context-aware Internet of Things, it is a framework in which we studied the synergies between the different enabling technologies and paradigms involved in the smart city, but that can be generalized for any smart environment. The architecture presented in figure 3.3 is derived based on the analysis of the requirements of SEs and the literature. The founding principle is to allow the three key enabling technologies (being Cloud Computing, Internet of Things and Big Data) to, (i) support a uniform methodology for the development of software solutions for SEs and (ii) support one another in optimizing and/or improving their own performance. The framework can be seen and analyzed from three perspectives.

3.2.1 View Perspective

The *View Perspective* or *Vertical Perspective* allows to determine the functional aspects of each enabling technology. The basis of the framework is the *Cloud View* and its service models. It does not only provide the necessary flexibility to store, process and deliver services while dealing with the challenges related to smart environments (e.g., scalability, heterogeneity, resource consumption, etc.), but also, its service models clearly delineate the scope of each service model. The *IoT View* (Things as a Service (TaaS)) is responsible for sensing the state of the world through collecting the data that is needed for any smart solution and also changing that state when the need arises. The *Big Data View* (Big Data as a Service (BDaaS)) provides the necessary services and tools to deal with the challenges related to the characteristics of data in SSs (e.g., efficient storage, fast processing or understanding through visualizing the data, etc.).

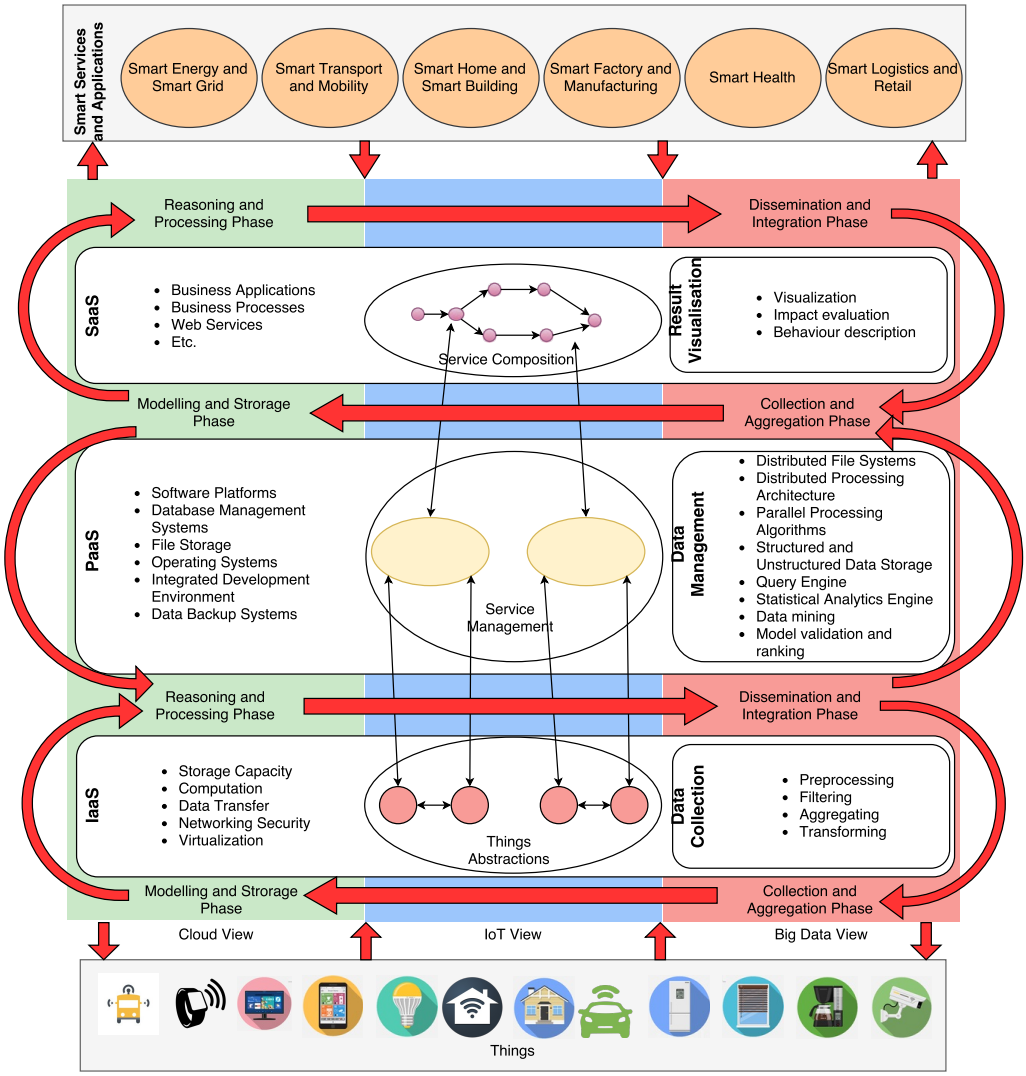


Figure 3.3 – Overview of the C2IoT Framework. From (Faieq, Saidi, et al., 2017b).

3.2.2 Layer Perspective

The *Layer Perspective* or *Horizontal Perspective* presents the collaboration possibilities between the different enablers in each layer of the framework based on the service models of cloud computing (Mell and Grance, 2011). In the *Infrastructure as a Service* layer, the idea is to present a cohesive, flexible and efficient infrastructure that is capable of effectively collecting, pre-processing and physically storing the data coming from the physical sources in the SE. The *Platform as a Service* layer provides the tools to manage the different offerings related to the operations offered by the underlying layer, thus deals with managing the services responsible for collecting the data, its pre-processing and storage and provides a basis to develop and host the applications. The *Software as a Service* layer provides several models, mediums and ways to deliver the products encompassing the business

logic of the applications that were built on the underlying layers.

3.2.3 Integration Perspective

The C2IoT framework is centered around context. The *Integration Perspective* capitalizes on the fact that context data is not only collected and leveraged by each enabler (e.g., Clouds, Things, Big Data) but also generated by each one of them and that this generated data can be leveraged by other enablers. This provides an integrated way to think and reason about context data and the enabling technologies and allows a flexible, cross-layer information flow exchange. For instance, objective Quality of Service data regarding the execution of a web service at the Software as a Service (SaaS) layer can be collected, pre-processed and visualized to allow a better understanding of its usage and improve its quality for future executions; which may require affecting modifications at the Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) layers of the framework.

3.3 Related works on the design, development and improvement of Smart Systems

In this section, we summarize the contributions of selected research works that have dealt with the design and development of SSs. The presented works have been selected based on their coverage of the different dimensions described in the previous chapter and the frameworks that we introduced earlier in this one. A synthesis is provided in the end of each subsection to position the different related papers within the proposed frameworks.

3.3.1 On the design of related smart system approaches

In this section, we focus on analyzing the design approaches adopted in the development of SSs according based on the capabilities of the PeRMI framework and the application domain that are taken into account by each SS in the selected related works. We will describe which capabilities were taken into account and present the key concepts and techniques involved. The goal is to identify some commonalities that may be leveraged to make the process of building SSs more generic. In general, very little work has been done on the engineering of systems that are designed for SEs in literature. Most of the works focus on problems related to data (e.g., data management, data access, data processing, etc.). Mostly, scientists and engineers stuck to long standing practices to develop their solutions or tried to adapt them to the specifics of the targeted application.

To tackle the issue of resource scarceness, particularly water, the authors in (Angelopoulos et al., 2011) propose a service-based smart system for home plant irrigation. The SS is capable of sensing different context parameters related to the plants and their environmental conditions. It is then able to adjust water flow efficiently for optimal irrigation of the plants. This adjustment is based both on the type of the plant and the environmental conditions.

In his paper, Demirkan (Demirkan, 2013) discusses the challenges related to today's smart environments, especially healthcare, from different perspectives spanning financial, quality, collaboration and political views. To deal with these issues, the author proposes a conceptual building block for a smart healthcare system. This building block is based on a Service-Oriented approach to deliver on the functional aspects of the healthcare systems. The goal is to promote value co-creation through the reuse and on-the-fly composition of provided services using virtual resources.

(Sivamani et al., 2013) take a knowledge-based approach to identify the core concepts used in a smart vertical farm. Specifically, they model the system's components as a set of ontologies at different levels of abstraction. At the highest level, they identify the system, the users, the context, the services, environmental parameters, the network and the location as the building blocks of the smart vertical farm system. Through these concepts, the paper deals with monitoring the situation in the farm and control the responses if a situation arises.

In (Kabir et al., 2015), the authors propose a service-based approach to building adaptive and context-aware smart homes based on machine learning. They adopt a three layer architecture containing a sensor layer to monitor the user and the home, a middleware layer to store and process collected data and an application layer to select the appropriate services based on the user's context. Instead of using rules, the authors use machine learning to detect the user's situation and select the services to be executed.

(M. Chen et al., 2016) put forth Smart Clothing to address the issue of comfort in classic wearables. The authors discuss how smart clothing can be a solution for unobtrusive health monitoring. Although the paper focuses on technological aspects, the proposed architecture involves a signal collection module for different types of health data. Depending on the application (e.g., emotion care, emergency response, etc.), the collected data are processed to detect events and situations of interest.

In their paper, (Park et al., 2017) focus on smart manufacturing and propose a predictive situation awareness model where the aim is to improve the manufacturing process. To that end, they capitalize on the high automatability of

manufacturing processes and the data provided by the different smart things involved in the aforementioned process to achieve situational awareness and to predict future situations. The authors operate under the assumption that being able to predict future situations would reduce costs and time while improving the quality of the manufactured artifacts.

The paper (Sultan and Ahmed, 2017) presents the SLASH framework for designing and implementing smart home systems that are both adaptive and self-learning. Sensors are installed to collect data over time. Particular situations are then learned from these data reflecting user behavior. The responses to these particular situations are carried out using actuators from smart devices.

To improve the effectiveness and efficiency of Intelligent Transport Systems, (J. Chang et al., 2017) identify the integration between data aggregation and business services as an important challenge. Towards that goal, they design a context-aware service system that is meant to improve road safety in smart transport environments. The designed system is based on two components, (1) a semantic context model to represent the context of the driver, (2) a reasoning engine to deduce the driver's situation from his context to which a service configuration is associated.

In (Mshali et al., 2018), the authors propose a context-aware e-health monitoring system targeted at the elderly and isolated individuals living alone. The system collects context data, mainly using geriatric norms and scales, to check the status of the smart home user. These data also go towards identifying the activities that the user performs within the home, which in turn allows the learning of the user's behavior. Once a user behavior profile is established, the proactive detection of possible risky situation becomes possible using forecasting techniques.

The paper in (Gullà et al., 2016) proposes a smart kitchen system for aiding people with different types of disabilities in accomplishing kitchen-related tasks. The system evolves around the Web UI that serves to guide the targeted group of users accomplish their desired goals using an ability-oriented design approach. At the backend, the system collects several data about the user and his surrounding (e.g., behavior, preferences, disabilities, etc.) to shape its Web UI in the most appropriate manner to the user.

In (Abdelhafidh et al., 2018), the authors propose a smart system that focuses on the monitoring of water pipelines. The paper describes a system architecture that deals with data collection, data storage, data processing and analytics. The authors implement a model-based leak detection mechanism to allow the detection of leak situation by leveraging data sent by sensors installed in the pipelines. The responses to the detected leak situation is implemented as an adjustable

mechanism for controlling the water flow in the pipeline.

To deal with the user-centric vision characteristic of smart environments, the authors in (Palanca et al., 2018) use a goal-based approach to design and implement self-adaptive service-based smart home systems. This approach aims to facilitate the interactions between the human-user and the smart system. They rely on a formal definition of the goals and a semantic description of the services to propose an architecture that is capable of creating and adapting predefined plans of action, in the form of service compositions, to reach each goal. To reach a plan of action, the authors use a Multi-Agent System methodology while respecting temporal constraints.

(H. Lee and J. Lee, 2018) propose a Smart Service System-based Smart Factory (4SF) architecture that can be added to existing Smart Factory systems without major changes and effort. The architecture serves to guide the design and development of Smart Factory systems based on the identification of high-level concepts. The OODA loop (Observe-Orient-Decide-Act) is used to organize the different views involved in the design of the system. It allows the definition of the parameters to be monitored and their organization, how to make decisions from these data and how to implement actions to improve the factories performance.

In their paper, (Santofimia et al., 2018) propose a service-based architecture to deal with the integration and interoperability issues in building smart homes. The authors propose a new product metamodel to enable capturing service capabilities at different levels (syntactic and semantic). They, later on, capitalize on their product metamodel to propose a technique that is capable of on-the-fly service composition and reconfiguration. The resulting system is considered autonomous and self-learning.

The paper in (Sood et al., 2018) presents a smart flood management framework. The proposed architecture can be divided into two capabilities. First, monitoring of flood causing attributes is done by means of sensors installed to create hexagonal structures. They are transferred and stored centrally to allow their processing. Second, forecasting and prevention are done using different AI methods through the processing of collected data and available prevention attributes.

Synthesis

This synthesis is meant to position the related works according to the PeRMI framework presented in section 3.1. This would allow us to extract the general trends and concepts that are in use within the software engineering community to deal with the issues of SSs. Also, the analysis of the different related works, their strengths and weaknesses should help us identify the gaps in current literature.

Table 3.1 presents our analysis of the related works according to the capabilities and elements of the *PeRMI* framework. The table indicates what the identified works have discussed or addressed in the designs or design approaches they proposed.

As can be noticed in table 3.1, most of the related works focus on the automation of the *Perception* capability. This is mostly achieved through the monitoring of the dimensions (i.e., attributes) that are of interest to the application domain and learning the user's behavior. *Response*-wise, customization is the most used technique. This is only natural as most of the literature on smart systems has focused on the smart home as the potential application, where the comfort of the user is the priority. Overall, *Manual Intervention* is the most understudied capability. This can be attributed to the controversial idea that SSs are autonomous systems exclusively, which also explains the focus on the automation aspects of SS in literature.

3.3.2 On the technologies of related smart system approaches

The goal of this section, is to present how related smart system works have made use of technologies to implement and enable building SSs. Overall, the use of technologies is erratic across related works and depends on the goals of the proposed approaches. We focus particularly on the use of Cloud Computing (CC), Internet of Things (IoT) and Big Data to extract best and common practices within the research community.

To tackle the issue of resource scarceness, particularly water, the authors in (Angelopoulos et al., 2011) propose a service-based smart system for home plant irrigation. The approach presented in the paper is based on Wireless Sensor Network (WSN)s. Indeed, different sensors and actuators like soil sensors and electro-valves are used to monitor the soil's humidity and air temperature and also to control water flow. The technology is completely localized to the home environment, making it a small-scale IoT application.

(Demirkan, 2013) presents the SHSF (Smart Health System Framework) to conceptualize and build smart health systems. The paper makes CC the main technology in its building block of the framework. It discusses the role of cloud computing in streamlining service offerings across the healthcare industry. It also relies on sensing technologies (e.g., RFID, Barcode, etc.), different web technologies (i.e., web 2.0, 3.0 and semantic web) and big data analytics to hold patient records and offer simple and understandable products to both health professionals and patients.

In (Sivamani et al., 2013), the authors propose a vertical farm ontology (VFO),

Works	Perception			Response				Manual Intervention		Traceability
	Automation	Trust	Validity	Collaboration	Optimization	Customization	Perception Management	Response Management	Authorization	
(Angelopoulos et al., 2011)	✓				✓	✓				
(Demirkan, 2013)	✓	✓		✓	✓			✓	✓	
(Sivamani et al., 2013)	✓		✓	✓		✓			✓	
(Kabir et al., 2015)	✓			✓		✓		✓		
(M. Chen et al., 2016)	✓		✓		✓	✓				✓
(Park et al., 2017)	✓	✓	✓		✓		✓			
(Sultan and Ahmed, 2017)	✓			✓		✓				✓
(J. Chang et al., 2017)	✓		✓	✓			✓	✓		
(Mshali et al., 2018)	✓	✓			✓	✓	✓			
(Gullà et al., 2016)	✓		✓	✓		✓	✓	✓		
(Abdelhafidh et al., 2018)	✓		✓		✓					
(Palanca et al., 2018)	✓		✓	✓		✓	✓			✓
(H. Lee and J. Lee, 2018)	✓		✓	✓			✓			✓
(Santofimia et al., 2018)	✓			✓		✓		✓		
(Sood et al., 2018)	✓		✓		✓			✓		

Table 3.1 – Analysis of the related works with respect to the *PeRMI* framework.

to tackle the issues of integration in Smart Agriculture. The authors mainly focus on WSN as the backbone of their smart vertical farm system. This is a strong indication to the use of the IoT stack as the technological enabler of the system. Devices like heat sensors, thermostats, humidifiers and air conditioners, are supposed to be installed to monitor and control the environmental parameters within the scope of the vertical farm.

(M. Chen et al., 2016) put forth Smart Clothing to address the issue of comfort in classic wearables. Several technologies are discussed as possible implementation choices. However, the proposed architecture uses smart clothes as data sources, Cloud Computing to provide storage and processing resources, and Big Data analytics to extract the intelligence needed depending on the targeted application (e.g., fitness, elderly care, emotion care, etc.).

In their paper, (Park et al., 2017) focus on smart manufacturing and propose a predictive situation awareness model where the aim is to improve the manufacturing process. It assumes that smart manufacturing systems use IoT to both monitor the functional aspects of the manufacturing process (and the related quality parameters) and to adapt the process in order to accommodate different settings (e.g., failures at the production lines).

The paper (Sultan and Ahmed, 2017) presents the SLASH framework for designing and implementing smart home systems that are both adaptive and self-learning. Like most smart systems, the SLASH framework relies on IoT to implement data collection processes. Big Data techniques are used to learn user behavior from the collected data. This would allow smart home systems to predict what the user would do after an event is triggered.

In (J. Chang et al., 2017), the paper describes the design of a context-aware service system for smart transport environments. The implementation relies on the IoT to monitor context information related to the driver, the car and the environment. The proposed architecture also describes the use of edge computing units to transfer data between the vehicles and the roadside units to minimize communication delays.

The paper in (Gullà et al., 2016) proposes a smart kitchen system for aiding people with different types of disabilities in accomplishing kitchen-related tasks. The system uses smart objects to capture and control the usage of kitchen equipment (e.g., refrigerator, microwave, oven, dishwasher, etc.). The smart kitchen is considered as a cloud platform that gives network storage service and can be accessed from logic network applications.

In (Abdelhafidh et al., 2018), the authors propose a smart system that focuses on the monitoring of water pipelines. The proposed system relies on cognitive

IoT to implement a WSAN (Wireless Sensor Actuator Network) that is capable of sensing data related to the pipeline and controlling its water flow. CC is used to store the generated data, while Big Data analytics are used to process the data in order to make the decision about opening or closing the water flow within the pipelines.

(H. Lee and J. Lee, 2018) propose a Smart Service System-based Smart Factory (4SF) architecture that can be added to existing Smart Factory systems without major changes and effort. The authors map each step in the OODA cycle to a particular technology stack. Figure 3.4 shows the mapping between the functional elements of the approach against the technologies adopted for their implementation.

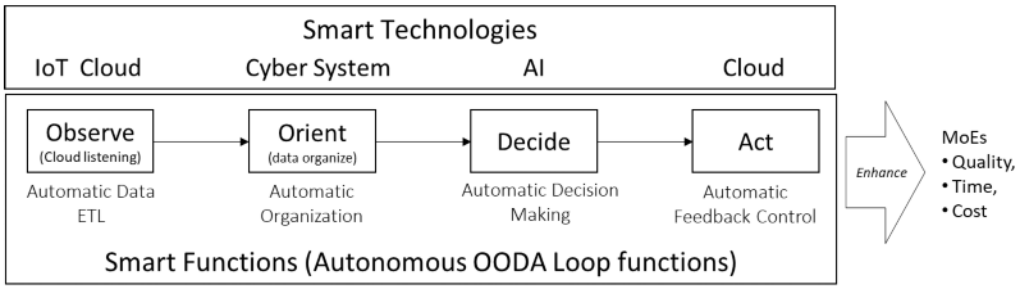


Figure 3.4 – Mapping between OODA loop and enabler technologies. From (H. Lee and J. Lee, 2018)

In their paper, (Santofimia et al., 2018) propose a service-based architecture to deal with the integration and interoperability issues in building smart homes. The proposed approach is based on the IoT as it's the main provider of services and data. It also encompasses services from technologies like CC, Fog and edge Computing to store and transfer the data homogeneously through an abstraction layer. An AI scheduler is also used to plan the composition of service to achieve the goals set by the system as responses to particular events.

The paper in (Sood et al., 2018) presents a smart flood management framework. It relies heavily on IoT to collect data on flood causing and preventing attributes in real time. Fog Computing is also used to allow some preprocessing at zone-level before sending data for permanent storage in the cloud. To enable the processing of data for forecasting and prevention, big data technology is used in both streaming and batch forms.

Synthesis

In the following, we focus on analyzing the identified related works with respect to the C2IoT framework presented in section 3.2. The goal here is to

extract some general trends on the use of the aforementioned technologies in SS and to help identify the gaps in the literature. Note, given the numerous elements and techniques involved in each technology stack and layer, we only check their use and not their relevance.

Table 3.2 presents a summary of the analysis. It was conducted based on the different perspectives of the *C2IoT* framework. In the *View* perspective, a check mark means that at least some elements of the technology stack were used in the associated research work. Similarly for the *Layer* perspective, a check mark means that at least elements of the layer were used or discussed in the associated work. However, this perspective is tightly coupled to the use of CC and thus only works using CC are considered. The *Integration* perspective is different. Since integration means the use of one technology element to improve another, there are several possible combinations that need to be considered (e.g., the use of IoT data to improve CC operations). Hence, a column was created for each view and layer depending on whether the integration type (i.e., horizontal or vertical). In the example of (Sood et al., 2018), the table should read "BD techniques were used to improve IoT operations".

Works	View Perspective			Layer Perspective			Integration Perspective					
	CC	IoT	BD	IaaS	PaaS	SaaS	Horizontal			Vertical		
							CC	IoT	BD	IaaS	PaaS	SaaS
(Angelopoulos et al., 2011)		✓										
(Demirkan, 2013)	✓	✓	✓		✓							
(Sivamani et al., 2013)		✓										
(M. Chen et al., 2016)	✓	✓	✓	✓	✓							
(Park et al., 2017)	✓	✓		✓								
(Sultan and Ahmed, 2017)		✓	✓									
(J. Chang et al., 2017)	✓	✓		✓								
(Gullà et al., 2016)	✓	✓		✓	✓							
(Abdelhafidh et al., 2018)		✓	✓									
(H. Lee and J. Lee, 2018)	✓	✓	✓	✓	✓							
(Santofimia et al., 2018)	✓	✓		✓	✓	✓						
(Sood et al., 2018)	✓	✓	✓	✓	✓			BD				

Table 3.2 – Analysis of the related works according to the *C2IoT* framework. The Integration Perspective is clearly underexplored as most of the works on SS focus on the Layer and View Perspectives.

Table 3.2 gives clear indications to the strong association between the IoT and SS. Indeed, every single identified related work has used at least the sensing and/or actuation capabilities of the IoT. This is normal as IoT is responsible for data collection which constitutes the basis of any SS. Meanwhile, the use of CC and Big Data remains somewhat erratic depending on the targeted applications and the objectives of the research. At the *Layer* perspective, most of the literature

either focuses on the IaaS or PaaS models. This is very normal because SaaS models are reserved for end-users due to the limited control over the system. From the *Integration* perspective, with the exception of (Sood et al., 2018), there is a severe lack of studies that leverage data from one enabler technology to improve the operational capabilities of other enablers. This fact advocates for the need for more integration and collaboration between technological enablers either horizontally and vertically.

3.3.3 On the improvement of related smart system approaches

In this section, we analyze related works dealing with the design and implementation of SS to extract adaptation possibilities. Both design and technological aspects are taken into account to position the adaptation efforts. For design aspects, adaptations targeting the way services are planned and selected are of particular interest.

To tackle the issue of resource scarceness, particularly water, the authors in (Angelopoulos et al., 2011) propose a service-based smart system for home plant irrigation. Since every plant has its needs of water depending on different soil and environmental conditions, the authors use adaptation rules to control the states of the electro-valves. The conducted experiments show the efficiency and effectiveness of this irrigation scheme.

(Demirkan, 2013) presents the SHSF (Smart Health System Framework) to conceptualize and build smart health systems. The framework discusses the issues related to healthcare services as they cannot be transported or stored. Adaptive allocation is thus discussed at two levels. First, at the level of manual interventions by medical staff by collecting information about the processes that require the most risk and optimization staff utilization. Second, at the IT level, to ensure an optimal functional capacity to save costs. Analyzing and predicting the demand is a component of the SHS Framework.

In (Kabir et al., 2015), the authors propose a service-based approach to building adaptive and context-aware smart homes based on machine learning. To make the smart home system adaptive, the authors rely on a class of reinforcement learning called Temporal Differential (TD). TD leverages user feedback to learn the best plans of actions (i.e., service compositions) to deal with new situations instead of relying on prescribed plans. This approach presents the advantage of being automatic while it also suffers from the cold start problem.

In their paper, (Park et al., 2017) focus on smart manufacturing and propose a

predictive situation awareness model where the aim is to improve the manufacturing process. The model relies on the General Boyd's OODA (Observe-Orient-Decide-Act) adaptability loop (Boyd, 1995) to allow for feedback control and the adaptation to current product lines conditions. The proposed prediction model also allows for proactive adaptation in the production line when the confidence in the occurrence of an impending situation is high.

The paper (Sultan and Ahmed, 2017) presents the SLASH framework for designing and implementing smart home systems that are both adaptive and self-learning. The SLASH framework aims at automation some of the functions of some actuators based on user behavior. It leverages machine learning techniques to adapt the home to the user's routine and preferences.

In (J. Chang et al., 2017), the paper describes the design of a context-aware service system for smart transport environments. A composition of business services is invoked when a situation is detected. To make this system adaptive, the authors propose a priority algorithm to control the invocation of business services based on the different severity levels of the driver's situation.

(Mshali et al., 2018) propose a context-aware e-health monitoring system targeted at the elderly and isolated individuals living alone. To minimize the data circulating within the network, the system uses adaptation at two levels. First, the system discards irrelevant context data for monitored activities. Second, an adaptation technique is proposed to dynamically adjust sensing time intervals and also adjusting the times of the day when the sensing is performed.

The paper in (Gullà et al., 2016) proposes a smart kitchen system for aiding people with different types of disabilities in accomplishing kitchen-related tasks. An adaptation engine is designed to improve the usefulness of the system and make it more customized to the end-user. Two adaptation strategies were adopted in this context. A static adaptive behavior that works with static information (e.g., type of disability) about the user and a dynamic adaptive behavior that leverages data reflective of the user's behavior.

In (Abdelhafidh et al., 2018), the authors propose a smart system that focuses on the monitoring of water pipelines. The system focuses on the detection of leaks within the pipeline. However, adaptation is implemented as a dynamic valve opening-closing mechanism that allows controlling the water flow within the pipeline and thus minimizes water waste as illustrated in figure 3.5.

In (Palanca et al., 2018), the paper proposes a goal-oriented approach to design smart home environments. The authors discuss and argue the importance of self-adaptive behavior. In this paper, this characteristic is implemented using an MAS (Multi-Agent System), mainly used to reach a plan of action (i.e., service

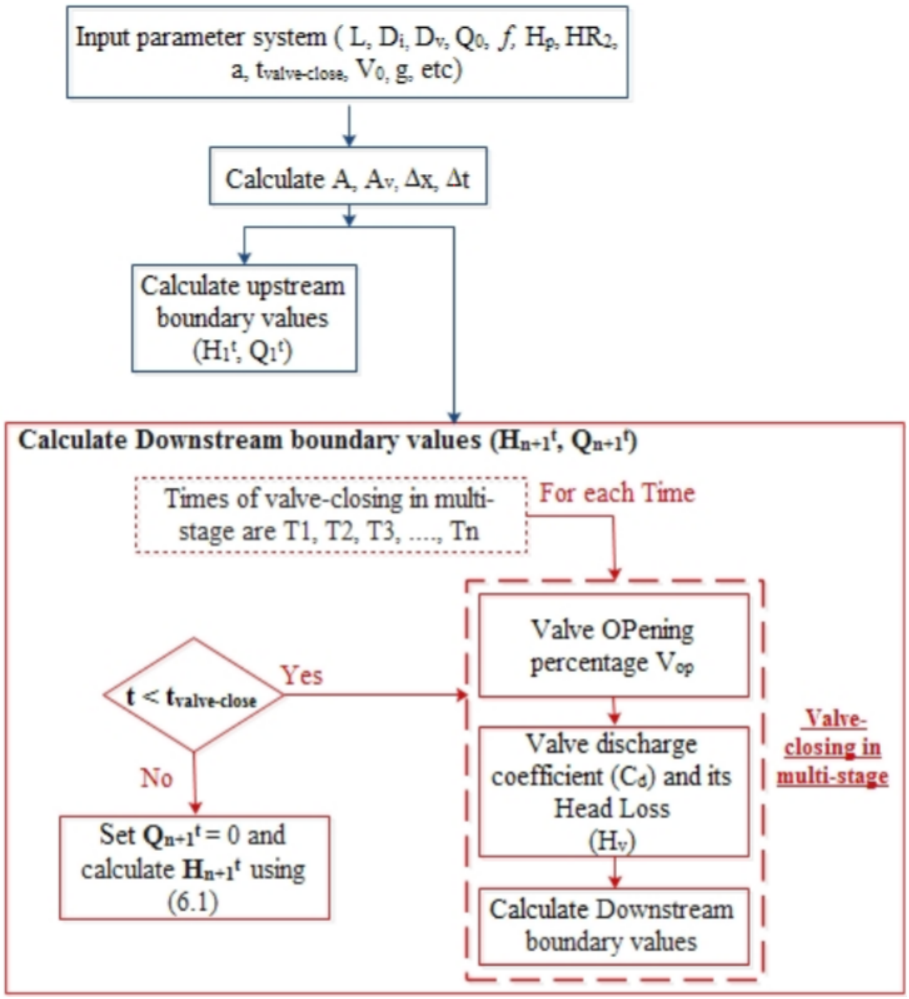


Figure 3.5 – Procedure of model-based leak detection. From (Abdelhafidh et al., 2018)

composition) while respecting temporal constraints. This allows the system to reuse, discover and adapt plans to deal with unforeseen situations.

(H. Lee and J. Lee, 2018) propose a Smart Service System-based Smart Factory (4SF) architecture that can be added to existing Smart Factory systems without major changes and effort. Adaptability is tackled by design through the use of the adaptability loop OODA to guide the design of the Smart factory system. However, no explicit techniques or tools were mentioned to operationalize this adaptation since the paper mainly discusses design and technological issues.

In their paper, (Santofimia et al., 2018) propose a service-based architecture to deal with the integration and interoperability issues in building smart homes. To deal with interoperability, they propose a virtual-network messaging protocol that is technology- and location-independent called IDM (Inter-Domain Messaging) serving as an adapter (Villa et al., 2017). The protocol basically adds a

layer of abstraction over existing network protocols allowing their interoperability through their adaptation to a common standard.

The paper in (Sood et al., 2018) presents a smart flood management framework. To save energy at the IoT devices level, the authors propose a shutdown policy for IoT devices based on the probability of flood in each zone. The use of adaptation rules to shutdown devices according to a static threshold (i.e., the probability of flood) makes the system easily adapted, however it requires manual configuration.

Synthesis

We have seen in the previous sections the role adaptation techniques play in improving (in terms of performance and flexibility) the performance of systems in general and SSs in particular. We focused particularly on adaptation techniques targeting system design elements and technology elements in all sorts of systems and applications. In this section, we explored how the identified related works have used adaptation techniques in SS engineering. Particularly, and due to prominence of service-based adaptations in literature, we do not include architectural styles within our analysis. The aim here is to extract trends in the use of adaptation and common adaptation concepts. These trends would make it possible to create abstractions that can be integrated at design time to allow for easy adaptations later at runtime.

A summary of the conducted analysis is presented in table 3.3. We analyze the identified related works based on the elements of the *PeRMI* framework for design capabilities and the *C2IoT* framework for technological enablers. Note, only the *View* perspective of the *C2IoT* is analyzed to allow a clearer view on the technologies involved in the adaptation. A check mark means that at least some of the concepts related to different capabilities and enablers were equipped with an adaptation capability to improve or add flexibility to some of their operations. The quality of the adaptations in each work is out of the scope of the manuscript as they fall outside the goals of conducting this analysis.

Several insights can be extracted from table 3.3 regarding the use of adaptation techniques in SS design and implementation. From a design perspective, adaptation is mostly used to improve the *Response* capability of the SSs. This is arguably due to the fact that SSs have a higher level of control over the *Response* capability compared to the other capabilities, mostly because they are influenced by the behavior of the human user. This conclusion is also supported by the small number of works that have worked on adaptation to help improve the *Manual Intervention* capability. From the technology perspective, as would be expected, most works target the IoT for possible adaptation strategies. This is due to the

Works	Design			Technology		
	Perception	Response	Manual Intervention	Cloud Computing	Internet of Things	Big Data
(Angelopoulos et al., 2011)		✓			✓	
(Demirkan, 2013)		✓	✓	✓		
(Kabir et al., 2015)	✓	✓				
(Park et al., 2017)		✓			✓	
(Sultan and Ahmed, 2017)	✓				✓	
(J. Chang et al., 2017)	✓				✓	
(Mshali et al., 2018)	✓				✓	✓
(Gullà et al., 2016)	✓	✓			✓	
(Abdelhafidh et al., 2018)		✓			✓	
In (Palanca et al., 2018)		✓			✓	
(H. Lee and J. Lee, 2018)	✓	✓	✓			
(Santofimia et al., 2018)	✓			✓	✓	
(Sood et al., 2018)		✓			✓	

Table 3.3 – Analysis of the related works according to the adaptation levels.

fact that most SSs rely inherently on the sensing and actuation capabilities of the IoT like we concluded earlier in section 3.3.2. It is however surprising that, while a lot of the identified related works have used elements of CC in their approach, very little works have considered improving their performance.

3.4 Discussion

In this chapter, we started by introducing some methodological frameworks that were developed based on our analysis of the scientific context and background of this thesis presented back in chapter 2.

We argue that the capabilities of the *PeRMI* framework can be a helpful tool to analyze SSs. It presents the advantage of being holistic as it covers different aspects of an SS from the way it collects data and processes it to form the system's perception of its world, to the way it responds to particular events and situations and interacts with the end user. It is also extensible and flexible because the elements of interest in each capability can differ based on the application's functional and non-functional parameters. Some possible adaptation and improvement techniques that have been used in literature according the capabilities introduced within the *PeRMI* framework have also been introduced and discussed. This allowed us to add a new dimension to our analysis of the related smart system works that were introduced later in the chapter.

To homogenize the use of the technologies investigated in chapter 2, we proposed the *C2IoT* framework. We argue that this framework can be a useful tool to normalize the use of enabling technologies for SS development as it contains the elements of the three technologies, namely CC, IoT and Big Data. The use

of context information to allow the collaboration between the different elements of these enablers also allows the evaluation of their integration, which is also an interest to many researchers (Alansari et al., 2018; Botta et al., 2016; Hashem, Yaqoob, et al., 2015). Note, the fact that *C2IoT* is based on CC and its service models makes it hard to evaluate solutions that do not explicitly use CC in their architecture (from a *Layer* perspective). However, this can be remedied by a finer analysis at the *View* perspective through the specification of which components of the IoT and/or the Big Data stacks are used in the solution.

After the introduction of the methodological frameworks, we introduced a summary of the selected related works according to the different dimensions of interest in this thesis, namely, the design capabilities, the technologies and the improvements.

First, from the perspective of design capabilities, the conducted analysis of the related works has produced some interesting insights. However the conclusions drawn remain subject to application-dependence. Indeed, most of the works focus on specific aspects related to the application domain of the SS. For instance, SSs targeting the home domain tend to focus on user's comfort which translates to automating the perception that the system has of its user; while SSs targeting agriculture would focus on how the system should respond to particular events (e.g., adapting water flow). This application-dependence, plus the lack of a common way to define SS in terms of the involved concepts makes the SSs hard to integrate and thus challenging to weave in a much needed System of Systems (SoS) fabric. Moreover, it does not help in establishing best practices and benchmarks on SS engineering and development, which would allow the practice to become more standardized.

The past points have led us to focus our efforts on the following needs to put a stepping stone towards having a methodical way to engineering smart systems:

- A general product model that encompasses the concepts of the *PeRMI* framework and that can also easily be associated with the *de facto* methodologies like Service Oriented Computing (SOC) and context-aware engineering to facilitate their implementation.

- A design method that would allow the identification and the definition of the model's concepts depending on each application domain and also operationalize the links between these concepts.

Second, from the technological perspective, the analysis of the related works on SS described in the previous section shows the disparities between the used

enablers. This is most likely to be caused by the different scopes of the application domains targeted by the SSs. For example, a smart home is less likely to use the cloud because is doesn't need that much infrastructure and also because there may be privacy and security concerns. However, the most interesting observation is that very little research has been done to integrate the previously mentioned technologies together. Here, integration means not only collaboration but also mutual improvement. Indeed, the use of feedback data from one technology component to improve the operations of another component could be very beneficial to the providers of these technology stacks. We attribute this gap to the little efforts done by major IT companies in the field to develop generic and interoperable solutions and facilitate migration between them. Another important observation is that most of the literature considers these technologies as a mere tool to the deployment of their solutions. However, we argue that in order to truly create an SE, these technologies, or at least general concepts pertaining to these technologies, need to be though of as part of the design of the solutions.

To bridge the gaps and tackle the issues mentioned previously, we argue that the following points should be taken into consideration in order to have a uniform way to integrate technological advances into SS:

- A way to describe and define technology concepts within the product model of the SS. We argue that this would allow to weave technology into the design of the targeted SSs.

- A way to allow the integration between the discussed technologies in the pursuit of mutual benefit. We argue that way, the different components of the *C2IoT* framework can improve on another.

Last but not least, from the improvement perspective, the analysis of the identified related works according to the adaptation type has shown that there is an uneven focus on the different capabilities and technologies. Moreover, given the user-centric vision of SE, it is surprising that only few works have worked on improving *Manual Intervention* among the identified works. Note that *Manual Intervention* does not only represent the capacity of the end user to intervene on the system but also the administrators of the system. Meanwhile, as a result of the focus on automation, the system may evolve in unexpected way which would make troubleshooting a lot more complicated. Thus, we argue that this causes for the use of adaptation approaches that are capable of supporting a user-in-the-loop model. This calls for a method or technique that can inform the users of the possible improvement but leaves the final decision of implementing the

improvement to user, and at the same time can be easily automated in case the user does not need to make the final decision.

Based on the discussion points brought about in the previous paragraphs, we've decided to explore the following research directions to tackle the shortcomings of current literature:

- Integrate improvement aspects into the design of the SS. This can be a way to push system engineers to think about improvement early on and define the data dimensions and schema they need to monitor in order to achieve said improvement.
- Use *recommendation* as a possible approach to implement adaptations. We argue that this the best way to ensure that the user stays in the loop and at the time not be bothered by each improvement that the system can make.

Part II

AS3: A method for building Adaptive Service-based Smart Systems

Chapter 4

Designing adaptive service-based smart systems: Approach overview

Software is getting slower more rapidly than hardware becomes faster

Niklaus Wirth

In part I of this manuscript, we have explored some of the most prominent approaches and technologies that are used to build Smart System (SS)s. The conducted literature analysis has led us to identify the issues and gaps that exist between the different aspects of SS. This allowed us to formulate some key insights on which to develop our contributions. In this chapter, we present the general approach we follow to tackle the identified gaps and issues. This approach is designed to take into account the various aspects discussed previously (i.e., *Perception*, *Response* and *Manual Intervention*) via the abstraction of prominent concepts. It also allows its integration and interoperability with other SSs. The approach is based on our view of SSs. This view is presented in a high level metamodel describing the main components of an SS. This metamodel would also serve later as the blueprint to the design of SSs and we argue that it provides a medium towards SS integration. The goal of this chapter is to present the rationale and the main concepts and constructs behind the development of the AS3 (Adaptive Service-based Smart System) method, which will be presented in detail in the next chapters. To that end, we start by defining the high level concepts that constitute an SS in general. Afterwards, we introduce the *Smart System Loop*, which represents our vision of an operational SS. It highlights the building blocks of any SS and their interactions, while providing a direct link to the capabilities discussed previously in the *PeRMI* framework and the high level definition of an

SS. Last but not least, we present a high level process model that constitutes the backbone of the AS3 method, in which the goal is to provide a way to design SSs via defining and building *Smart System Loop* instances.

4.1 Smart Systems: A model-based definition

Smart systems are complex as they run in highly connected and collaborative environments. They involve numerous entities at different levels of participation, each of which interacts with other entities to achieve one or several common business goals. However, from a high level point of abstraction, an SS is just like any other system. The International Council on Systems Engineering defines a system as "an arrangement of parts or elements that together exhibit behavior or meaning that the individual constituents do not."[1]. Based on this definition and on our own conception, we describe an SS as *"a set of elements which are either entities, associations or resources, where the entities represent the parts of the system, the associations represents their arrangement, and the resources are used by both entities and associations to achieve the goals of the system"*. Note, the entities of a system or a whole system could be associated with other systems. An SS also uses different resources to achieve its functionality and saves its state and the states of its elements in a log. We summarize our view of an SS in figure 4.1, illustrating a conceptual smart system's metamodel. This metamodel only presents high level concepts of an SS and will be refined as we dive deeper in the definition of the AS3 method later.

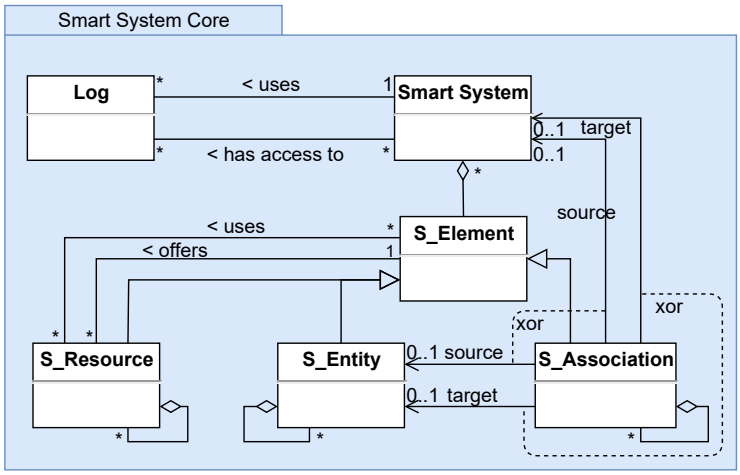


Figure 4.1 – Smart System Metamodel: A High level view of what constitutes a smart system.

[1] https://www.incose.org/about-systems-engineering/system-and-se-definition

4.1.1 Smart Entities

There are a lot of elements that constitute an SS. We call an *S_Entity*, any element of the SS that can exist independently from both the system and other entities. In an SS, we argue that these entities need to reflect the capabilities we discussed earlier in chapter 3.1 (i.e., perception, response, manual intervention). However, given the complex nature of SSs, the number of actual elements in a targeted application can be in the order of the tens if not the hundreds. This makes the process of designing, evaluating and improving the targeted systems really challenging. Adding a level of abstraction helps mitigate the related challenges and allows to reason about the entities of the SS in general as first class citizens.

This level of abstraction also allows to distinguish between the entities of the system and the eventual relationships that exist between these entities. Indeed, while an entity is the building block of any system, the same entity can be part of multiple systems at once. However, how that entity relates to the other entities of those systems is what makes the latter ones different. Note that an *S_Entity* has its own set of properties that form its state. An *S_Entity* also has a set of operations that allows it to communicate and expose its state to other entities of the system.

4.1.2 Smart Associations

Smart Associations (i.e., *S_Association*s) represent the different relationships that may exist between different entities of the smart system, or/and between the entities and other existing smart systems. These *Association*s define the nexus between the different capabilities that are supposed to be exhibited by any SS. In other words, the concept of *S_Association* encapsulates the logic of how the system perceives its environment, how it responds to particular events and situations and also, how these responses influence the system's perception of its environment. Given the strong connectedness of SSs, the type and number of associations that entities can have are endless. Moreover, these associations can be long or short standing in time, and they can evolve in unexpected ways. It is thus favorable to add a level of abstraction to model these types of relationships and allow engineers to think of these associations as moving parts in the targeted SS.

As we stated before, *S_Association*s are what makes SSs different from one another even if these systems involve the same entities. In figure 4.1, an *S_Association* defines the relationship between two smart entities, two smart systems or a smart entity and a smart system. The reason behind this choice is to allow a fast discovery and integration between systems and entities that are external to a particular

system. Given the different capabilities of SSs and since associations practically dictate how the transitions between these capabilities are performed, this level of abstraction also allows the abstraction of how these associations are implemented and what they entail.

4.1.3 Smart Resources

Software systems cannot function without virtual and hardware resources. This is especially true for SSs as they operate in a world where an abundance of computing resources are in place. An *S_Resource* is any type of computing resource that can be used by an *S_Entity* or an *S_Association* to accomplish their functionalities (e.g., Central Processing Unit, Graphical Processing Unit, Random Access Memory, etc.). In the presented model (see figure 4.1), we consider resources as elements that are independent from the business logic they implement and execute. This is mainly because resources are nowadays transient and can be provisioned and released quickly due to the advances in virtualization and containerization. This separation allows us to think about resources as first class citizens in any system. Hence, resources can be provisioned and released from particular entities or associations, allowing the possibility to optimize resource usage.

The existence of several types of computing resources, mounted on different types of devices and using different types of operating systems and platforms creates a heterogeneous landscape. Adding abstractions allows to classify computing resources in a manner that would allow engineers to concentrate on the functional and non-functional aspects of these resources, rather than particular implementations that are related to the devices that use these resources. It is important to keep in mind that most of the devices used today have scarce resources and thus present some resource constraints at the system level. This means that the resources on these devices are barely enough to enable its functions. However, while optimization through virtualization is not possible in these cases, the sheer number of these devices in smart environments allows optimization via activation.

4.1.4 Logs

Each system, during its execution, knows a lot of events that can be either internal to the system or while communicating and collaborating with external systems. The *Log* represents these different types of events. In the presented metamodel (see figure 4.1), this concept is directly linked to that of the *Smart System*. This is because the *Log* captures all the events that happen within the

system. Be it related to a change of the state of an *S_Entity*, an interaction between two entities, a change in the state of an association, a resource provision operation, etc.. This multitude of event types than can happen within the system is precisely the reason behind introducing the *Log* concept. Indeed, storing all the pertinent properties related to the different types of events occurring within the smart system proves to be extremely challenging given the complexity of smart systems.

The second relationship that links the *Smart System* concept to that of the *Log* is an access relationship. This means that while an SS maintains at least one log to document its execution events it also may access logs from other smart systems it collaborates with. The rationale behind this choice mainly relates to the possibilities of improvement for the system. Indeed, we argue that analyzing system logs that record the activities and events in collaborating systems can provide a great number of insights into how the target SS could improve. However, this entails a great deal of data consolidation to discern how the data in each system's log relates to the data in the target system's log. This is one reason behind the need for further specialization and definition of the types of entities, associations and improvement strategies, which constitutes the subject of the next sections and chapters.

4.2 The *Smart System Loop*

We have discussed in earlier chapters the different capabilities that need to be supported by any SS. We started by arguing that an SS needs to be able to construct a perception of itself within its environment to best perform its functions. Then, we stated that once a perception is formed, an SS is required to act based on that perception to address any functional requirements that are related to the targeted application domain. We called this a *Response* capability. Moreover, to allow these systems to evolve and adapt to their execution environments, we argued that the users of these systems need to be involved and they should maintain control over the SS. Hence, through the *Manual Intervention* capability, the users should be able to either change the behavior of the system manually by operating directly on the perception and response elements, or stay informed regarding the changes within the system if these changes were the result of self-adapting techniques.

To build effective and efficient SSs for smart environments, there is a need to bridge the gap between both the smart and service concepts, which can also be mapped to the concepts of sensing and responding in the *S-R* framework (Nechkoska, 2020). On one hand, there is an abundance of data about the *users*

that are accessible in smart environments. This richness comes mainly from the different sources of data (e.g. sensors, smart phones, smart objects, etc.) that are connected to the network, particularly the Internet and the constant interactions between the users and these sources of data. We refer to these data as *Context* data and adopt its definition from (Abowd et al., 1999) "*... any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*". To capitalize on this richness, a smart system should be capable of analyzing these data to detect particular *situations* within the environment and react accordingly (C. K. Chang et al., 2009). In this manuscript, context-awareness represents the techniques used to make the system self-aware (smart) and able to perceive its situation, as well as the situation of its users.

In addition, there are several concepts involved in typical SOC (Service Oriented Computing) approaches. In this manuscript, we cluster these under three main concepts, namely *Service*, *User* and *Goal*. *Service* usually performs a specific task and can be classified into Atomic Service and Composite Service (Casati and Shan, 2001). An Atomic Service is a single unit which can no longer be decomposed, while a composite service is the ordered, conditioned or parallel execution of two or more services. A *Goal* is a high level construct that describes the needs of an entity. They can be composed of smaller subgoals and tasks that need to be performed to achieve the higher goal (Rodrigues et al., 2019). In SOC, a goal is also called an abstract service (Fang et al., 2004) because it represents the functional aspect of a composite service, abstracting its implementation details. *Users* can be developers or domain experts who create the abstract services by specifying the tasks that need to be performed and the order in which they are performed later on by the services, or final users who consume the available services to accomplish particular tasks.

A smart system is adaptive by design. It needs this adaptability trait to face the changing landscape and requirements that characterize smart environments. In this thesis, we adopt a P/R (Perception - Response) paradigm (a.k.a. Sense - Response) to adaptability through the use of the adaptability loop (a.k.a. SIDA loop) (Haeckel, 1999). The SIDA (Sense - Interpret - Decide - Act) loop provides this adaptability through a continuous cycle of sensing, interpreting, deciding and acting on the information extracted from the internal and external environments. In our approach, we consider this cycle to be completely autonomous, as in the works of (H. Lee and J. Lee, 2018; Park et al., 2017). We further extend it using the concept of intervention to allow the users of the smart system to interact with the adaptability loop when the need to manually configure the system arises or

when the system needs specific information to function correctly. Our choice for adopting and extending the SIDA loop is because, while other popular and efficient control loops, such as the MAPE-K loop (Kephart and Chess, 2003b), comes from an autonomic computing background, SIDA loop comes from an organizational background and takes by default the human and social dimensions into account. This in turn made the extension with human type users more logical and intuitive.

From the concepts and abstractions discussed above, it can be noticed that there is an alignment between the adaptability, context and service perspectives. We represent this alignment in what we call the *Smart System Loop* illustrated in figure 4.2 containing the building concepts of service-based smart systems and their interactions. The *Smart System Loop* is composed of two main constructs. First, the outside SIDA loop and its corresponding building blocks (i.e., the *context*, *situation*, *goal*, *service*) represent the autonomous part of the system (i.e., what the system is programmed to do automatically). Second, the *user*'s relationships with the other building concepts represent the manual interaction between the user and the autonomous part of the system. Thus, we consider the *user*, *context*, *situation*, *goal* and *service* as the building concepts for every service-based SS. As a whole, the *Smart System Loop* represents our view of an SS from an operational perspective as it supports the different capabilities we have discussed in the *PeRMI* framework in the previous chapter. A detailed explanation on how the *Smart System Loop* supports those capabilities and the rationale behind the use of the corresponding building concepts is provided in the following sections.

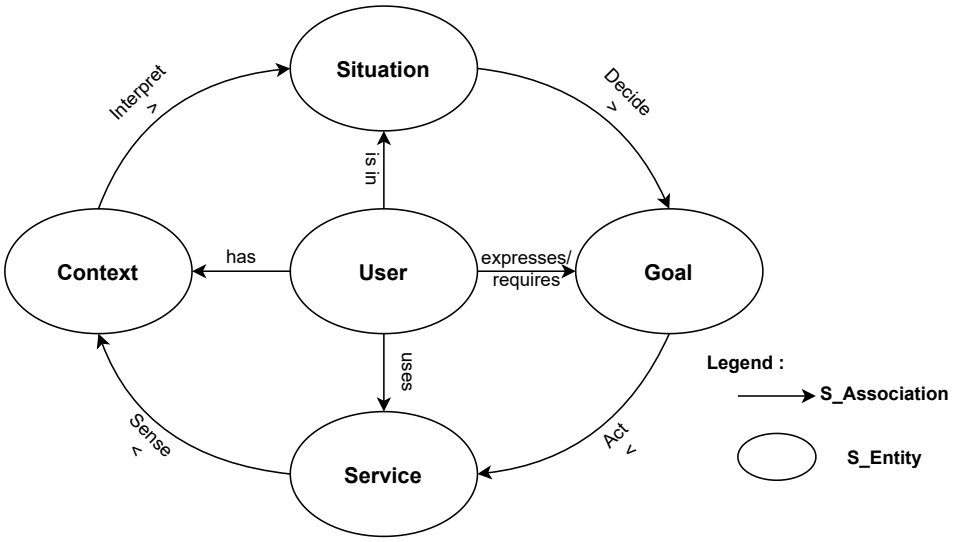


Figure 4.2 – *Smart System Loop* representing the building concepts and their interactions.

4.2.1 Perception elements in the *Smart System Loop*

The *Smart System Loop* illustrated in figure 4.2 puts forth three *Smart Entities* to represent the *Perception* capability. First, the *User* is an entity structure that represents the different human users and system users that are involved in and/or interact with the targeted SS. For example, in the *SMARTROAD* scenario presented in section 1.4, a human user can be a *driver*, a *passenger* or a *pedestrian*. The *User* entity is central in our approach to designing and building SSs. Indeed, it constitutes the moving part of every system as the state and behavior of each user continue to change over time.

To monitor the state and behavior of each user, the *Context* entity comes into play as a structure that can hold data and metadata related to contextual dimensions relevant to the users of the targeted system. Context data can be generated by the human users through their interactions with the system or through system users (i.e., services) by using a messaging protocol that is understandable by the system and these services.

On their own, these context data are still considered raw and remain of little value to the targeted system. This is mainly because, although they can be of varying informational value, they still represent relatively fine-grained user properties. Hence, to make these context data into something that the system can understand and respond to, the *Situation* entity is proposed. The *Situation* entity comes as a construct to bridge the gap between the level of granularity of context data and the level of granularity required by the system to recognize a need for action. It is a mechanism that allows the conversion of fine-grained context data to higher entities that can be easily stored and recognized by the system. Hence, the *Situation* entity provides a transition point between the perception elements and the response elements of the *Smart System Loop*. Figure 4.3 summarizes the relationships between these entities.

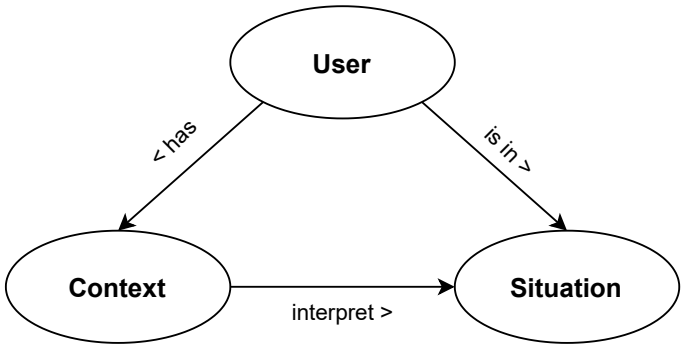


Figure 4.3 – The *Smart System Loop*'s perception concepts and their relationships.

4.2.2 Response elements in the *Smart System Loop*

The response elements present in the *Smart System Loop* come from a service view to allow for action to be taken smoothly within the system. The *Smart System Loop* allows to particularly focus on three entities within this view.

- First, the *User* entity is also present as a response element. This is because the *User* as an entity encapsulates both human users and system (virtual) users which are considered as both service providers and service consumers in the service view. Indeed, while we try to develop adequate and pertinent services for consumers, we also rely on resources in the form of other services to do so. However, when discussing the response elements of the *Smart System Loop*, we mainly focus on the role of the consumer as the consumer constitutes the entity that the system needs to satisfy.

- Second, the *Goal* entity allows to formulate what needs to be done to effectively respond to a risen situation. It constitutes the entry point to the response elements and the connection to perception elements through its relationship with the *Situation* entity. Goals are abstract concepts that encompass the actions to be taken by the system. They also provide a natural link to the services to be invoked and executed in order to respond to a particular situation as they also can be seen as abstract services.

- Last but not least, the *Service* entity represents an umbrella under which both functional and non-functional aspects of services can described. They embody the concrete way in which the actions are carried out in order to achieve the defined goals. As the only way to carry out concrete tasks, the execution of a service can have temporary or lasting effects on the environment of an SS. These changes are captured again as context data, potentially prompting a new cycle over the *Smart System Loop*. Hence, the *Service* entity also provides a way to perceive the changes in the environment around the system's users.

The response elements discussed above and the relationships between them are illustrated in figure 4.4.

4.2.3 Manual Intervention elements in The *Smart System Loop*

The perception and the response elements discussed previously constitute the autonomic part of the *Smart System Loop*. They illustrate how an SS autonomously oscillates between the phases of perception and response through identifying

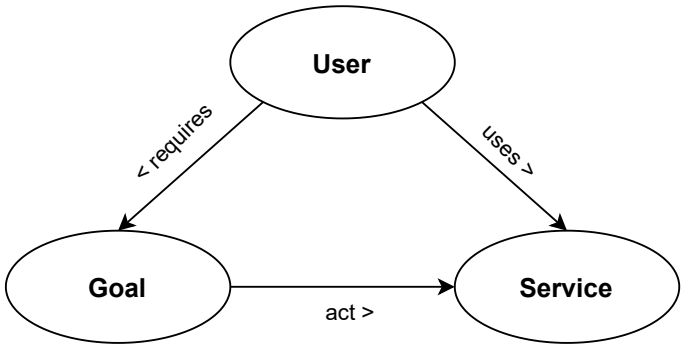


Figure 4.4 – The *Smart System Loop*'s response concepts and their relationships.

pertinent situations and services. However, the autonomous part of the *Smart System Loop* is isolated from the inputs of the user. This can cause the SS to start behaving differently from what is intended by the system engineers and what is hoped for by the users.

To address this issue, we introduce the *Manual Intervention* capability allowing the *User* to interact with the system in two different roles.

- First, as a final user of the SS, the *User* should be able to correct any system behavior that doesn't suit his needs and preferences. This inadequate system behavior can be the product of the system's learning algorithms that fuels it autonomous part and can be due to missing data or bad performance in final user's context. Expressing the SS's functions in the terms of a *Smart System Loop* allows to abstract the intricate technical details surrounding the learning algorithms and models mentioned above. It allows the final user trace the source of the inadequate behavior just by following the transitions between the different entities and eventually address the problem by acting directly on each entity (e.g., modifying the context properties, the goals or the transition rules).

- Second, as an architect of the SS, the *User* should be able to monitor, adapt and optimize the system's behavior and performance. Indeed, as the autonomous part of the system runs, it continues to learn and adapt to its environment. This can cause the system's architect to lose "control" of the inner-working of the system, which would make troubleshooting, evaluations and updates hard to perform. In this context, the goal of the *Manual Intervention* paradigm is to allow the architect of the system to stay informed and/or approve the policies that the autonomous part of the system learns before they are woven into the system. The possible interactions between the architect as a *User* and the other elements of the *Smart System Loop* allows

him to alter the way algorithms can detect situation by, for example, adding new context dimensions or adding more goals or services.

The *Manual Intervention* elements discussed above as the interactive part of the SS and the relationships between them are illustrated in figure 4.5.

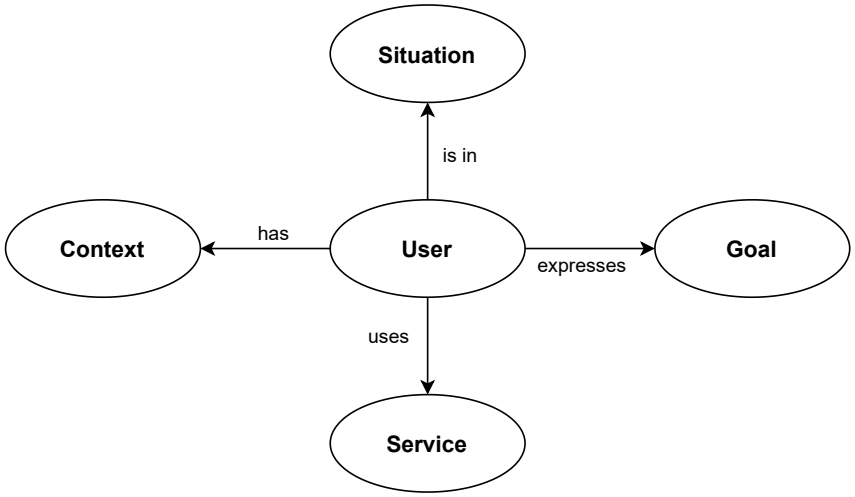


Figure 4.5 – The *Smart System Loop*'s manual intervention concepts and their relationships.

4.3 High level view of our approach to design Adaptive Service-based Smart Systems

As we've seen in our analysis of the state of the art, SSs have been addressed from different perspectives and disciplines. This has led to the accumulation of a big knowledge body surrounding these systems. However, a systematic method that can leverage this knowledge body is of great importance but still lacking. This section presents a high level view of the AS3 method that we're proposing to fill this gap and allow for a generic way to design and develop SSs. The method presents a systematic way to capture, identify and describe the elements involved in a targeted SS as well as the relationships between these elements. Indeed, knowing what contextual elements change the situation of which users, and then knowing what goals need to be set to respond to these arisen situations and the services that need to be invoked or implemented to satisfy these goals, form the the basis of our approach to SSs.

Before delving a little deeper into the AS3 method, a definition of what constitutes a method in our particular context is necessary. According to (Grady et al., 2007), "A method is a disciplined procedure for generating a set of models that describe various aspects of a software system under development, using some

well-defined notation.". According to the OMG modeling levels (OMG, 2016), a method is composed of a product meta-model (level M2) that defines the abstract syntax of the modeling language and a process model (level M1) whose execution (i.e., instantiation) produces product models that conform to the product meta-model (Cela et al., 2019). Thus, in this sense, the AS3 method provides a product metamodel containing the concepts allowing the design and development of adaptive context-aware and service-based SSs, and a process model that defines the goals to reach through the use of the defined concepts.

In this context, an intentional model was used to express the contents of the process model. Specifically, we adopt the MAP formalism from (Rolland, 2007) to define the intentional process model of the AS3 method. MAP is a flexible representation system that allows the description of a process model using the notion of intentions, and strategies to navigate between the different intentions. MAP's intentions are represented as graph nodes while the strategies are directed edges. Note, the same two intentions could be linked to more than one strategy, which makes the MAP a directed multigraph. A MAP can be decomposed into sections, with each section defined as a 3-tuple $< I_i, I_j, S_{ij} >$, where I_i is the source intention node, I_j is the destination intention node and S_{ij} is a strategy allowing the achievement of I_j stating from I_i.

At a high-level abstraction, the AS3 method focuses on the identification, definition and improvement of the different elements forming the SS. Figure 4.6 presents the MAP (i.e., process model) formalizing this high level view of AS3. The process model has two intentions at the top level. Each intention represents a goal that the method user (i.e., system engineer or architect) needs to accomplish. First, *Define the elements of the SS* intention represents the goal of identifying all the elements of the SS that make up the *Smart System Loop*. Second, *Discover improvements for the SS* intention represents the goal of getting possible ways and measures to improve the current version of the SS from an operational perspective.

The process starts by defining the elements of the SS from a description of the system's requirements. Once the elements have been defined, the system engineer can proceed in three different manners. First, he/she can explore other existing systems for possible elements he/she has missed by means of the identified elements. It implies exploring the existing relationships between the already identified elements of the targeted SS and the elements of the existing SSs that may collaborate or integrate with the targeted SS. This is translated by the <*Define the elements of the SS*, *Define the elements of the SS*, *by exploration*> section and allows him/her to complete the functional model of the system. Second, through

the <*Define the elements of the SS*, *Stop*, *by choice*> section, he/she can move to stop the process once he/she decides that the SS is working at optimal level and cannot improve any further. Note that the intention to *stop* in this process model does not mean stopping the system but only that the user is satisfied with the current version of the system. Third, he/she can trigger an analysis of the traces of execution left by the system to discover possible improvements to the targeted system. This is supported by the <*Define the elements of the SS*, *Discover improvements for the SS*, *by log analysis*> section and allows him/her to add, modify or remove the elements of the system to improve or optimize its performance. This last section involves using algorithmic structures on the execution traces to extract the possible improvement. Note that the execution traces in this case can be those generated by the targeted SS itself or those that the targeted SS is able to access as described in the smart system metamodel represented in figure 4.1.

Once the discovery of possible improvements to the SS is achieved, the method user can proceed in one of three ways. First, if some possible improvements have been discovered, he/she can proceed to performing the discovered improvements on the current version of the SS. This means that he can start adding, modifying and/or deleting elements of the SS depending on the discovered improvements; this procedure is supported by the <*Discover improvements for the SS*, *Define the elements of the SS*, *by enactment*> section. Second, he/she can choose to end the improvement discovery process which is translated by the <*Discover improvements for the SS*, *Stop*, *by choice*> section. Note that the intention to *stop* in this process model does not mean stopping the system but only the continuous improvement cycle of the system at the design level. Third, he/she can proceed to a comparative learning process tracking the improvements made to the system to discover the best configuration for the system and/or predict and detect emergent system behavior, supported by the <*Discover improvements for the SS*, *Discover improvements for the SS*, *by learning*> section. These sections will be further detailed in the next chapter.

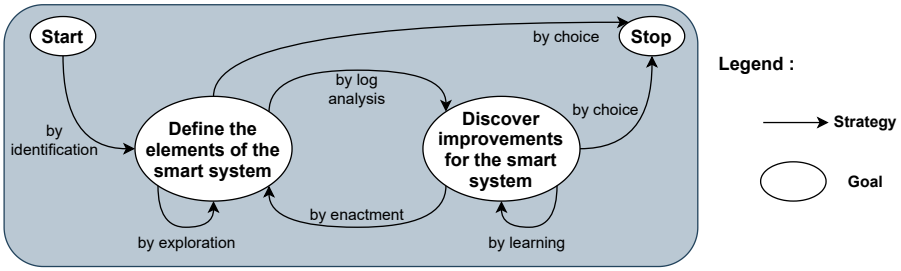


Figure 4.6 – A high level view of the intentional process model of the AS3 method.

4.4 Summary

In this chapter, the focus was on defining SSs in a way that would allow their adoption at a wide scale. We argue that this adoption wouldn't be possible if the elements constituting an SS are not defined separately from its application domain. To that end, we propose a metamodel that defines the concepts in play in the design of an SS. Namely, the *Smart Entity*, *Smart Association*, *Smart Resource* and *Log* concepts were introduced and defined. These concepts and the relationships thereof are then defined separately to allow a clear separation of the concepts and how each of these concepts contributes to the overall SS.

The defined concepts are then used to propose the *Smart System Loop* that we argue would help system engineers to envision the inner working of their targeted SS. The goal was to provide a link between the concepts of the metamodel, the capabilities of the SS and actual design choices that the engineers are brought to make. The *Smart System Loop* adopts an adaptive loop structure to express the adaptive trait of SSs. It can be seen as an instantiation of the SS's metamodel focusing on its functional capabilities. The *Context* and the *Situation* concepts as special *Smart Entities*, are responsible for representing the *Perception* elements, while the *Goal* and the *Service* concepts are responsible for representing the *Response* elements. The *User* is then introduced to represent both the final-user and architect of the SS depending on the nature of the operations getting carried out. Indeed, while the intervention of the final-user is recorded to personalize and customize the perception and the response capabilities of the system for that specific final-user, the intervention of the architect are used to improve the performance of the system at a global stage.

To allow system architects to think about systems as continuously improvable, we defined AS3's process model that would help track the intention that they should consider while defining their targeted SS. The proposed process model draws its expressiveness power from the MAP formalism and allows for different levels of abstraction and refinement. The process model works in harmony with the concepts defined in the metamodel of the SS to design and build the targeted SS. Indeed, each *intention* on the process model is achieved through at least one strategy that requires the instantiation of one or multiple concepts with regard to the application domain of the targeted SS. At the highest level of abstraction, the process model can be considered as a cycle of definition-improvement intentions. The process highlights the continuous improvement characteristic which is inherent to SSs.

4.5 Discussion

The main goal behind the artifacts proposed in this chapter is to push the system engineers to think about SSs, not as a new paradigm, but as the natural evolution of several paradigms and technologies. By doing so, we argue that system engineers would be less confused about what an SS is and what it entails in terms of design and development. In this chapter, the discussed artifacts remain at a high abstraction level as they depict general concepts to guide the line of thought of the system engineer toward the design of a target SS. Details regarding the implementation and instantiation of each concept will be discussed in later chapters.

The Smart System Metamodel presented in this chapter (see figure 4.1) has the advantage of being extensible, understandable and easily translated to a more formal definition. Adding to its abstraction level, this gives it a great level of expressiveness towards defining the different elements that the system engineer conceives as necessary to the development of the targeted SS. However, the downside with such a metamodel lies in the difficulty of identifying the mappings between the targeted application concepts with the metamodel's concepts. Indeed, metamodels can be too permissive without clear guidelines describing each concept and the real world concepts that can be mapped to it.

The process model described in section 4.3 was developed to cater to the issue discussed above. The goal is to frame the discussion surrounding the design and the development of SSs around the concepts involved in the targeted SS and its continuous improvement. Hence, every concept that can be extracted from the application domain needs to be tracked to the intentions of the process model. Indeed, the process model and the metamodel should always be read hand-in-hand, allowing the method user to identify the concepts to instantiate for each intention set. Figure 4.7 illustrates this way of reading the process model and the model.

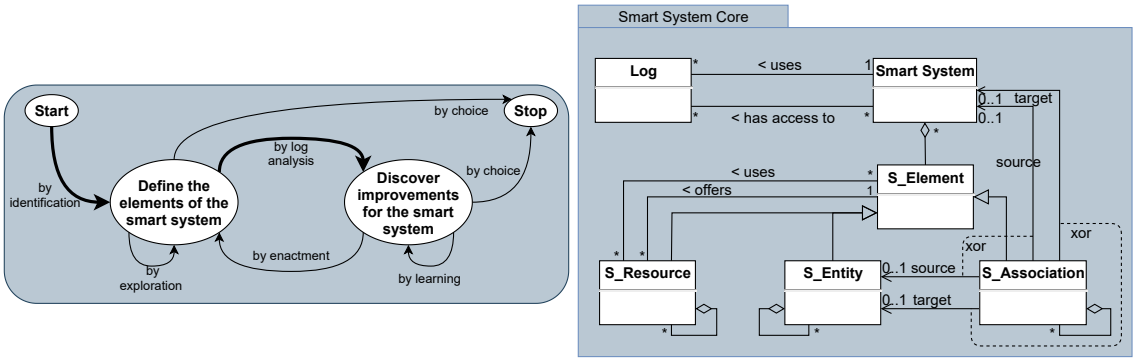


Figure 4.7 – A high level view of the AS3 method.

Moreover, an important feature that is exposed in the process model is the ability to integrate concepts in existing SSs that can potentially communicate with the targeted SS. We argue that this feature lays the ground for an easy integration towards achieving a System of Systems (SoS) paradigm. The reason we present this feature at this high level view of the method is that there is no consensus regarding how SSs are described and defined by practitioners. Hence, only the idea is put forward at this level to frame the discussion on more concrete techniques to make such a feature possible.

Having presented a high level view of the AS3 method, an important link that was missing is the connection between the method's concepts and the capabilities of SSs presented in the *PeRMI* framework (see section 3.1). The *Smart System Loop* was developed to close that gap. Indeed, we argue that this loop adds an explicit relationship between the concepts introduced in the *PeRMI* framework and the method's concept through the introduction of the *User*, the *Context*, the *Situation*, the *Goal* and the *Service* as special concepts of the method's *Smart Entity* concept. The full product metamodel supporting the AS3 method will be presented and described in the next chapter.

The downside to this loop is that we somewhat limit the design decisions that the engineers need to make in order to design and develop their targeted SS by specifying particular design concepts. However, because of the abstract nature of the method, system engineers can substitute the proposed special concepts with any concepts they're familiar with on the condition that the adaptability and the feedback structure of the loop are still guaranteed. Indeed, providing the link between the method's concepts and the SS's capabilities is not the only objective behind the *Smart System Loop*. This latter was also developed to allow the system engineer to reflect, early on in the design and development process, on the possible and potential improvements that the SS may undergo.

In the next chapter of this thesis, we'll delve deeper into the AS3 method as we explain the different goals and strategies that the method provides to design SSs. Starting from the high level view of the AS3 method presented in figure 4.7, we will provide the rationale behind the development of the different intentions as we refine the sections of the process model and the supporting concepts of the product metamodel.

To do so, we introduce what we consider an exemplary path to achieve the general purpose to *Implement efficient and self-improving service-based smart system*, which is the intention of the root map presented in figure 4.6. We focus particularly on two sections of the process model which are (1) <*Start*, *Define the elements of the smart system*, *by identification*> and (2) <*Define the elements of the*

smart system, Discover improvement for the smart system, by log analysis>. These sections are refined in figure 4.8. The choice to **refine** these particular sections is supported by the fact that they constitute the cornerstones of the approach.

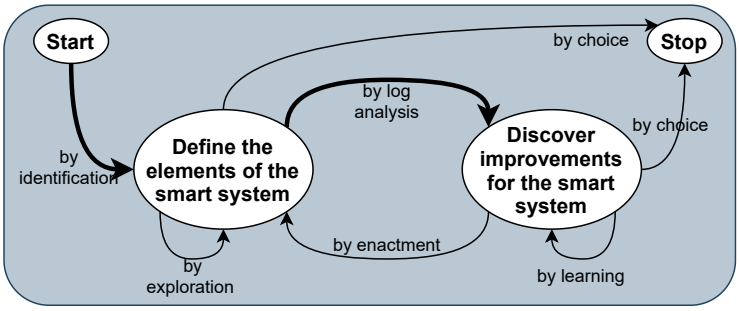


Figure 4.8 – AS3's Happy Path.

Chapter 5

Defining the elements of adaptive service-based Smart Systems: an intentional approach

Good methods can teach us to develop and use to better purpose the faculties with which nature has endowed us, while poor methods may prevent us from turning them to good account. Thus the genius of inventiveness, so precious in the sciences, may be diminished or even smothered by a poor method, while a good method may increase and develop it

Claude Bernard

The development of Smart System (SS)s requires the definition of the conceptual and physical components that make up their fabric. Although SSs are dependent on the application domain, we argue that formalizing a way to design these systems is beneficial within the perspective of their integration and interoperability. Inspired by the *Smart System Loop*, we attempt to reconcile between the different involved perspectives as we refine each section of the method's high level process model and its corresponding product metamodel in an intentional approach. In this chapter, we build upon our analysis of the literature regarding the different paradigms and technologies involved in making SS to refine the high level view of the AS3 method presented in the previous chapter (see figure 4.7). Specifically, we focus on extracting the concepts that are common to the design of SSs as we propose intentions and strategies on how to use the these concepts to *define the elements of targeted SSs*. To illustrate the use of the AS3 method in defining the elements of a targeted SS, we present a rundown of its use on the *SMARTROAD* system (see section 1.4)

5.1 Defining the elements of the smart system by identification

As mentioned above, this MAP section has the intention to *Define the elements of the smart system* starting from the system's requirements (<Start, Define the elements of the smart system, by identification>). In this thesis, we introduce three types of elements that constitute a smart system namely, **S_Entity**, **S_Association** and **S_Resource**. In other words, the goal is to have one or several executable smart system loops. To achieve this intention, the method user needs to identify these elements by instantiating the corresponding concepts. The MAP representing the intentions and strategies involved in this section are depicted in figure 5.1. This section of the MAP mainly consists of three intentions, (1) *Define the entities*, (2) *Define the associations* and (3) *Assign the resources*. In the following, each of these intentions will be covered in detail using at least one section of the process model. Because the process model is flexible, we will focus more on the sections that are usually targeted in the beginning of its execution and explain how other sections relate to them. Specifically, the focus will be on the sections that are shown with bold edges in figure 5.1.

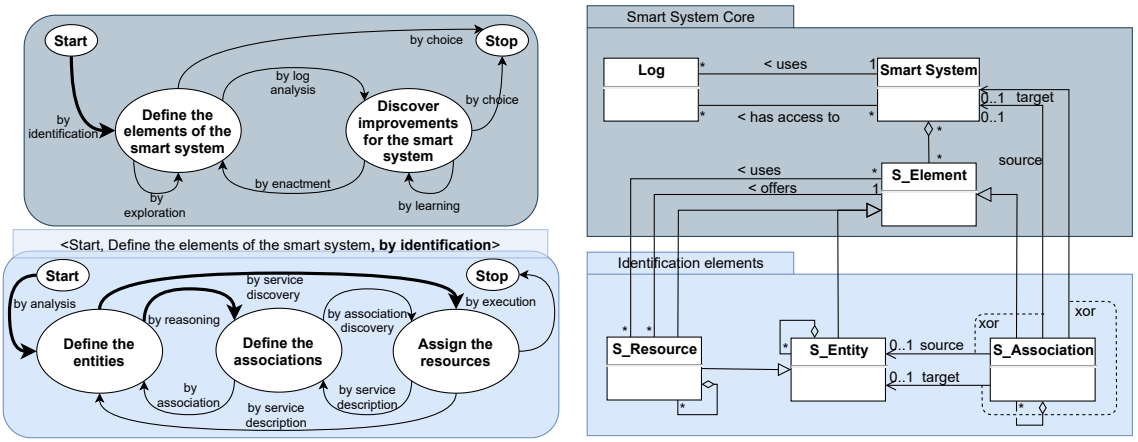


Figure 5.1 – Defining the elements of the smart system from the start by identification

Identifying and defining the entities means having a functional model of *what the system should be able to do*. This entails at a finer abstraction level, the instantiation of the concepts (i.e., entities) introduced in the smart system loop (see figure 4.2). Instantiating these concepts allows the coverage of what the system can perceive **(P)**, respond to **(R)** as well as identify where the need for manual interventions **(MI)** arises. Indeed, the system perceives its environment using the instances of **Situation** and **Context** concepts and the relationships between the identified instances. Then, it is capable of responding to special situations through the instances of **Goal** and **Service** concepts and the relationships between the identified instances as well as the special relationship between the situations

and the goals. The capability of a user to manually intervene in the system's configuration is possible through the instances of the **User** concept and their ability to interface with the other concept instances.

To ensure smooth transitions (i.e., flow of information) between the defined entities, a set of concepts representing the associations between the entities is introduced. These concepts come to operationalize the links between the different entities in the smart system loop (i.e., User, Context, Situation, Goal and Service). Hence, they specify how each concept instance pours into the other. For instance, one or several **Context** instances need to be interpreted to make the detection of a situation instance possible. To operationalize this link, the **Function** concept is introduced allowing a mapping between the context instance and the situation instance. More formally, a **Function** instance f_i is represented using equation (5.1), where F is the set of defined functions, C is the set of defined context instances and St is the set of defined situations.

$$f_i \in F : C_x \rightarrow St_x, \textit{ where } C_x \subseteq C \textit{ and } St_x \in St \tag{5.1}$$

Also, we consider that each **S_Entity** and **S_Association** may consume physical computational resources available to the system (i.e., owned by the stakeholders of the system or third parties) to accomplish their functional tasks and that these resources are instances of the **S_Resource** concept. These computational resources are described, exposed and handled by the provided business services. For example, to be able to capture and send the speed of a vehicle as a context instance through a video feed, a service would need to exploit an **Input** resource (i.e., video camera), a **Computing** resource (i.e., running the speed detection algorithm) and a **Network** resource (i.e., wireless network) to send the information to interested parties or make it accessible to them.

To create the previously mentioned instances, a set of guidelines are provided to the method user through a set of process models and their supporting product metamodel. Figure 5.2 presents a summary of these guidelines. In each step of the process model, the method user is brought to instantiate the corresponding concepts of the product metamodel and identify the associations between the instances. In the following, we present what we consider to be the most important parts of this process to guide the method user in defining the elements of the smart system and the concepts involved at each step.

5.1.1 Defining the entities by analysis

This section introduces the key concepts and goals involved in achieving the intention to *Define the entities* of the targeted smart system (<Start, Define the

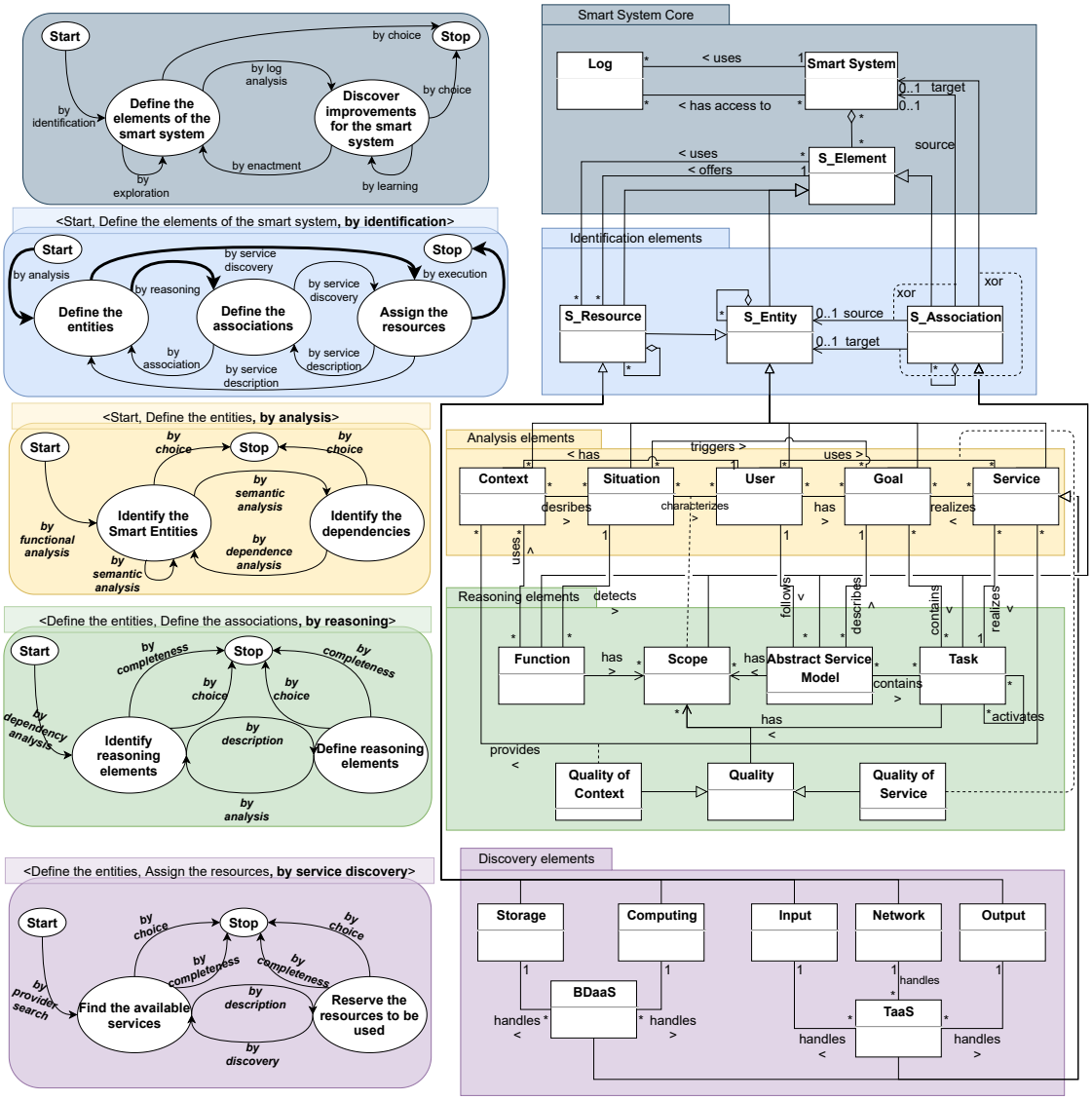


Figure 5.2 – A detailed view of the <Start, Define the elements of the smart system, by identification> section: Process model and its supporting product metamodel

entities, by analysis>). Figure 5.2 shows these intentions and concepts in yellow colored frames through a process model and its supporting product metamodel. Achieving this intention leads to having a model of the entities involved in the system with respect to the smart system loop introduced earlier in the thesis. This process starts by analyzing the requirements of the system as described in the requirement elicitation stage and identifying the entities making the system. These entities can be classified as instances of the **User**, **Context**, **Situation**, **Goal** or **Service** concepts. Identifying the dependencies between the identified entities by semantically and functionally analyzing them allows to construct a more detailed model of the targeted system allowing both a finer understanding of the targeted system and the possibility to discover new and unidentified entities.

5.1.1.1 Identifying the users.

Being the center of smart environments, we consider the identification of the users as a first and crucial step in the design of not only smart systems but any system. This is especially crucial and also more complicated in smart environments, as the users are not only of a human nature but can also be other autonomous systems (e.g., vehicles). Since we take a service-based approach to smart systems, we consider a user any participant in the targeted environment who is using or offering a service in the context of the targeted system. The identification of the users of a system in a particular domain involves answering the following questions:

- Who is going to use the system ? This allows to extract and identify a primary batch of the system's users, which will be later on enriched by the identification of the dependencies. This is usually captured using scenarios and/or use case diagrams in the system's requirement elicitation.
- Can a set of these users be clustered in a single group ? Clustering some of the identified users allows a level of abstraction from the peculiarities of each user, thus providing a uniform way to address situations where a group of users has to be involved.
- What are the relationships between the users ? Identifying the relations between the different users and/or group of users allows a better understanding of how the system perceives its environment and reacts to it based on the situations of each user and the impacts on other users.

Application to *SMARTROAD* To identify the users of the *SMARTROAD* system, we first analyze the case study to answer the questions indicated above. Then, we summarize the answers to these mentioned questions in the class diagram presented in figure 5.3.

Who are the entities that are going to use the system ? Since the first priority in road security is the preservation of human lives, obvious users of the *SMARTROAD* system are the *drivers* and the *passengers*. The list of users can be enriched using the stakeholders in the transport domain from table 1.1 (e.g., *Emergency Services*, *Road Operators*, *Automobile Industry*, etc.).

Can a set of users be clustered in a single group ? From the list of users extracted as an answer to the previous question, we can notice that those users influence the road transport ecosystem mainly at four levels, being: (1) person, (2) vehicle, (3) infrastructure and (4) environment. Hence, we cluster the users according to these levels. This classification is also compatible with the classification of risk factors from a road security standpoint (Force, 2014).

What are the relationships between the users ? The relationships between the different groups of users discussed in the previous answer can be modeled as aggregations. Since the user (i.e., person), through his vehicle, is using infrastructures that are implemented in an environment, we can consider the environment as a set of infrastructures where vehicles (consequently drivers and passengers) are in a state of mobility.

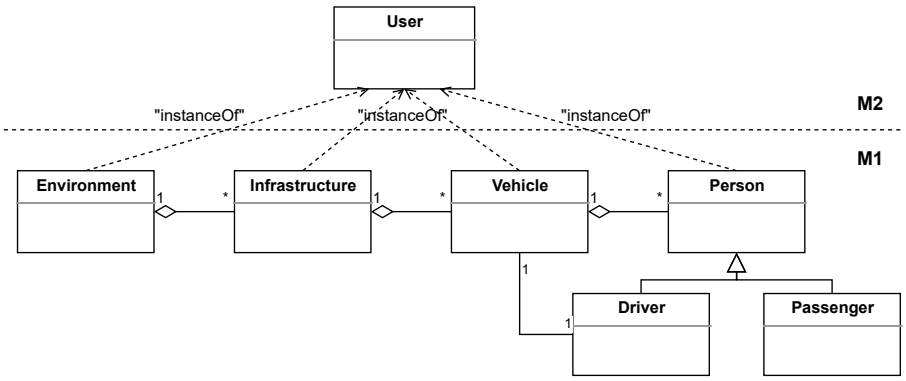


Figure 5.3 – The identified users of the *SMARTROAD* system

5.1.1.2 Identifying the situations

Being able to detect and respond to possible situations is the basis of AS3 to build smart solutions. Hence, it is important to identify, early on in the system development process, the situations of the users that the target solution is supposed to deal with. A *Situation* is defined in the Cambridge online dictionary[1] as "the set of things that are happening and the conditions that exist at a particular time and place". This definition alludes to four questions that need to be answered when identifying a situation which are the *what?*, *why?*, *when?* and *where?* being the things that are happening, the conditions, the time and the place of the situation respectively. Another question to be answered is the *who?* which can be directly mapped to an entity or a user of the targeted application domain.

Application to *SMARTROAD* In the *SMARTROAD* system, we are interested in the situations that represent a potential risk to the different users. Given the dynamic nature of the domain, the answers to *when?*, *where?* and *who?* questions are always directly mapped to the current time and place. Hence, we present in table 5.1, 4 examples of these situations while answering the remaining questions raised in section 5.1.1 (i.e., what and why).

[1] https://dictionary.cambridge.org/fr/dictionnaire/anglais/situation

Table 5.1 – Examples of the situations in the *SMARTROAD* system

User	*Situation*	*Question*	*Answer*
Person	Risky Health	What ?	The Person is not feeling well
		Why ?	The value(s) of monitored vital(s) is(are) abnormal
Vehicle	Risky Vehicle	What ?	The Vehicle is in an unreliable state
		Why ?	Some of the Vehicle's parts are bugging, faulty, deactivated or broken
Infrastructure	Risky Road	What ?	The road is risky to drive on
		Why ?	The road has cavities or missing signs
Environment	Traffic Congestion	What ?	A number of roads are blocked or flowing slowly
		Why ?	There is an accident on the road or it's rush hour

5.1.1.3 Identifying the context

In this thesis, we also refer to context instances as 'Context Dimensions'. These context dimensions represent the different data that can be accessed or generated by the smart system. We logically classify *Context Dimensions* into two types. *Primary Context Dimensions* are the necessary data or information that is required for the execution of the system and is used to infer the possible situations of the users. *Secondary Context Dimensions* are the data that are used to optimize or improve the operations of the solution's components or third party services. This is just a logical separation as the same *Context Dimension* can be considered as both *Primary* and *Secondary*.

Application to *SMARTROAD* As we stated before, context dimensions provide the data and metadata that are required to detect or infer the situations and also can serve as data to personalize or improve the performance of a task or as input data to execute a service. In the *SMARTROAD* system, we suppose that we have access to the context dimensions through services that are provided by smart objects. We identify the *Primary Context Dimensions* through the answers provided for the *Why?* question in the definitions of the situation. Table 5.2 presents the *Primary Context Dimensions* of the *SMARTROAD* system.

5.1.1.4 Identifying the goals

The goals represent the actions that must be performed to satisfy specific needs. These needs are dictated by the detected situations of the users and are specified by domain experts. The defined goals encompass the response elements for each collaborator or stakeholder in the detected situation. The elements to describe these goals are provided by answering the following question. *What is the system needed to do to successfully respond to a detected situation ?*

Table 5.2 – Examples of Primary Context Dimensions in the *SMARTROAD* system

Situation	*User*	*Context*	*Type*	*Access*
Risky Health	Person	Blood Pressure	Sensed	e.g., Smart Watch
		Body Temperature	Sensed	e.g., Smart Watch
		Pulse	Sensed	e.g., Smart Watch
Risky Vehicle	Vehicle	Brakes Status	Sensed	e.g., Brake Sensor
		Tire Pressure	Sensed	e.g., Tire Pressure Monitoring System
		Engine Status	Sensed	e.g., Mass Air Flow Sensor
Risky Road	Infrastructure	Cavities	User Defined	e.g., Manual User Entry
		Missing signs	User Defined	e.g., Manual User Entry
Risky Traffic	Environment	Accident	User Defined	e.g., Manual User Input
			Derived	e.g., Video Camera Feed
		Rush Hour	Derived	e.g., Traffic History

Application to *SMARTROAD* Identifying the actions that need to be performed in order to respond to a particular situation is the responsibility of the domain experts. The example shown in table 5.3 presents the high level goals of the different users of the *SMARTROAD* system in the case of a *Risky Health* situation.

Table 5.3 – Examples of goals for different users in the case of a *Risky Health* situation

Situation	*User*	*Goal*
Risky Health	Infrastructure	To adapt the speed of the vehicles in the infrastructure
	Environment	To choose a route for the person exhibiting the symptoms
	Environment	To inform a nearby health-care facility
	Vehicle	To inform nearby vehicles
	Driver	To get a route to the nearest health-care facility

5.1.1.5 Identifying the services

A service is a unit through which a specific task can be performed. This unit can consume data to perform the required task and/or produce data after the task is performed. In this thesis, we assume that several concrete services are capable of satisfying the same task which represents the functional part of the service. However, there also exists non-functional aspects of the services that relay the information about the quality of the services. Examples of the non-functional aspects of a service are numerous (e.g., response time, throughput, availability, performance, trust, security, price, etc.) and can be objective or subjective.

Application to *SMARTROAD* In this thesis, we suppose that each identified task can be executed by multiple services but with different QoS (Quality of Service) parameters. For example, getting the speed of vehicles can be done through a

radar or a video processing algorithm on a video feed. We refer the reader to table 5.4 for example available services in the *SMARTROAD* system.

Acronym	Service Name	Stakeholder
AEVW	Approaching Emergency Vehicle Warning	Emergency Services
CBW	Car Breakdown Warning	Automobile Industry
RMS	Road Monitoring Service	Road Operators
VSMS	Vehicle State Monitoring Service	Automobile Industry
ISA	Intelligent Speed Adaptation	Automobile Industry
IVS	In-Vehicle Signage	Automobile Industry
TJAW	Traffic Jam Ahead Warning	Road Operators
ERP	Electronic Road Panel Service	Road Operators
RWW	Road Works Warning	Road Operators
WWS	Weather Warning Service	Weather Services
SMSD	Short Messaging Service for Driver	Automobile Industry
RSS	Route Selection Service	Automobile Industry
ERS	Emergency Rescue Service	Emergency Services
NAAS	Nearby Area Alarming Service	Road Operators
TSS	Towing Selection Service	Navigation Services
OCS	Oil Calculation Service	Automobile Industry
GSS	Garage Selection Service	Navigation Services
GSSS	Gas Station Selection Service	Navigation Services
HSS	Hospital Selection Service	Health-care Services

Table 5.4 – Examples of provided services

Going back to the example scenario in section 1.4, the identified services are used to achieve the general goal of the response. In the case of the $riskySpeedResponse$, these services are *Intelligent Speed Adaptation* to compute the appropriate speed for the vehicles in the road, *Dangerous Nearby Area* to compute the safe distance between the cars in the road and *In-Vehicle Signage* to inform the drivers about the situation. Note, since services can also be used to expose resources like sensors and actuators, some services are capable of providing data about the identified context dimensions as is the case in $vehicleSpeed$ using the *Car*'s local speedometer or a *Road* radar.

5.1.2 Defining the associations by reasoning

In this section, we present the concepts that allow the method users to materialize the relations between the instances of the building blocks that were identified after achieving the intention *Define the entities* (<Define the entities, Define the associations, by reasoning>). Figure 5.2 presents in green the general strategies and intentions involved in this section, supported by the product metamodel concepts. The intentions of this section are twofold. First, to *Identify the reasoning elements*, the method user needs to instantiate the concepts allowing to operationalize the relationships between the identified entities by analyzing

the dependencies between the analysis elements and between the reasoning. Second, to *Define reasoning elements*, the method user needs to describe the identified reasoning elements with respect to the scope of the user's situation.

This section of the MAP can be reversed. Indeed, once the method user defines the associations and notices that some new entities have not been defined yet, he can proceed to the definition of the new identified entities. This is what is covered by the <*Define the associations*, *Define the entities*, *by association*> section of the map presented in figure 5.1. Note that in this case, the metamodel concepts stay the same while the intentions are reversed accordingly to the intentions in the section as the starting points become the defined associations.

5.1.2.1 Defining the scopes

The **Scope** concept is used to materialize the relation between the **Situation** and the **User** concepts. Since the situation of the user is the pivotal link between the perception and response capabilities of the system, it is recommended to start by defining the different possible scopes of the identified situations. This concept can hold several properties regarding the situation the user is in and can help establish variability points in the relationships between the user and situation instances (e.g., spatiotemporal validity, severity, priority, etc.).

Application to *SMARTROAD* In the $riskySpeed$ situation discussed above, the $RiskySpeedCar$ scope materializes the relationship between the Car as a user and the $riskySpeed$ situation. For instance, a car car_1 traveling at the speed s might cause a $riskySpeed$ situation on the road $road_1$, while it would be perfectly fine on the road $road_2$. Traveling at a speed exceeding the speed limit by 30km/h on the road $road_1$ might be much riskier than traveling with 5km/h over the speed limit on that same road, as An increase in average speed of 1 km/h typi- cally results in a 3% higher risk of a crash involving injury, with a 4–5% increase for crashes that result in fatalities.. All of these dimensions constitute the scope between the user and the situation[2].

5.1.2.2 Defining the functions

The **Function** concept is introduced to materialize the relation between the **Context** and **Situation** concepts. While it depends on the **Scope**, it allows to translate/map particular context instances to the identified situations. Equation (5.1) presented a way to define the functions. However, to be operational at a finer

[2]https://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/speed_en.pdf. Last accessed on 23 Mai 2021

level, each function should be associated to an identified scope. Hence, several functions could be defined to detect the same situation from the same context entities depending on the scopes.

Application to *SMARTROAD* We use *Functions* to interpret the context data and detect the situation of the users. Table 5.5 shows an example of two functions that can be applied to reason about two context dimensions (i.e., *BT* for Body Temperature, and *HR* for Heart Rate) to detect a *Risky Health* situation. As can be seen in the figure, the defined functions can map the contextual data to actual situations. The definition of these functions is the responsibility of the application domain experts. For example, in the case represented in table 5.5, the functions are defined with the help of experts with medical knowledge.

Table 5.5 – Function examples to detect the Risky Health situation in the *SMARTROAD* system

Situation	*Function*	*Primary Context Dimension*	*Condition*
Risky Health	Medium Risk	Body Temperature (BT)	(BT >38 && BT <41) or (BT <36 && BT >34)
		Heart Rate (HR)	(HR >140 && HR <200) or (HR <40 && HR >30)
	High Risk	Body Temperature	BT >41 || BT <34
		Heart Rate	HR >200 || HR <30

5.1.2.3 Defining the tasks

This concept is used to concretize the relationship between the **Goal** and the **Service** concepts. While it also depends on the **Scope**, it describes the functional (i.e., business) part of the service and represents a small atomic piece of workload allowing the achievement of the goal or a sub-goal. In this thesis, we consider the task as the business functionality provided by the service to simplify the association between the service and the task. The dependence on the **Scope** makes it possible to have the same task performed by different entities or at different levels of spatiotemporal validity.

Application to *SMARTROAD* The tasks in the example scenario are tightly related to the identified services since we assume that each service performs a task. For instance, *Intelligent Speed Adaptation* is a task that can be performed in two scopes. First, at the scope of the *Road*, to target all the vehicles in the *Road*. Second, at the scope of the *Car* in a $riskySpeed$ situation, to target that particular car.

5.1.2.4 Defining abstract service models

This concept is introduced to link up between the **Goal** and the **User** concepts. It allows to describe the sub-goals of the users responding to a particular situation through the coordination of different identified tasks. **Abstract Service Model**s describe the control and data flows to the execution of the tasks in order to achieve the final goal. Note that different users can have different goals to achieve while responding to the same situation. The identified **Abstract Service Model**s depend on the **Scope** and this dependence allows the method user to define several abstract service models to achieve the same goal but depending on the properties of the scope (e.g., geospatial properties, priority, etc.).

Application to *SMARTROAD* Abstract Service Models (ASM) or Service Models (SM) (used interchangeably) provide a concrete way to achieve the identified goals. Table 5.6 shows the abstract service models that are triggered as a response to the identified situations. In this case, we represent Service Models (SMs) as Business Processes that describe how the goal is going to be achieved. We use the example services that we presented earlier in table 1.1 to compose abstract service models for each situation. Each service can target a specific user and thus can have different *Scopes*.

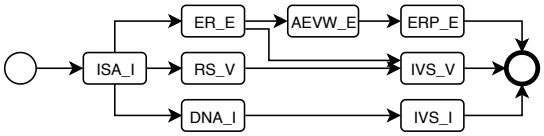
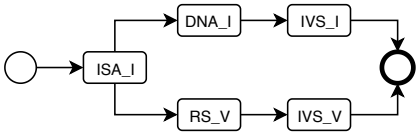
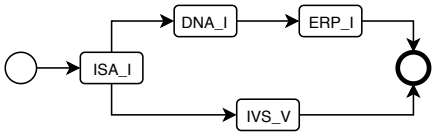
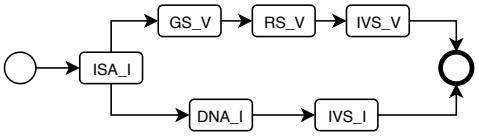
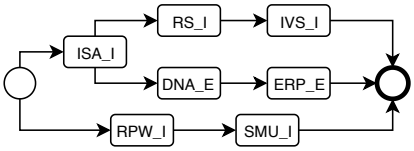
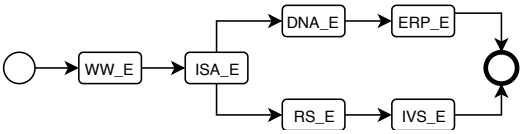
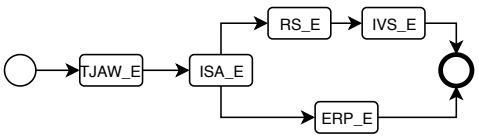
5.1.2.5 Defining quality parameters

This concept is introduced to describe the quality of an action through a **Service**. We distinguish between two types of **Quality** concepts. First, **Quality of Context** represents the service quality parameters related to the operation linked to the handling of the **Context** instance data (e.g., precision, availability, coverage, etc.). Hence, this link materializes the relationship between the **Service** and **Context** concepts. Second, **Quality of Service** represents the service quality parameters related to the operation linked to the service being invoked by the **User**. It allows to measure the quality of the service properties for each user apart (e.g., response time, throughput, price, etc.). Due to their dependence on the **Scope**, the same **Quality of Context** or **Quality of Service** properties can have different values based on the user's situation.

5.1.3 Assigning the resources by service discovery

In this section, we present the concepts that allow the method users to specify and describe the resources that the system needs and/or uses and the operations

Table 5.6 – List of abstract service models that are triggered to respond to detected situations in the *SMARTROAD* system. To show the scope of the task, we concatenate the abbreviation of the task with the first character of the user type (e.g., we put 'IVS_E' to designate the In-Vehicle Signage task targeting the Environment as a scope).

Situation	Goal Description	Triggered Abstract Service Models
Risky Health	Adapt the speed of the vehicles in the infrastructure, choose a route for the person exhibiting the symptoms and inform a nearby health-care facility and nearby vehicles.	
Risky Driver	Adapt the speed of the vehicles in the infrastructure, choose an alternative route for the driver presenting the risk and inform nearby vehicles to be mindful.	
Risky Speed	Adapt the speed of the vehicles in the infrastructure, inform nearby vehicles to be mindful and the speeding vehicle to slow down.	
Risky Vehicle	Adapt the speed of the vehicles in the infrastructure, inform nearby vehicles to be mindful and choose and guide the vehicle in question to a garage for repair.	
Risky Road	Inform the authorities of the road problem, inform nearby vehicles to be mindful, adapt the speed of the vehicles in the infrastructure and choose different routes for them.	
Risky Weather	Warn the users about the weather, inform nearby vehicles to be mindful, adapt the speed of the vehicles in the environment and choose different routes for them.	
Risky Traffic	Warn the users about the traffic, adapt the speed of the vehicles in the environment and choose alternative routes for them.	

they allow (<Define the entities, Assign the resources, by service discovery>). Figure 5.2 illustrates the general strategies and intentions involved in this section

along with the product metamodel concepts allowing their setup in purple background. Two intentions constitute the basis for this section. First, the intention to *Find the available services* allows the exploration of the offered services using the identified **User** instances. Second, the intention to *Reserve the resources to be used* enables the attribution of appropriate computational resources for the found services allowing their smooth execution.

This section of AS3's process model is similar to the <Define the associations, Assign the resources, by service discovery> section. The only difference resides in the starting intention. Indeed, while in this present section the starting intention is that of *Define the entities*, the starting intention in the latter section is *Define the associations*. This ultimately means that the service discovery strategy can be triggered with two separate inputs, which are the entities and the associations. Note, that the goal of both of these sections is to assign the resources that are capable of handling the different processing aspects related to the defined entities and associations.

5.1.3.1 Resource types

The **S_Resource** concept represents the different types of resources that the system can use. We distinguish between 5 types of resources. First, the **Storage** resource type allows to store data (e.g., hard drive, flash drive, etc.). Second, the **Computing** resource type allows to process data and handle arithmetic operations (e.g., CPU, GPU or TPU). Third, the **Input** resource type allows the capture of data from the environment (e.g., sensor, photo camera, touchscreen, etc.). Fourth, the **Network** resource type allows any type of communication over a particular protocol stack (e.g., WiFi unit, DSL unit, Radio unit, etc). Fifth, the **Output** resource type allows to output information in a specific format or a specific channel (e.g., screen, speaker, etc.).

5.1.3.2 Service types

In this thesis, we focus on services exposed by software agents. We suppose that each **User** of the system is capable, through a software agent, of offering a set of services. We classify the types of resources discussed above under two different service types. First, the **BDaaS** meaning **Big Data as a Service** describes the operations that are related to Big Data (e.g., data storage, processing, etc.). Second, the **TaaS** meaning **Things as a Service** describes the operations that are related to the communicating things or smart objects (e.g., sensing, displaying, sending or receiving data, etc.). The **BDaaS** and **TaaS** concepts are explained in more details in the following section. More information about these concepts can

be found in section 3.2, in which a study of the synergies between the different enabling technologies and paradigms involved in smart city systems is conducted.

Application to *SMARTROAD* The aim is to take stock of the resources at the system's disposal. These resources can be either internal to the system (i.e., federated by the system's administrators) or external but usable by the system. Table 5.7 shows some examples of resources that are used in the context of the *SMARTROAD* system. In the first example, for instance, a temperature sensor resource is described as an input resource exposed as a service of type 'Thing' (TaaS) allowing the operation of reading the data from the sensor and can be programmed to constantly stream the sensed temperature data.

Table 5.7 – Examples of resources that are used in the *SMARTROAD* system

Resource	*Type*	*Service Type*	*Interface*	*Programming model*
Temperature Sensor	Input	TaaS	Read;	Stream
EHR	Storage	BDaaS	Create; Read; Update; Delete;	Query
Route Selector	Computing	BDaaS	Compute;	RPC

5.1.4 Defining the entities by service description

Choosing services as the development paradigm for SS design and development has many advantages. As we have stated in section 2.2.3, service-orientation is the *de facto* paradigm for building highly distributed, multi-tenant software systems. One important characteristic of service-orientation and services in general is that they expose a lot of information regarding themselves and their functionalities in the form of service descriptions or services interfaces. In the design phase, the method user can leverage those service descriptions to define new entities that he/she may have missed (<Assign the resources, Define the entities, by service description>).

Service descriptions represent an important resource in bridging the gap between the high level design of the system and the low level technical details that go towards the implementation of the functionalities of the high level design. Hence, writing the service descriptions allows the method user to reflect back on the high level design and identify and define missing entities and associations. To make such connections, the method user can leverage several parts of what constitutes a service description depending on the service description language used for such an endeavor. But, in most of these languages, we can find three parts that constitute a good source of analysis and reflection:

- **Inputs and Outputs**. The inputs and outputs of the service are considered to be its signature in most of the service description languages. Reflecting

on what the service is supposed to consume and produce in terms of data can provide crucial elements to identify new entities or associations in the targeted SS. This is especially true when considering the sources of the data and their recipients.

- **Service provider**. Service providers are an important aspect of the service description. This is especially true when the method user is considering the use of externally provided services. Indeed, external services can come with several dependencies and can cause the scope of the targeted SS to grow considerably.

- **Service user**. The potential users of each service is also an important topic to reflect on when designing services. At this point, reflecting on what the service can effectively cover as user can lead to expand the scope of the targeted SS by adding new users, or contract its scope by removing previously identified elements.

These mentioned parts of the service description can be used to redefine the entities in this present section of the process model and the associations in the <Assign the resources, Define the associations, by service description> section of the process model. Hence the *by service description* strategy is the same in both sections. In the case of defining associations, service description can provide valuable information, especially if the description language allows the specification of effects and preconditions as part of the service's signatures. Indeed, these concepts of preconditions and effects allow the method user to understand new dependencies between the services and other entities that may or may not already have been identified in the initial design.

5.2 Define the elements of the smart system by exploration

As mentioned earlier in section 5.1, the goal of the AS3 is to help the method user to define executable smart system loops. The present section of the method's process model assumes the existence of one or several other SSs that the targeted SS is supposed to interact or collaborate with. This is especially important in domain applications involving multiple stakeholders. Hence, the existing SSs can be represented as external smart system loops that represent the entities, associations and resources of the system and that can directly be identified as potential elements of the targeted smart system. Figure 5.4 illustrates the way in which the targeted SS can collaborate and interact with existing SSs through their Smart System Loops.

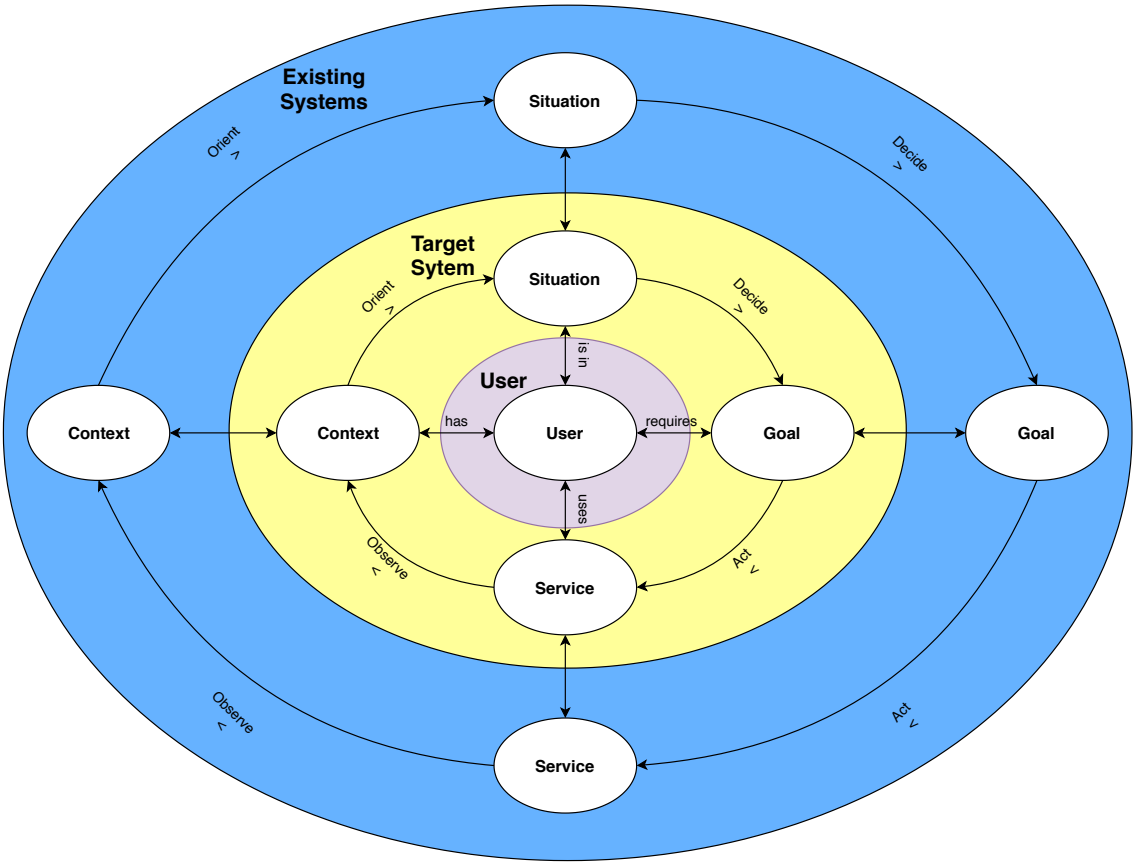


Figure 5.4 – System interaction and collaboration through smart system loops.

The general intention of this section is to enable the method user to identify existing elements within the existing SSs that can be of interest to the targeted SS. This is useful for two reasons mainly:

1. Enrich the design of the targeted SS by identifying potential system integration possibilities early on in the system development cycle.

2. Reduce the amount of time and effort that it would take to identify and reimplement these elements of interest to the targeted SS.

While the general intentions of this section are different from those presented in section 5.1, the product metamodel presenting the concepts to explore remains the same. This is mainly because the concepts are related to the Smart System Loop and the capabilities of SSs which are constant throughout the design of any SS. Since the concepts remain unchanged, the relationships that exist between each pair of these concepts also remain intact. As a result, the strategies between the intentions are the same as the ones presented in section 5.1 in figure 5.1. This section can be hence refined in a similar manner into a new MAP focusing on the

identification of new entities from the analysis of existing SSs. Figure 5.5 presents the MAP formalizing the intentions at this section of the AS3 method.

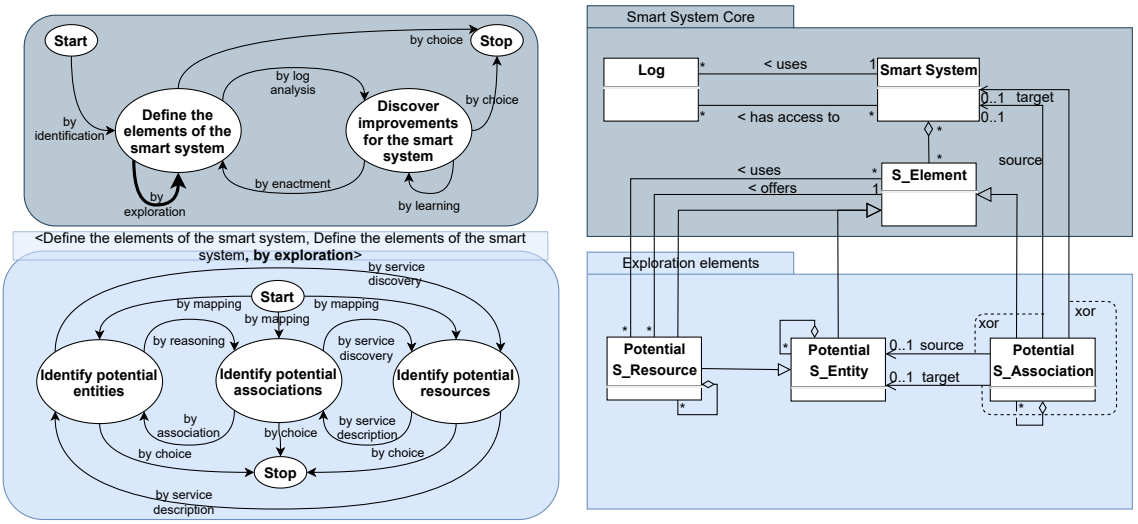


Figure 5.5 – Defining the elements of the smart system by exploring existing smart systems.

As presented in figure 5.5, there are three (3) new sections that are particular to this section and to the exploration strategy. Unlike in the <Start, Define the elements of the smart system, by identification> section, the first targeted intention can be either to *Identify potential entities*, *Identify potential associations* or *Identify potential resources*. This is because we suppose that we already have a set of entities, association and resources that were identified in the previously mentioned section. In the new section the *by mapping* strategy allows the method user to map the set of identified elements to the existing elements with the existing systems.

While the mapping and matching can be achieved by using semantic or synthetic matching techniques, the most important takeaway is to be able to explore design elements and artifacts under specific entities (i.e., user, context, situation, etc.), associations (i.e., scope, function, etc.) and resources (i.e., input, output, etc.). This allows stakeholders, architects and engineers to enrich their initial system design and facilitates any potential interactions with existing (external or internal) SSs.

In the corresponding product metamodel, the concepts of *Potential S_Entity*, *Potential S_Association* and *Potential S_Resource* represent the elements that are found through the mapping between the elements in the existing SSs and the targeted SS. These concepts only represent the tip of the iceberg when it comes to the AS3 method as each of these concepts can be further decomposed into several concepts. This hierarchical structure allows the method user to easily navigate the potential design benefits behind each element within the existing SSs and

thus facilitates the mapping to the product metamodel as can be seen in figure 5.2.

Application to *SMARTROAD* To illustrate how exploration can improve the design of the targeted SS, we use the motivating scenario presented in 1.4. In the presented scenario, the sr_{pp} service is used as an external resource to inform the police of a situation where they need to intervene. Supposing that the service is described and provided by the police department, this identifies the police department as a potential user of the targeted SS. Furthermore, while reading the service description, the method user is made aware that this service requires an image of the VIN (Vehicle Identification Number) as an input to send a successful call to the service. The VIN image can now be considered as a new context dimension that needs to be added as an entity to the targeted SS's design in order to make the interaction with sr_{pp} service possible.

5.3 Initial design of the *SMARTROAD* system

In this section, we first present the design of the *SMARTROAD* system and its components through the system architecture and a component diagram. Then, we move on to explain in more details the workings of each component. This initial design is based on the elements defined throughout this chapter. It proposes a technical description of the resulting SS where the Smart System Loop can be easily traced. This description will be used to implement a *SMARTROAD* system simulator in order to retrieve the data to be used for the discovery of possible improvement later on.

5.3.1 System Overview

Figure 5.6 presents the system architecture when applied to the *SMARTROAD* case study. We chose the layers based on a Separation of Concerns principle (Dijkstra, 1982). The functions of each layer in the system architecture is detailed in the following subsections.

5.3.2 Context Collection

This is the first step and the bottom layer that is needed in every context-aware or knowledge-based information system or software. It provides the necessary methods and processes to extract the necessary data from the users of the system. As we stated before in section 3, context is every bit of information that can help in identifying the situation of a user. The context factors to take into account are

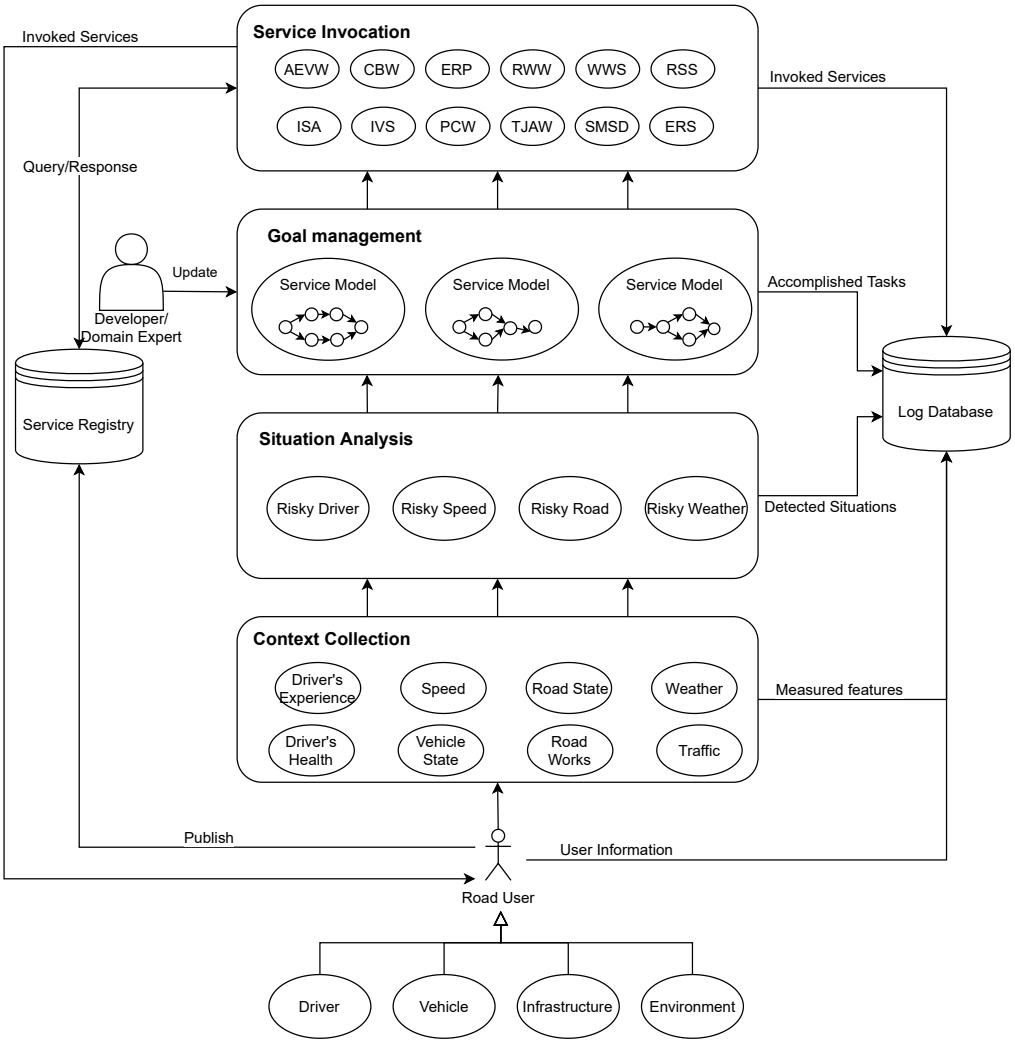


Figure 5.6 – Initial System Architecture for the *SMARTROAD* prototype

closely related to the application. In our case study for example, the goal is to make the road secure. Hence, the context factors that are of interest are the ones that would allow the identification of there being a risk, the nature of the risk, the actors responsible for it and the actors that are affected by it. We denote by $C = (c_1, c_2, \ldots, c_n)$ the set of context factors that are relevant to the application and n the number of these factors.

Context factors can further be divided into two categories. The first category being the **Primary Context**, which consists of the set of context factors that are directly involved in the logic of the system (e.g. Risk factors in our case study). The context factors in this category are a must have for the system to operate. The second category is the **Secondary Context**, which assembles the context factors that may enhance the performance of the system in specific tasks (e.g. location of the user). The context factors in this category are not necessary for the system to

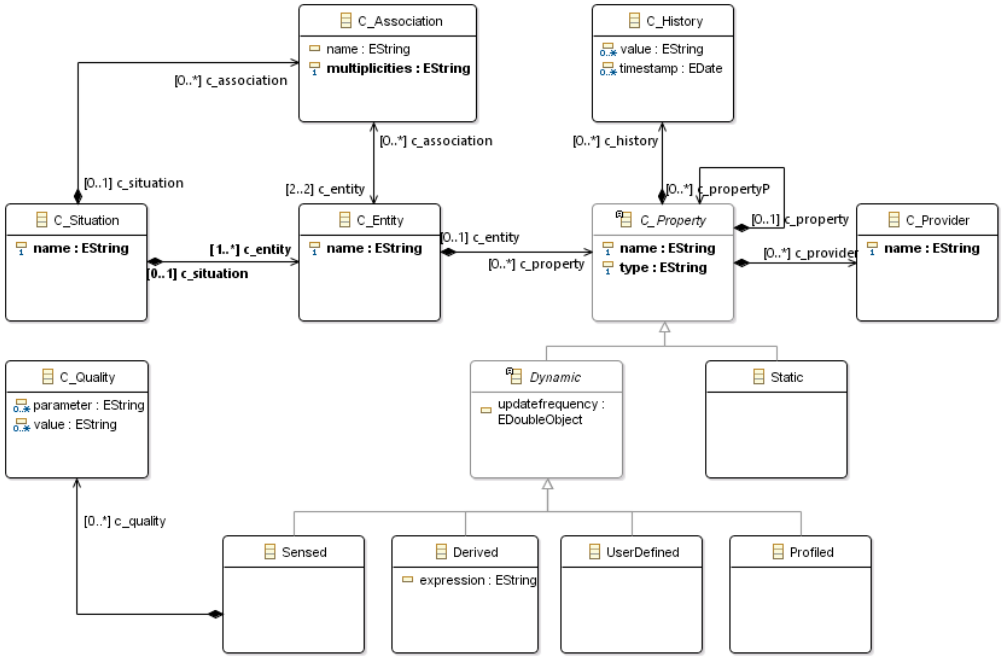


Figure 5.7 – MOF compliant context metamodel

function. However, they can significantly improve its performance when and if they are available and used effectively.

We propose a MOF (MOF, 2014) (Meta Object Facility) compliant metamodel (presented in figure 5.7) to simplify the specification of contextual properties related to the users of the system (Faieq, Saidi, et al., 2017a). This enables the classification of the contextual properties and answers the questions associated with the nature of each context property. At the same time, it allows an abstraction from specifying an exact format to transmit the collected data (e.g. key-value, eXtensible Markup Language (XML), ontologies, etc.).

In this thesis, we assume that the needed context information is accessible through services provided by the actors in the system. In our case study, we consider each risk factor as a primary contextual factor that is monitored through the previously mentioned services (e.g. the speed of the vehicles, the health of the drivers, etc.).

5.3.3 Situation Analysis

Situation Analysis is the task of reasoning over the collected context information. The goal of this layer is to be able to identify the situation of one or more users of the system. To do that, we analyze the collected values of the context factors that are of interest in the application. The analysis consists on setting rules about the values of context factors that would allow an action to be

Environment		Infrastructure		Vehicle		Driver	
Traffic	Weather	Condition	Safety	Speed	State	Experience	Health
0	1	1	0	0	0	1	0

Table 5.8 – User status example

performed upon the activation of a rule statement (e.g. *if* VEHICLE_SPEED ≥ ALLOWED_SPEED *then* INFORM_POLICE). Domain experts are usually tasked with defining these rules if the application domain is of a sensitive nature (e.g., Road security).

We periodically check the situation of the user through analyzing the different context factors specified at the modeling stage. After applying each specified rule on its corresponding context factor, the result is a one hot vector of size n. We call this vector the *Status* of the user. An example of the resulting vector based on our case study in presented in Table 5.8. The example shows that the current driver is inexperienced, driving in a dangerous road and in a bad weather.

5.3.4 Goal Management

Goal Management encompasses the different tasks that are necessary to the design, storage and execution of orchestrations (i.e. service models). The goal of this layer is to provide the developers and domain experts with the necessary tools to create the service models. Particularly, they need to specify the tasks to be performed, the order in which those tasks will be performed and any eventual data dependencies between the tasks.

This layer reacts to the signals sent by the Context processing layer. Indeed, service models are designed to react specifically to each situation that is defined in the Context Processing layer. The Service models are just abstractions of the needed concrete services. They only describe the general behavior of the services and the tasks that are to be performed. For simplicity, we defined a unique service model for each situation. In our case study, the list of situations and their corresponding service models are presented in table 5.6.

5.3.5 Service Invocation

The Service Execution layer is responsible for the discovery, matching and consumption of the concrete services that are available in the registry. The resulting services need to satisfy one or multiple tasks in the service model. The goal here is to handle: i) checking the availability of the services ii) network access between the system and the available services and iii) the data dependencies of the service (i.e., the inputs required by each service if any). As the service discovery (Q. Gu

and Lago, 2009) topic is out of the scope of this thesis, we only use the name of the task to discover functionally relevant services in the service repository.

5.4 Summary

As we've noticed and discussed in part I of this thesis, building SSs is a laborious endeavor that tends to involve several concepts from different scientific, technical and technological perspectives. In this chapter, we introduced the key design elements of the AS3 method to frame the discussion on building SSs as well as their improvement. Inspired by the *Smart System Loop*, we attempted to reconcile between the different involved perspectives as we refined each section of the method's high level process model and its corresponding product metamodel in an intentional approach.

To that end, we started by refining the *elements* of the SS into three connected concepts, which are the *entities*, the *associations* and the *resources*. Each of these elements contributes to the design of an SS in a different way. The *entities* are the building blocs of any SS and they reflect its capabilities to sense and respond to the changes in its internal and external environment. The *associations* represent the glue that ties the building blocs together to form the necessary shapes enabling the potential complex behavior needed to satisfy the requirements of the SS, while the *resources* represent the computational means to perform such complex behavior.

At this level of the AS3 method, defining the elements of the targeted SS represents the general intention and is the first step towards its development. In this context, the proposed process model and its corresponding product metamodel describes three sections allowing the definition of these elements.

- Starting from the requirements, the <Start, Define the entities, by analysis> section provides the necessary intentions, concepts and strategies to arrive at an initial design that provides the basis for a first *Smart System Loop* (execution). The overarching principle in this section is to be able of analyzing the requirements of the system and extract the concept instances for each concept in the product metamodel and then identify and define relationships and associations that may have been missed during the requirements elicitation stage.

- Starting from an initial design of the targeted SS, the <Define the elements of the smart system, Define the elements of the smart system, by exploration> section provided the intentions, concepts and strategies to enrich the initial

design and the scope of the initial design. This is mainly achieved by exploring existing SSs and supposes that at least a documented design of these existing SSs is accessible to the method user. Indeed, these existing systems need to be expressed as external *Smart System Loops* to be exploited by the method user, which is only possible if they were designed using the AS3 method or have enough accessible artifacts to be 'converted' into a *Smart System Loop*.

As a case study that pertains to transport and road security, the *SMARTROAD* SS is a good case study to illustrate the usefulness and the relevance of the AS3 method in developing large-scale SSs that can be thought of as System of Systems (SoS). To design the *SMARTROAD* SS, as a new standalone system, we started by limiting the fragments that we were going to use to two main sections which constitute the *Happy Path* of the AS3 method. We call it the *Happy Path* because it represents the nominal path with which the method user can start and arrive at a first system design.

Next, we started designing the *SMARTROAD* system according to the different fragments of the AS3 method selected in the *Happy Path*. Several design artifacts were generated along the way, leading us to general architecture and a first design of the *SMARTROAD* SS. This first designed was used to implement a simulation of the *SMARTROAD* system using Java. We used this simulation to generate execution traces in order to showcase how the system can be improved using recommendation tools.

5.5 Discussion

The fragments of the AS3 method described in this chapter cover many design and development aspects of SSs. We argue that using an intentional approach to describe the fragments of the method helps and pushes the method user to think holistically about the targeted SS rather than focusing on specific functional aspects at a time. This not only allows the method user to enrich the design of the targeted SS but also setup the field for discovering runtime improvements from execution traces.

In order to be able to enrich and improve the targeted SS, the concepts introduced in the product metamodel are of vital importance as they allow the categorization of the system's entities into finer grained classes that are considered as first class citizens to any SS. These concepts were developed through a careful analysis of several of the literature and our own vision of how SSs and systems in general tend to behave.

Throughout the development of the AS3 method and its different fragments, a particular attention was given to the homogeneity of the intentions and concepts involved. This is reflected by the minimal changes and the high similarity between these different fragments when it comes to achieving the same general intention.

Because of the level of abstraction that is chosen to describe the method's fragments, the method user is left with a lot of the decision making that goes to implement a target SS. This can limit the adoption of the AS3 method in general. However, this level of abstraction is also the main strength of AS3, as it allows the method user and other researchers in the field to extend and refine the AS3 method to make it more complete.

Chapter 6

Discovering improvements for adaptive service-based Smart Systems: a recommendation-based approach

The true function of philosophy is to educate us in the principles of reasoning and not to put an end to further reasoning by the introduction of fixed conclusions

George Henry Lewes

One of the characteristics of Smart System (SS) is their ability to adapt to changing execution environments and improve their overall performance with regard to the satisfaction of their users. In the previous chapter, we presented a systematic way to define the elements of an SS based on an intentional approach. Specifically, we presented the process model and the corresponding metamodel to reach the intentions of each section of the process model towards the goal of defining the elements of a targeted SS such as SMARTROAD. In this chapter, we build upon the defined elements to define ways of making an SS more adaptable. We ground our work on a recommendation approach to continuously improve the design and the performance of the targeted SS via the discovery of possible improvements. This provides system engineers with a decision support system that is valuable to stay in control of the changes made to the system. A rundown of the recommendation-based approach to the improvement of some aspects of the *SMARTROAD* system (see section 1.4) shows its pertinence and effectiveness.

6.1 Discover improvements for the smart system by log analysis

Here, we present the strategies for method users to be able to enact improvement to the system. Figure 6.1 illustrates the general intentions, strategies and concepts involved in pink background color. There are three intentions that need to be achieved. First, the intention to *Define evaluation attributes* allows the method user to specify key factors on which he/she wants the system to improve or optimize. Second, the intention to *Assess evaluation attributes* makes it possible to harness the values of the attributes measured from different users under different situations. Third, based on the recorded values, the intention to *Describe improvements* allows the system to extract patterns in the attributes. These patterns are presented to the method user as a set of recommendations to redefine the elements of the as-is system.

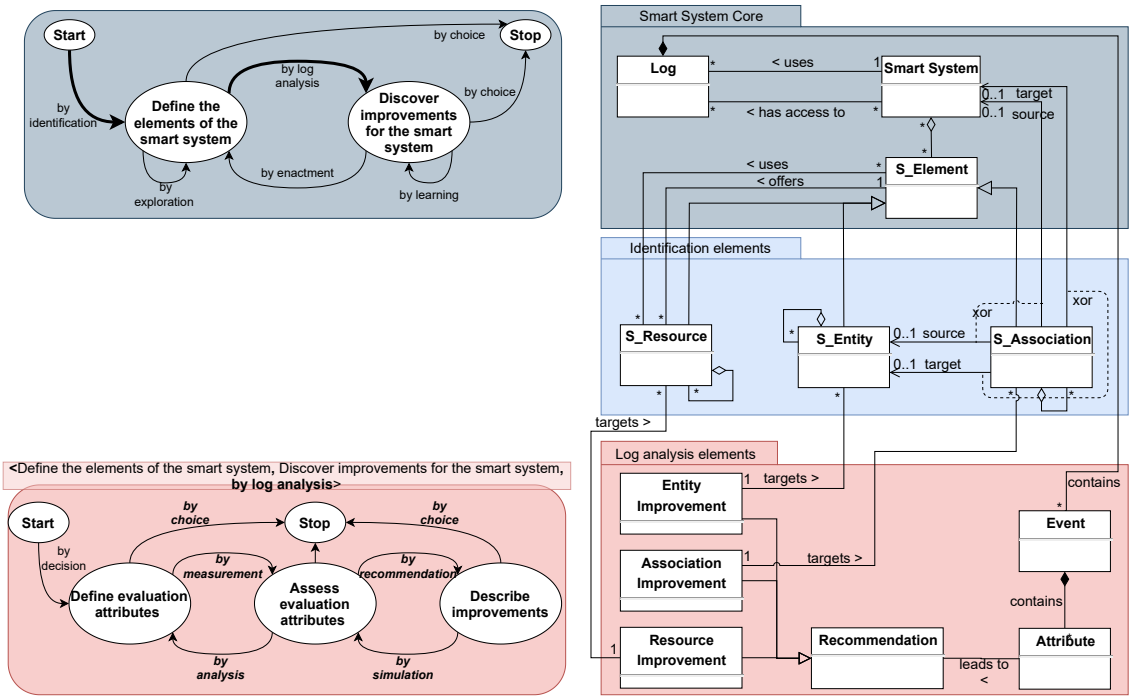


Figure 6.1 – Discovering improvements to the smart system by log analysis.

Application to *SMARTROAD* Possible improvements in the *SMARTROAD* system can target any of the concepts that were introduced in the sections related to the intention *Define the elements of the smart system*. In the following, we target two concepts belonging to the *Reasoning elements*, namely, *Quality of Service* and *Service Model*. For the *Quality of Service* concept, the goal is to improve the recorded quality of service. Improving the recorded quality of service implies the selection

of the best service for the user in his particular context. To improve the *Abstract Service Models*, the goal is to discover new tasks that can be integrated in the existing Service Models, or to create variants of the existing Service Models integrating the newly discovered tasks.

6.1.1 Defining evaluation attributes by decision

To allow the system to improve, the method user is invited to specify the attributes that the system needs to monitor. But, before instantiating the *Attribute* concept, the method user needs to think about how these instances can be measured and/or collected. Note, the *Attribute* concept is an umbrella concept that covers the instances of *S_Element* (e.g., Throughput as an instance of Quality of Service can be an Attribute instance). The specified attributes are monitored and each of their values are stored in the *Log* as part of an *Event*. The *Log* serves as a unit grouping together the events happening at the current system's level in a simple, flexible and expressive data model. An *Event* describes an action that was executed by the system. It can refer to another event if the current event was triggered by the referred event. The set of recorded events constitutes the *Log*. The *Attribute* concept represents and holds each piece of data related to an *Event*. It can be static or dynamic, sensed or defined, objective or subjective. The set of recorded attributes constitutes an event. For example, considering the *SMARTROAD* smart system, an attribute can be the *Driver's identifier* or the *response time* of the *Intelligent Speed Adaptation* service in case a $riskySpeed$ situation is detected.

The discovered improvements are based on the analysis of the system's execution traces and/or the users' feedback. In this thesis, we focus on *Quality of Service* and *Abstract Service Model* as concepts that need to be improved to better serve the needs of the users.

Application to *SMARTROAD* For the *Quality of Service* instances in the *SMARTROAD* system, we chose Response Time (RT) and Throughput (TP) as Quality of Service instances. These instances represent objective data that can be measured after each invocation of a service by a user. *Response Time* represents the elapsed time between the time the user sent his request and the time he/she received the service's response in seconds, while *Throughput* represents the maximum speed of the network connection between the user and the service in Mbps. We retrieved the RT and TP data from the WSDREAM dataset [1] collected by (Zheng, Y. Zhang,

[1] https://github.com/wsdream/wsdream-dataset/tree/master/dataset1

et al., 2014). Regarding the *Service Model* instances, we operate under the assumption that services can be triggered manually by the user or automatically to respond to a prescribed situation. Indeed, following the *Smart System Loop*, presented in figure 4.2, the *Service* concept can be reached through a defined *Goal* or through a manual invocation by the *User*. This assumption implies that a user invokes a service manually because (i) he/she needs the service and (ii) the service wasn't automatically provided to him. Consequently, this difference needs to be monitored to facilitate the distinction between a manually invoked service and an automatically triggered service for a user. We translate this property by tracking the Service Model at the source of the invocation if the Service was automatically triggered.

The dataset describes real-world QoS evaluation results from 339 users on 5,825 Web services. The dataset also contains the location information (e.g., Country, Autonomous System, Latitude, Longitude) of the users and the services. For more details, users can refer to the work of (Zheng, Y. Zhang, et al., 2014). In our prototype we map the country to the environment and the AS (i.e., Autonomous System) to the Infrastructure in both the services and the users files.

We aggregated and preprocessed the data to generate three separate files. The files contain the following elements:

- The *users* data file containing the information about the users (i.e., the values of their context factors and their situation)

- The *services* data file containing the information about the services (e.g., the tasks they perform, location, etc.)

- The *invocations* data file containing the information about the invoked services by each user (i.e. QoS parameters). To simplify the process of reconstructing the service models and filtering the atomic tasks, we also stored the preceding task when performing the current task and the service model it belongs to. Table 6.1 shows an extract of the invocations file, where the User column represents the identifiers of each user, the Service column represents the identifiers of the invoked services, the SM column represents the Service Model from which the service was executed (indicating the user's situation) and Response Time and Throughput represent objective QoS data about the invocation of each service by each user. The *invocation* data file is available on *csv* format in the linked repository at footnote [2].

[2]https://github.com/faieqs/SMRec

Table 6.1 presents a sample of the execution log of the *SMARTROAD* system. Each row of the table represents an *Event*, while each column represents an *Attribute*. To simplify the reading of the table, we refer to each attribute data by the name of the concept instance they represent followed by an identifier if there are multiple instances (e.g., Driver_7). For the SM (Service Model), we use the same name as the Situation instance since it is a unique instance and if the field is empty, we consider that Service was manually invoked by the user.

Table 6.1 – Example of the log file data generated by the *SMARTROAD* system

User	*SM*	*Service*	*Throughput*	*Response Time*
Driver_7		Weather_serv_03	24.1	5.12
Infrastructure_2	Risky Vehicle	SpeedAdaptation_serv_2	14.5	0.48
Vehicle_42	Risky Vehicle	Garage_serv_4	5.4	1.45
Driver_18		Towing_serv_06	16.34	2.15
Vehicle_68		Traffic_serv_01	45.14	1.84

6.1.2 Assessing evaluation attributes by measurement

In this section, the target intention is to *Assess evaluation attributes*. This means that the method user needs to be able to specify assessment measures for the attributes. The assessment measures allow the method user to decide whether the system needs to improve or not and/or whether more evaluation attributes need to be defined in order to make a more informed decision. These decisions are based on the measurement strategies that the method user uses to evaluate the defined attributes.

Application to *SMARTROAD* As we stated in the previous section, the evaluation attributes can be static or dynamic, sensed or defined, objective or subjective. The adopted measurement strategies vary accordingly. For instance, referring to the example scenario of the *SMARTROAD* system in subsection 1.4, assessing the *number of accidents* in the road for a period of time can indicate whether the *riskySpeedResponse* is appropriate for dealing with the situation or whether the response should be modified.

The method user can setup a strategy to measure the attributes allowing him to assess the system's evaluation attributes. The result of this assessment is what triggers the search for improvement possibilities. Regarding the *Quality of Service* in the *SMARTROAD* system, we chose the fact that *a user had never needed the service's task before* as an assessment strategy to trigger the improvement process. In this case, the improvement process entails the search for the best service in terms of QoS to satisfy this particular user's need. On the other hand, regarding

the *Abstract Service Model* in the *SMARTROAD* system, we chose to setup a period of time ($timePeriod =' Week'$) to trigger the improvement process, which in this case, is the search for possible tasks to be added to the response to each situation.

6.1.3 Describing improvements by recommendation

The improvement of the system is a continuous process. In this thesis, this process is triggered by the method user but is performed by the system itself based on the measured evaluation attributes (e.g., users, quality attributes, abstract service models, etc.). The *Recommendation* concept is the center of this phase. It is used to propose the possible improvements to method users, developers and domain experts. We chose recommendation as an improvement technique as it does not carry out actions autonomously, but can easily be automated. The recommendations at this phase can target any of the concepts that are introduced to achieve the intention to *Define the elements of the smart system* (i.e., see section 5.1). *Entity Improvement* represents the recommendations targeting the concepts of the Analysis elements package (i.e., yellow background in figure 5.2). *Association Improvement* represents the recommendations targeting the concepts of the Reasoning elements (i.e., green background in figure 5.2). *Resource Improvement* represents the recommendations targeting the concepts of the Discovery elements (i.e., purple background in figure 5.2). At each level, the improvements can be classified into two categories, (1) discover new elements and (2) modify existing elements. For instance, at *Entity* level, a new type of system users can be discovered through the analysis of the execution log, existing goals can be modified, etc.. Referring back to the *SMARTROAD* system, the decision to add the $policeIntervention$ task to the $riskySpeedResponse$ abstract service model can constitute a recommendation from the system to the system engineers. We consider this type of recommendation to be an *Association Improvement* as it concerns the *Task* and the *Abstract Service Model* which are types of associations.

Application to *SMARTROAD* On the SMARTROAD, the proliferation of services in smart environments has made service selection a challenging task. On one hand, the selected services should be tailored to the need of the user and deliver the best QoS possible at execution time. On the other hand, the high collaboration between the different stakeholders in these environments makes it difficult for the service developers and domain experts to conceive service models that are capable of satisfying the specific requirements of each user. To address these challenges, we propose a recommender-based system that comes atop of the initial design

of the *SMARTROAD* system presented in 5.6. The new system architecture in presented in figure 6.2 The goals of this system are threefold:

1. Analyze the context of the user to personalize the provided services to his particular situation and needs
2. Simplify the collaboration procedures between the different stakeholders to define the service models through recommending the tasks to be integrated
3. Offer the best services in terms of Quality to each user through analyzing previous invocation data

In Service Computing, service recommendation is a topic of great interest. The recommendations are generated based on one or several characteristics (e.g. time, location, QoS, trust, current activity, etc.). In this paper, we focus on two characteristics to generate service recommendations for two different users of the system.

First, for the final users of the system, we generate the best services that are capable of satisfying a particular task. We base the recommendation on the QoS recorded by the system in previous invocations. Second, for the developers and domain experts, we recommend the tasks that may be integrated or added to the predefined service models. To do so, we analyze the services that were invoked by the users and whose service model (i.e., task) is not integrated in the predefined service models. The techniques used at this level are considered as the improvement discovery process and are introduced in the next section.

Figure 6.3 shows a process that illustrates the order in which each task of the system is executed. Figure 6.4 illustrates the actors and the software components of our proposed system.

6.1.3.1 Improving Quality of Service through recommendation

The characteristics of smart environments, such as mobility, heterogeneity and the provision of a myriad of services providing the same functionality (i.e., task), make the task of improving service selection important but challenging. Moreover, there are also problems related to the collected data. First, the data is incomplete. This is mainly because it is impossible to collect all possible invocation possibilities in real world scenarios. Second, the collected QoS data are objective. Hence, they don't reflect a general trend or a profile and depend on the service's supporting infrastructure and demand.

To tackle these challenges, the goal here is to provide the best service in terms of QoS to achieve each task in the service model. This translates to a prediction

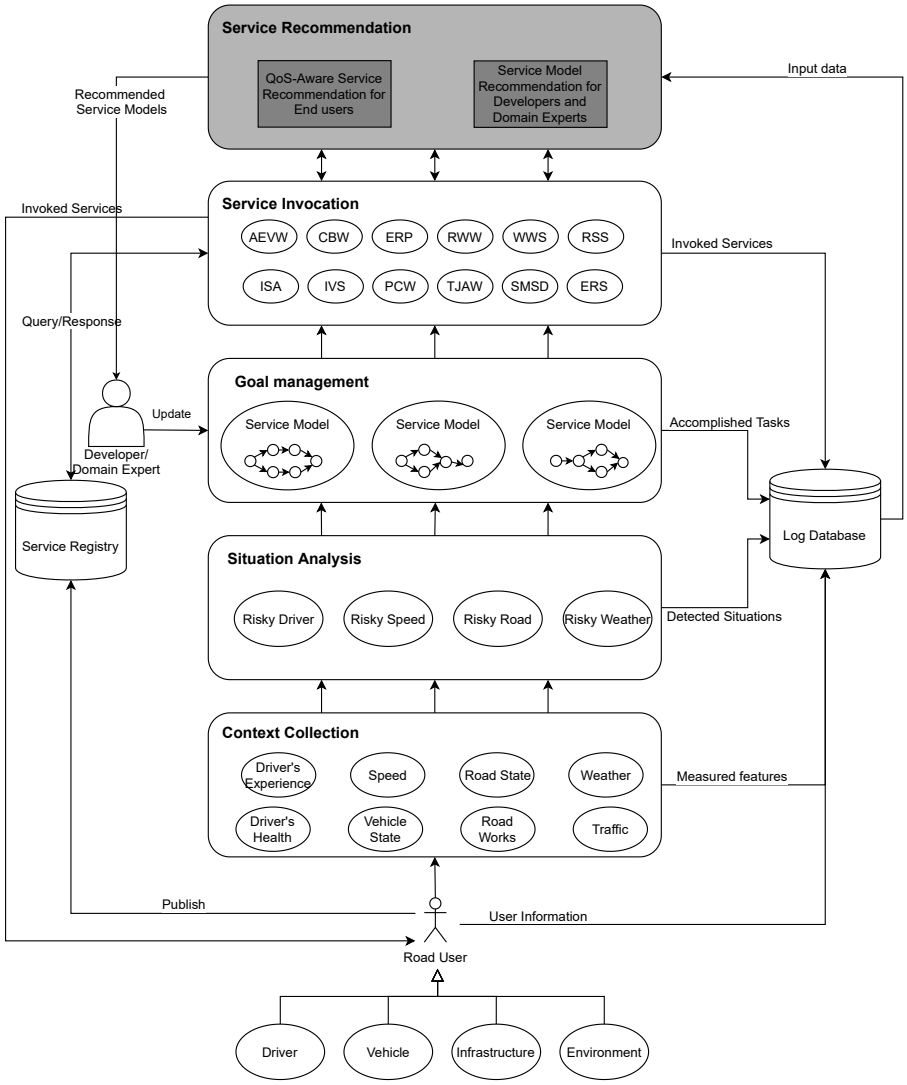


Figure 6.2 – Recommender-based System Architecture for the *SMARTROAD* prototype

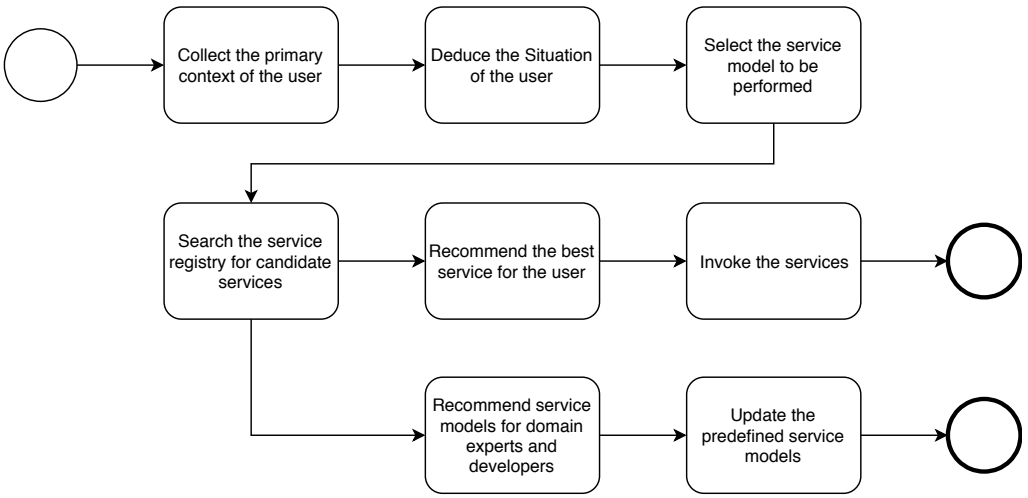


Figure 6.3 – Execution process of the Recommender-based Service Composition System

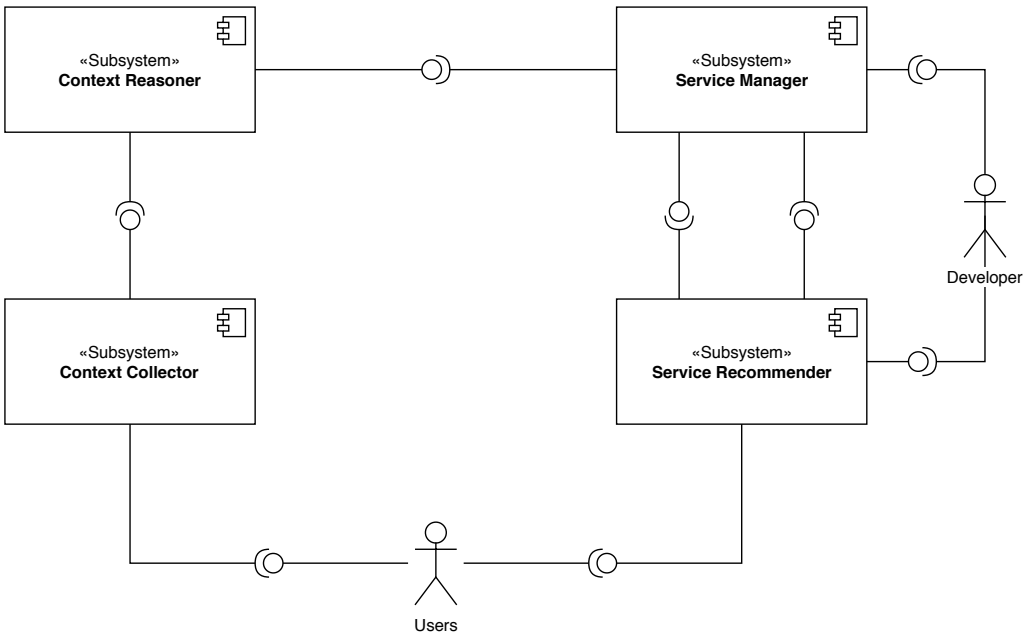


Figure 6.4 – Recommender-based Service Composition System: Component Diagram

problem where the goal is to predict the missing values of a matrix $R \in \mathbb{R}^{nu \times ns}$ where nu is the number of users and ns is the number of services. The rows of the matrix R represent the users of the system U, where each user $u \in U = (1, \ldots, nu)$ can be represented by a partially observed vector $r_{(u)} = (R_{u,1}, \ldots, R_{u,ns}) \in \mathbb{R}^{ns}$. The columns of R represent the services S, where each service $s \in S = (1, \ldots, ns)$ can be represented by a partially observed vector $r_{(s)} = (R_{s,1}, \ldots, R_{s,nu}) \in \mathbb{R}^{nu}$. Each cell in $c \in R$ represents a quality of service property value recorded when a user $u \in U$ invokes a service $s \in S$.

The challenges here are mainly related to the nature of the data generated by the service. In this paper, we focused on QoS data which are objective; meaning that contrary to classic recommendation problems where the data are subjective and hence reflect the user's satisfaction with an item (i.e., movie, music, news, etc.); data in QoS service recommendation data is objective and hence only represents objective factors about the services for it a certain user. Basically, that means that unlike recommendations over subjective data where identical user are more likely to give a close ratings for the same item, it is highly possible that the identical users have varying ratings for the same service in QoS data.

Application to *SMARTROAD* To achieve this task, we adopted the framework proposed by (Sedhain et al., 2015) called AutoRec, which is a Collaboration filtering framework based on Autoencoders. In their approach, they aim to design an autoencoder which can take as input a set of partially observed $r_{(s)}$ or $r_{(u)}$, project it

into a low-dimensional latent space, and then reconstruct it in the output space to predict missing ratings. They formally defined the problem as, given a set X of vectors in $\mathbb{R}^d$, and some $k \in \mathbb{N}_+$, an autoencoder solves

$$\min_{\theta} \sum_{r \in S} \|r - h(r;\theta)\|_2^2$$

, where $h(r;\theta)$ is the reconstruction of input $r \in \mathbb{R}^d$,

$$h(r;\theta) = f(W.g(Vr + \mu) + b)$$

$f(.)$ and $g(.)$ are activation functions. Here, $\theta = \{W; V; \mu; b\}$, where $V \in \mathbb{R}^{k \times d}$ and $W \in \mathbb{R}^{d \times k}$ are the weight matrices for encoding (i.e., projecting) and decoding (i.e. reconstructing) the input respectively, and $\mu \in \mathbb{R}^k$ and $b \in \mathbb{R}^d$ are biases. This objective corresponds to an auto-associative neural network (Stone, 2008), with a single hidden layer (HL) with k-dimension. The parameters θ are learned using back-propagation. To avoid overfitting, we regularize the learned parameters by introducing the regularization term $R = \|W\|_F^2 + \|V\|_F^2$ and use a regularization rate parameter λ to control the strength of the regularization.

To the best of our knowledge, the application of this approach in the Service Computing domain was never attempted. Hence, this paper can be considered a stepping stone towards the integration and application of more deep learning frameworks in Service Computing. The hyperparameters of our model are shown in figure 6.2. The data used for the experiments regarding QoS data was extracted from WSDREAM dataset[3] coupled with our developed simulation to create a log file (i.e., invocations file)[4].

We ran an implementation of the User-Based AutoRec (U-AutoRec) approach on the data collected in the *invocations* file using TensorFlow[5]. The experiment was conducted using a Laptop with a seventh generation Intel i5 processor, 8GB of RAM and 6MB of cache memory. The model parameters chosen throughout the experiments are presented in table 6.2.

The knowledge about QoS properties is not complete in real use cases, as that would imply that every user has invoked every available service. To reflect this aspect of sparsity on the WSDREAM dataset, we used less of one 3rd of the data to build the model. Particularly, we built the model using 4 different matrix density levels, being $MatrixDensity = 5\%, 10\%, 20\%, 30\%$. $MatrixDensity = 5\%$ means that we randomly select 5% of the data to predict the remaining 95%. Each experiment was ran 5 times.

[3] https://github.com/wsdream/wsdream-dataset/tree/master/dataset1
[4] https://github.com/faieqs/SMRec
[5] https://www.tensorflow.org/

Evaluation Metrics To evaluate our U-AutoRec-based model when compared to other QoS prediction methods, we focused on the accuracy of the predicted values. We define accuracy as a function describing the distance between the observed data and the predicted data. In our case, the observed data corresponds to the QoS data recorded in the WSDREAM dataset, meaning Response Time and Throughput data. The predicted data corresponds to the data predicted by our model as an output. In this context, We use MAE and RMSE as two basic and popular error metrics that are commonly used in evaluating Recommender Systems (Candillier et al., 2007). While MAE measures the mean magnitude of errors over a set of prediction, RMSE is the square root of the average of squared differences between prediction and actual observation. This makes RMSE more interesting when large errors have a lot more negative effects on the system that small errors. MAE and RMSE are defined as :

$$MAE = \frac{\sum_{u,s} \mid r_{u,s} - \widehat{r}_{u,s} \mid}{N}$$

$$RMSE = \sqrt{\frac{\sum_{u,s}(r_{u,s} - \widehat{r}_{u,s})^2}{N}}$$

Where $r_{u,s}$ denotes the expected QoS value of service s observed by user u, $\widehat{r}_{u,s}$ is the predicted QoS value, and N is the number of values.

Performance Comparison We compare the performance of U-AutoRec with the performance of several other methods. In literature, each method was evaluated on a different set of $MatrixDensity$ values. This makes the comparison highly challenging as source code is rarely publicly available. Hence, in this paper, we compared the U-AutoRec method with the methods reported in the WS-DREAM package (WSDREAM, 2011). The package reports the performance of several methods. Some of these methods are based on Neighborhood (Sun et al., 2013; J. Wu et al., 2013; Zheng, Ma, et al., 2011), some on models (Yu et al., 2014; Y. Zhang, Zheng, et al., 2011; Zheng, Ma, et al., 2013) and others are location-aware (X. Chen, Liu, et al., 2010; X. Chen, Zheng, et al., 2014; He et al., 2014; Lo et al., 2012; Tang et al., 2012). Table 6.3 shows the mean MAE and $RMSE$ values of different prediction methods in response time (RT) measured in seconds and throughput (TP) measured in Kbps, using 5, 10, 20, and 30 percent as $MatrixDensity$ (MD) values.

Table 6.2 – Model Parameters

N° HL	k	Learning Rate	λ	Batch Size	Optimizer	A. Function
1	500	0.0001	0.1	32	Adam	sigmoid

The results presented in table 6.3 show the performance of the proposed model based on the U-AutoRec framework when compared to other methods. From table 6.3, it is clear that model-based approaches outperform those based on Location-awareness and neighborhood. Furthermore, the proposed model performs well even when compared to other model-based methods. Analyzing the performance of the approach on Throughput (TP) data, we notice that the proposed model achieves 33.2% gains in *MAE* and 35.3% in *RMSE*, when compared with the individual best approaches over all matrix density values. We refer the readers to (Faieq, Front, et al., 2019) for a more detailed analysis and discussion of the results.

The results presented in table 6.3 show that model-based methods generally outperform those based on Location-awareness or neighborhood. Furthermore, U-AutoRec performs exceptionally well even when compared to other model-based methods. This is especially true, when analyzing the performance of the approach on Throughput (TP) data, where U-AutoRec achieves 33.2% gains in *MAE* and 35.3% better in *RMSE*; when compared with the best approaches over all matrix density values individually. On the throughput (TP) data, U-AutoRec also scores an improvement of 32.1% in *MAE* and a 29.4% gain on *RMSE*. While U-AutoRec already outperforms the other methods in most most cases, there is still room for improvement. Indeed, with the huge number of combination that can be used to finetune the model (i.e., number of iterations, optimizers, number of hidden nodes, activation functions, etc.), it is most likely that our experiment did not reach the limits of the AutoRec framework. Extensive experiments and finetuning techniques are needed in this direction to identify the best parameters for the model.

We have also ran experiments using I-AutoRec, which follows the same model as U-AutoRec with the difference that it relies on the items representations rather than the users representations to build the model. I-AutoRec revealed some interesting results. I-AutoRec performs generally better on throughput data, reaching an $MAE = 14.63$ and an $RMSE = 38.26$, when $MD = 10\%$; and an $MAE = 12.55$ and an $RMSE = 34.77$, when $MD = 20\%$. However, U-AutoRec still outperforms I-AutoRec in both response time (RT) and throughput (TP) data. This leads us to believe that response time data is more sensitive to the features of the users, while throughput data is also sensitive to the features of the services. This is logical as response time would change depending on the network state of the users (e.g. signal strength), while throughput depends on the network state of the service's supporting infrastructure and that of the user.

Methods	MD = 5%				MD = 10%				MD = 20%				MD = 30%			
	RT		TP		RT		TP		RT		TP		RT		TP	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
UIPCC (Zheng, Ma, et al., 2011)	0.6253	1.3879	26.7568	60.7985	0.5815	1.3302	22.3700	54.4563	0.4498	1.1968	18.9276	48.2950	0.4110	1.1422	17.0797	45.0599
ADF (J. Wu et al., 2013)	0.6094	1.3613	24.9961	60.7939	0.5443	1.2924	21.5013	54.2893	0.4636	1.1898	16.6536	45.2008	0.4276	1.1398	14.8244	41.7186
NRCF (Sun et al., 2013)	0.5532	1.4547	23.3275	59.9498	0.4905	1.3678	18.8571	52.9977	0.4261	1.2581	14.3444	44.5142	0.4059	1.1975	12.8267	40.7493
RegionKNN (X. Chen, Liu, et al., 2010)	0.5883	1.5426	25.6324	67.8678	0.5477	1.5129	24.8380	67.5510	0.5158	1.5214	24.0361	66.1760	0.5091	1.5193	23.7984	65.5971
LACF (Tang et al., 2012)	0.6374	1.4436	23.1685	58.9666	0.5659	1.3420	19.6257	53.1050	0.4827	1.2298	16.6669	47.6247	0.4453	1.1720	15.2358	44.7729
LBR (Lo et al., 2012)	0.5499	1.4741	18.3187	56.0215	0.4802	1.2858	15.4272	47.0197	0.4300	1.1610	13.6512	41.3552	0.4103	1.1214	12.9824	39.2269
HMF (He et al., 2014)	0.5595	1.5248	19.1320	58.7088	0.4815	1.3105	15.7187	48.3461	0.4296	1.1661	13.6319	41.6678	0.4072	1.1178	12.7767	39.0613
LoRec (X. Chen, Zheng, et al., 2014)	0.6479	1.3957	27.7773	62.5533	0.5557	1.3087	24.7118	58.4275	0.4659	1.2040	21.7440	54.0139	0.4437	1.1504	20.5338	52.0747
PMF (Zheng, Ma, et al., 2013)	0.5690	1.5371	19.0946	57.9078	0.4866	1.3163	15.9741	48.0393	0.4308	1.1690	13.9085	41.7277	0.4082	1.1206	13.1782	39.4635
NMF (Y. Zhang, Zheng, et al., 2011)	0.5456	1.4727	18.8824	57.5177	0.4775	1.2824	15.5717	47.7903	0.4291	1.1616	13.5259	41.6482	0.4063	1.1191	12.7851	39.5068
BiasedMF (Yu et al., 2014)	0.5934	1.3821	21.8369	56.8575	0.5135	1.2625	17.8522	48.3275	0.4569	1.1786	14.9224	42.1408	0.4280	1.1416	13.6980	39.6656
LN-LFM (Yu et al., 2014)	0.5602	1.3065	20.4018	52.4209	0.5007	1.2284	17.7273	46.9713	0.4512	1.1541	16.3428	44.1434	0.4351	1.1282	16.0625	43.4570
Our Model	**0.361**	**1.010**	**13.542**	**37.119**	**0.3343**	**0.9035**	**10.360**	**30.668**	**0.2953**	**0.8019**	**8.644**	**26.569**	**0.2685**	**0.7688**	**7.952**	**24.562**

Table 6.3 – Performance comparison between the QoS prediction models. MD (Matrix Density) refers to the portion of the dataset for which the values are known.

6.1.3.2 Improving Abstract Service Models through recommendation

Due to the ever changing business landscape and emerging user needs, changing business processes is a must. However, domain experts and system engineers cannot keep up with the pace of these changes. This exposes the need for recommendations about the tasks to be integrated in the predefined service models and business processes.

Application to *SMARTROAD* The goal here is to inform domain experts and developers about the need for potential changes in the predefined service models. Even further, we aim to provide them with the potential elements to be integrated into that change. To do so, we analyze the execution traces (i.e. $Events$) that were recorded to discover new patterns in the data. We proceed first by filtering the traces to extract the tasks which do not belong to any predefined service model (i.e. $UniqueAtomicTasks$, where UAT which means $UniqueAtomicTask$ is an element of *UniqueAtomicTasks*). In the mean time, we group the atomic events together by task (i.e. *GroupedAtomicEvent*, where GAE which means $GroupedAtomicEvent$ is an element of $GroupedAtomicEvents$). Second, we extract the set of users who needed the tasks of the invoked services (i.e. $Users$, where U which means $User$ is an element of $Users$). Third, we analyze the events targeting those users to extract the most frequently invoked service models (i.e. $MostFrequentServiceModelSets$, where FSM which means *FrequentServiceModel* is an element of *MostFrequentServiceModelSets*). In this final step, we use the FPGrowth algorithm (Han et al., 2000a), which is a classic Frequent Item Mining and Association Rules Mining approach known for its efficiency and effectiveness when compared to other known algorithms like Apriori and Eclat (Garg and Kumar, 2013). The general process is presented in algorithm 1.

The proposed algorithm takes as input two parameters. First, a log file containing a set of *Events*. Each *Event* in the log file is a tuple $(User, Task, ServiceModel)$ where each tuple (i.e. event) represents a performed task, its targeted user and the Service Model it belongs to if any. Second, a value P with $0 < P \leq 100$ representing a percentage threshold after which the recommendation becomes relevant to the domain experts and developers. Note, this threshold affects only atomic tasks and is used as a minimum support to check whether an atomic task is frequently associated with a Service Model. As output, the algorithm returns a set of recommendations R. Each $r \in R$ is a tuple (SM, AT) where SM is a predefined Service Model and AT is an Atomic Task than be used to extend SM.

Filtering the data allows to only extract the item sets that are associated with the atomic tasks. However, the domain experts and the developers still have

to configure the frequent item set mining algorithm by setting the minimum support. The setup of the algorithm's parameters is needed to make the decision of adding the atomic task to the predefined service models.

Algorithm 1: Algorithm for Service Model Recommendation for Developers and Domain Experts.

Input : The set of recorded events $Events$, such that each event is described as: $Event = (User, Task, ServiceModel)$ and P being the threshold after which the service is considered relevant for recommendation

Output : Set of pairs $R = (SM, AT)$, Where SM is the predefined Service Model and AT is the recommended Atomic Task for integration

```
1  R ← ∅;
2  AtomicEvents ← getAtomicEvents(Events);
   ;                                  // Events with empty ServiceModel field
3  GroupedAtomicEvents ← AtomicEvents.GroupbyTask();
4  UniqueAtomicTasks ← AtomicEvents.GetUniqueTasks();
5  foreach UAT ∈ UniqueAtomicTasks do
6      foreach GAE in GroupedAtomicEvents.getGroup(UAT) do
7          Users ← GAE.getUsers();
8          NumberOfUsers ← Users.getCount();
9          ServiceModels ← ∅;
10         foreach U ∈ Users do
11             ServiceModels.Add(U.getServiceModels());
12         FrequentServiceModels ←
            FPGrowth(ServiceModels, NumberOfUsers × P/100);
13         if FrequentServiceModels ≠ ∅ then
14             MostFrequentServiceModelSets ← FrequentServiceModels.pop(3);
               ;                                  // get first three results
15             foreach FSM ∈ MostFrequentServiceModelSets do
16                 R.Add(UAT, FSM);
17 return R
```

To validate our approach, we injected records of executed atomic services in the invocations file with different probabilities. The objective is to prove that the proposed algorithm is capable of extracting and recommending relevant Atomic Tasks for them to integrate into existing service models; thus achieving continuous improvements. Hence the accuracy metric for this part is the capacity of our algorithm to discover these injected patterns. We considered two situations where we injected the following patterns.

- That 80% of the users who encountered a RiskyVehicleState situation have searched for a Gas Station to fill the gas tank of their vehicle (i.e. GSSS). Note, the GSSS service is not part of any predefined Service Model (see figure 5.6).

- That 40% of the users who encountered a RiskyVehicleState situation have searched for a Towing service to transport their vehicles (i.e., TSS).

We ran an implementation of algorithm 1 in Python, which is available in the linked repository at footnote [6]. As inputs, we entered the path to the *invocation* file and $P = 50$ being the threshold percentage of the users who have performed the task before it is linked to the service model. While no results were found for the TSS service, the GSSS was recommended to be integrated in the RiskyVehicleState as expected. We expect the algorithm to reveal interesting recommendations when applied on real data.

6.2 Discovering improvements for the smart system by learning

The main intention of the previous section was to discover improvement to the elements of the system and enrich its design and functional scope. In this fragment of the AS3 method, the objective is to allow the SS to optimize its performance by learning from its own past behavior recorded through the logs. Since this objective targets the performance of the system as a whole, it is only logical that is involves all of its elements and how they relate to each other.

In this context, the performance of the system is reflected by the usage it makes of the available resources. Indeed, resource usage can change drastically as improvements are brought to the SS. These changes are hard to anticipate beforehand and cannot be optimized at design time. Hence, there is a need to learn from the system's behavior and performance over time to adopt adequate strategies to resource usage through scaling (vertically or horizontally).

To better understand the objective behind this fragment, we formally describe the problem to guide the method user in defining the system's performance objective. Let us consider an SS as several entities that are interacting together to propose added value services to their users that cater to his needs and preferences. Also, an SS is technology-agnostic and is able to improve over time. Smart systems are dynamic and thus their configuration varies in time. For instance, new services might be added to the system or new collaborations between different organisations might be created or removed.

To study the behavior of smart systems and discover how they can be designed with the intention to self-optimize, we formally define a smart system, we denote SS, as a 4-tuple $SS_t = (X, Y, R, RS), t \in \mathbb{N}$, where X is the set of entities (i.e., instances of the building concepts in the smart system loop) in play in a targeted

[6] https://github.com/faieqs/SMRec

system SS, Y is the set of binary relationships between the entities of X, where $\forall y \in Y, \exists (x_1, x_2) \in X, x_1 \neq x_2$ where $x_1 \xrightarrow{y} x_2$. X represents the instances of the building concepts of the system (i.e., $X = U \cup C \cup ST \cup G \cup SR$) and each element x in X ($x \in X$), is a tuple $x = (id, name, description)$, where

- U is the finite set representing the users of the system.
- C is the finite set of context dimensions accessible through and by the system.
- ST is the finite set of possible situations of the system.
- G is the finite set of goals that can be achieved through the system.
- SR is the finite set of services that can be provided or consumed through the system.

RS defines the set of binary relations between the business services SR and the configurations of resources they use to achieve their business goal $CR = \{CR_0, CR_1, \ldots, CR_n\} \mid n \in \mathbb{N}$, where each CR_i is a subset of R ($CR_i \subseteq R$, $0 \leq i \leq n$), and the union of all $CR_i \in CR$ is a subset of R ($CR_0 \cup CR_1 \cup CR_2 \cup \cdots \cup CR_n \subseteq R$). R is the set of resources that are usable by the system SS (they can be internal or external resources that can be controlled by the system). These resources allow the basic information-related operations of acquiring data (input), transforming it (compute or process), presenting it (output) and storing it. We assume that the resources R expose their functionalities as services and these services are in turn composed to create added value services. This kind of services provide IT support for business services and are capable of scaling up and down depending on the demand. Hence, each business service can invoke several resource-based services to achieve its task. Formally, we define RSS as a mapping between each business service and the resource-based services it uses $RSS = (sr, cr) | sr \in SR, cr \subseteq CR$.

We call $SS_{IP} = \{SS_0, SS_1, \ldots, SS_n\}, n \in \mathbb{N}$, the *Improvement Path* of the smart system SS and $I = \{i_0, i_1, \ldots, i_m\}, m \in \mathbb{N}, m \leq n-1$, the set of improvements allowing the system SS to move from one configuration to another. Formally, we note $\forall (SS_j, SS_{j+1}) \in SS_I, \exists i \in I, SS_j \xrightarrow{i} SS_{j+1}$. Note, the term *Improvement* is relative and does not necessarily mean a gain in performance. It simply implies that the system administrators enacted a past improvement recommendation or have enacted their own improvement policy.

In this context, we formally describe the improvement problems that need to be addressed in this method fragment using the following statements:

1. *Find the best resource configuration $CR_b \subseteq R$ that minimizes resource consumption.*

2. *Find configuration patterns that optimize/improve the performance of the system.*

Given enough historical data on the SS's design and performance, these problems can be tackled using recommendation techniques to help the system engineer in making the best decision to improve the SS.

6.3 Defining the elements of the smart system by enactment

Continuous improvement is an important characteristic of SSs. It allows the SS to be adaptable and optimize its performance accordingly to the conditions in its environment. As we've noticed back in chapter 3, continuous improvement paradigms have been focused on optimizing the performance of the system at runtime while considering the set of system constituents (i.e., design elements) to be fixed. In this thesis, we argue that focusing only on improving the performance of the current state of an SS limits the scope of possible improvements. Hence, we discuss, propose and describe how continuous improvement should cover design elements as well as runtime performance.

While the details of how the AS3 method deals with continuous improvement at both the design and runtime performance levels are presented in the previous sections of this chapter, we introduce in this section how the discovered improvement can be brought about at the design level. Hence, the notion of recommendation that we'll be using in this section is the byproduct of the *Improvement discovery* intention that was introduced earlier. At this point, let us consider a *recommendation* as a data-based suggestion introducing a possible improvement to the current state of the system.

One of the guiding principles that led us to recommendation as a technique to discover and implement improvements is that it gives the method user the choice of whether to incorporate the discovered improvements or not. This constitutes the basis of the present section and is materialized by the *by recommendation approval* strategy in the corresponding process model. Figure 6.5 presents this section of the method with the involved concepts and intentions.

The structure of this section's process model is not any different from that of the MAP section presented in section 5.2. The only difference is in the adopted strategy *by recommendation approval*, in which the method user can approve or deny the provided recommendation. Approving the recommendation means the redefinition of the elements of the SS and hence can lead the method user to start

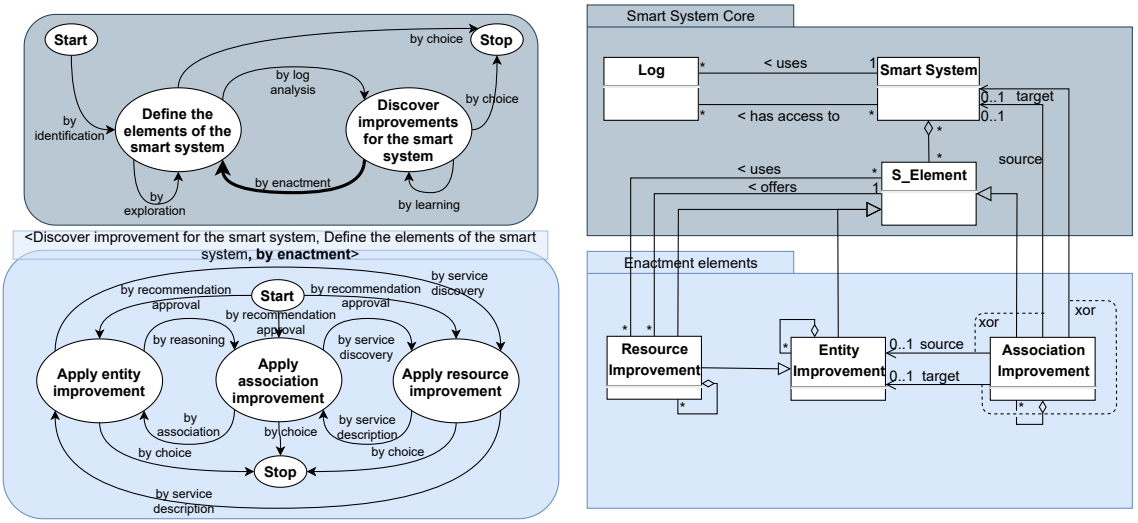


Figure 6.5 – Defining the elements of the smart system by enactment of discovered improvements.

a new identification cycle to grasp the full scope of the impact of the recommendation. This is because the concepts of entities, associations and resources are connected and the strategies allowing the method user to navigate between them remain valid.

In the corresponding product metamodel, the concepts of *Entity Improvement*, *Association Improvement* and *Resource Improvement* represent the recommendations as they were encoded in the improvement discovery process, and thus depend on the algorithms used in that process. However, since the improvements at the design level are recommendations targeting the elements of the system, the concepts and relationships of the product metamodel remain unchanged.

To illustrate, we will proceed with an example around the resources of an SS. In the life cycle of any system and especially in SSs, the lack of resources can cause the system to misbehave or shutdown some or all of the system's components. Resource failure (or predicted resource failure) can be considered as a runtime problem that warrants the recommendation of *Resource Improvement*. By issuing this recommendation, the method user can approve the recommendation and reassign new resources to the defined entities or redefine the entities in a way that allows him to assign different resources, depending on the issued recommendation.

Application to *SMARTROAD* For instance, the context dimension *vehicleSpeed* identified in the example scenario presented in section 1.4 can be obtained through a radar as a resource, in which the functionality to sense the vehicle's speed is exposed as a service. Technical issues or failure at the radar's level prompts the

method user to reassign the resources. In this example, this can be achieved through two options. First, the method user can change the nature of the resource through which the system senses the speed of the vehicles. To that end, the method user could either switch to a video camera as a resource that is capable of detecting the vehicles speed. Second, the method user can take this as an opportunity to install new radars in the area or define new ways to get the information about the *vehicleSpeed*.

While some resource failures need modifications at the entity level to remedy or mitigate their impact on the SS, other resource failures only concern the association elements of the AS3 method. Thus, the *Association Improvement* concept is also used to allow the method user to potentially improve the state of the system by acting on the associations. In turn, this allows the method user to redefine the associations to mitigate the failure's impact on the system and resume a normal or degraded operational status. Also, redefining the entities due to resources failure usually triggers a redefinition of the associations involving the redefined entities.

Going back to the example above, the introduction of the video camera as a new resource to get the *vehicleSpeed* context information needs to be accompanied by a different function that maps the *vehicleSpeed* and the *roadSpeedLimit* context dimensions to the *riskySpeed* situation. This is because the *vehicleSpeed* context information as computed by the video camera can be of a lower quality than that captured by the radar. The introduction of a new algorithm that can provide better precision on the vehicle's speed can be considered as an association improvement. This association improvement would allow the method user to redefine the function mapping the context dimensions to the situation at the association level without redefining any entities.

6.4 Summary

In this chapter, we focused on how SS can be designed to be improved through the careful definition of data dimensions that need to be monitored. Monitoring data of interest and the processing of these data, are invaluable in order to provide the method user with the appropriate information to help him make informed decisions towards the improvement of the targeted SS and the optimization of its performance. With these general intentions in mind, two fragments of the AS3 method were designed. As improvement and optimization are runtime processes that heavily on data, these two fragments represent modeling elements designed to help the method user to setup the base for such processes by selecting the right data and expressing the needs in a way that would allow choosing the right

techniques.

- Starting from a complete definition of the elements of the targeted SS, discovering improvements to the SS requires the analysis of the log containing execution traces of the system. To that end, evaluation attributes that need to be included in the log, need to be taught about and defined at design time. These evaluation attributes represent special properties of the SSs entities and their associations that allow the system to improve or self-improve over time. To implement the discovery process, we chose recommendation as a paradigm that would allow the method user to stay informed and in control of the changes and improvements that can be brought to the system but that also can be easily automated to make the system self-improve.

- Over time, the functional changes that are brought to an SS can have an unexpected impact on its performance. Indeed, resource usage can become exceedingly big and thus expensive and difficult to sustain. Ergo, discovering the best system configuration to minimize resource consumption or finding a compromise between the functional and operational aspects of the targeted SS can be necessary to keep a sustainable system. To that end, in section <Discover improvements for the smart system, Discover improvements for the smart system, by learning>, we proposed a formal model of the system's evolution to help the method user to formally define his optimization objective. Defining the optimization objective facilitates the choice of techniques that can be used to find the optimal system configuration that can achieve that objective.

- Defining, or more accurately, redefining the elements of the targeted SS can also be a byproduct of the continuous improvement process. This is a fairly new idea that we are promoting to enrich and improve the design of the SS at runtime. To that end, the <Discover improvements for the smart system, Define the elements of the smart system, by enactment> section introduced the intentions, concepts and strategies that allows the method user to improve and enrich the design of the target SS through approving or denying recommendations. At this level, the method user is also encouraged to analyze the impact that enacting a recommendation may have on other elements and thus on the general design of system, as each instance of an element can be connected to another instance of the same type or of a different type.

We also focused on two aspects of the *SMARTROAD* system that can be improved using the gathered data. First, a model based on the AutoRec framework

for the prediction of QoS values was used to improve service selection based on QoS attributes. Second, an algorithm for recommending tasks to be integrated in existing service models or creating new variability models was designed and developed to improve the goals of the *SMARTROAD* system. The conducted experiments show the efficiency and effectiveness of the recommendation policies proposed. QoS Prediction wise, we achieved an improvement of 30% on MAE over existing methods. Task recommendation wise, the proposed algorithm produces the expected results when applied to synthetic data.

6.5 Discussion

One of the strengths of the AS3 method is that it allows to bridge between current advances in AI techniques, such as those used in recommender systems and optimization, with current challenges in software engineering and service computing. This is a highly anticipated and novel contribution that characterizes AS3. This is especially true in the case of using recommendation techniques to enhance and enrich the design of SS and their usage in enabling the vision of System of Systems (SoS) paradigm.

On the recommendation tools developed in this chapter, while U-AutoRec already outperforms the other methods in most most cases, there is still room for improvement. Indeed, with the huge number of combination that can be used to finetune the model (i.e., number of iterations, optimizers, number of hidden nodes, activation functions, etc.), it is most likely that our experiment did not reach the limits of the AutoRec framework. Extensive experiments and finetuning techniques are needed in this direction to identify the best parameters for the model.

We have also ran experiments using I-AutoRec, which follows the same model as U-AutoRec with the difference that it relies on the items representations rather than the users representations to build the model. I-AutoRec revealed some interesting results. I-AutoRec performs generally better on throughput data, reaching an $MAE = 14.63$ and an $RMSE = 38.26$, when $MD = 10\%$; and an $MAE = 12.55$ and an $RMSE = 34.77$, when $MD = 20\%$. However, U-AutoRec still outperforms I-AutoRec in both response time (RT) and throughput (TP) data. This leads us to believe that response time data is more sensitive to the features of the users, while throughput data is also sensitive to the features of the services. This is logical as response time would change depending on the network state of the users (e.g. signal strength), while throughput depends on the network state of the service's supporting infrastructure and that of the user.

While, the recommendation algorithm developed for improving the service models of the *SMARTROAD* SS performed as expected on synthetic data, it does not prove its ability to generalize over real data collected from systems that are span long running periods. This is because those systems usually don't have such a clear cut classification of the service's task. Indeed, services are usually described using API or service descriptions that expressed in free text, making it harder to pinpoint the exact task performed by the service. More investigation of service description languages is needed to make the recommendation tools developed within this thesis applicable to real settings.

Chapter 7

Conclusions and Perspectives

The true function of philosophy is to educate us in the principles of reasoning and not to put an end to further reasoning by the introduction of fixed conclusions

George Henry Lewes

During this PhD project, we had the opportunity to touch and work on several scientific, technical and technological aspects directly impacting the design and development of Smart System (SS)s. In this last and concluding chapter, we summarize and discuss the research work conducted during this PhD project and how it relates to the research questions stated in the introduction chapter (see section 1.2). We also discuss the shortcomings of our contributions and the challenges we faced in regards to answering these mentioned research questions and the development of our research. Last, we draw up some perspectives following the different aspects of our work.

7.1 Conclusions

The evolution of systems and technology over time has given rise to a new generation of systems that are capable of leveraging data to offer pertinent information and services to users in almost all aspects of life, from domotics to work to entertainment. This evolution has mainly been enabled by the advances in wireless communication, electronics and Artificial Intelligence (AI), giving rise to new paradigms such as the Internet of Things (IoT), Cloud Computing (CC), Big Data.

However, the majority of the research efforts so far have been focused on data (Lim and Maglio, 2019). Very little works have discussed a general approach to the design of the systems operating in smart environments, as most existing systems tend to be application-specific and developed end-to-end in an *ad hoc* manner. This is surprising as one of the most important promises of the 'smartification' paradigm is *connectedness* and the only way to achieve it is through a general view of what a smart system is and should be.

This previous finding has led us to formulate reflect on the manner systems are designed and developed and formulate our general and guiding research as *How can we design and build systems and software solutions that are suitable for smart environments ?*. To answer this research question, we've conducted an extensive review of the literature on the subject and developed several frameworks to frame the discussion on the different facets of SS. Specifically, we differentiated between design concerns, technology concerns and improvement concerns and studied if and how different related works have dealt with these concerns in building SSs. The results of this analysis have provided the basis for our contributions later on. Despite the fact that these facets are connected in a lot of ways as design impacts technology and improvement and vice versa, each of these facets plays a different but important role in building SSs.

In the following subsections, we'll briefly talk about these different aspects as we make the connection to the contributions that were made in this PhD and how they address the research questions stated at the introduction of this thesis. After that, we'll present some of the limitations that we were able to discern about our contributions, before talking about some of the challenges that have impeded the realization of some steps that were initially planned in the context of this thesis.

7.1.1 Contribution Summary

As the reader might have noticed already, some of the contributions we presented throughout this thesis have come early on in the form of frameworks that were supposed to frame the discussion on the identified aspects of SSs.

On the design aspect, we've introduced the *PeRMI* as a helpful tool to analyze the capabilities of SSs. This framework presents the advantage of being holistic as is covers different aspects of an SS from the way it collects data and processes it to form the system's perception of its world, to the way it responds to particular events and situations and interacts with the end user. It is also extensible and flexible because the elements of interest in each capability can differ based on the application's functional and non-functional parameters. This has allowed us to analyze different related works on SSs, despite the big differences in their targeted

application domain and design considerations.

On the technological aspect, we have identified three technology stacks that have had a major impact on SSs, namely, CC, IoT and Big Data. However, while these technologies are prominent, we have noticed that their usage in SSs remains nonuniform and highly erratic. To homogenize the use of these technologies and guide their usage in building SS, we proposed the *C2IoT* framework. We argue that this framework can be a useful tool to normalize the use of enabling technologies for SS development as it contains the elements of the three technologies. The use of context information to allow the collaboration between the different elements of these enablers also allows the evaluation of their integration, which is also an interest to many researchers.

On the continuous improvement aspect of SSs, we have identified and analyzed improvement possibilities at both design and runtime that could be of potential benefit to SSs. We've discussed how these improvements could target the design aspects using the *PeRMI* framework, as well as the technological aspects using the *C2IoT* framework. This analysis has led us to start thinking about the possible answers to ***RQ4***, especially through the consideration of the different participants in the design and the development of SSs and the levels of involvement that they wish to keep in the evolution of these SSs. This calls for a way that can inform the users of the possible improvement but leaves the final decision of implementing (or not implementing) the improvement to user, and at the same time can be easily automated in case the user does not need to make the final decision.

To answer ***RQ1***, we developed the *Smart System Loop* as a life cycle that encompasses the building blocks of any SS. The proposed *Smart System Loop* integrates views and elements from four (4) different perspectives. Each of the perspectives involved in its development was analyzed to some extent in the first part of this thesis. The basis and the first perspective of the *Smart System Loop* is the *PeRMI* framework. The *Smart System Loop* implements the core principals of the *PeRMI* framework expressed through the *Perception*, *Response* and *Manual Intervention* capabilities. To concretize this implementation, we use concepts that are predominant in two design methods, namely, context-aware engineering and service-based engineering as the second and third perspectives involved in the *Smart System Loop*. The concepts of context-aware engineering, namely, the *Context*, the *Situation* and the *User* fit perfectly with the focus on the *Perception* and *Manual Intervention* capabilities of the *PeRMI* framework. In the meantime, the concepts of service-based engineering, namely, the *Service*, the *Goal* and the *User* implement the *Response* and *Manual Intervention* capabilities of the *PeRMI* framework.

To explain how these concepts relate to each other and answer ***RQ2***, a fourth perspective is introduced, which is partly based on the adaptability loop (or SIDA loop). The relations and operations introduced within the adaptability loop (Sens-Interpret-Decide-Act) fit perfectly with the relationships that exist between the concepts of *Service -> Context -> Situation -> Goal -> Service*. These relationships allows the software architect to understand how the concepts of an SS depend on each other and work in harmony to insure the adaptability of the SS. To make the *Smart System Loop* complete and set the field for answering ***RQ4***, we've also defined the relationships between the *User* and the four concepts. The *User* concept was introduced to represent both the final-user and the SS's architect depending on the nature of the operations getting carried out. Indeed, while the intervention of the final-user is recorded to personalize and customize the perception and the response capabilities of the system for that specific final-user, the intervention of the architect are used to improve the performance of the system at a global stage.

Based on the *Smart System Loop*, and to answer our general ***RQ*** and ultimately ***RQ4***, we have proposed the AS3 method as the guiding hand towards the design and the development of smart systems. The proposed AS3 method was designed to allow the method user to think about systems as continuously improvable. To that end, we defined intentional process models that would help track the intention that SS engineers should consider while defining their targeted SS. The proposed process model draws its expressiveness power from the MAP formalism and allows for different levels of abstraction and refinement. In the mean time, we've also designed a product metamodel for SSs based on the concepts and relationships defined within the *Smart System Loop*. The AS3 process model works in harmony with the concepts defined in the metamodel for SS to design and eventually build the targeted SS. Indeed, each *intention* in the process model is achieved through at least one strategy that requires the instantiation of one or multiple concepts with regard to the application domain of the targeted SS. At the highest level of abstraction, the process model can be considered as a cycle of definition-improvement intentions and highlights the continuous improvement characteristic which is inherent to SSs.

Starting from this high level abstraction process model and its corresponding product metamodel, we started decomposing and refining each section of the process model into separate fragments. The resulting fragments provide an extensive guide to design and develop SSs from high level requirements to systems that are built to improve through the introduction of the *Recommendation* concept. The AS3 method starts by an identification intention in which the method

user in guided to identify and define the instances of the concepts depicted in the corresponding product metamodel regarding the targeted SS, using the requirements and domain knowledge. The relationships between the concepts can then be leveraged to check for missing entities and then be leveraged to define how each instance relates to another with regards to the targeted SS. To goal here is to arrive at one or several complete instances of the *Smart System Loop*. Once a first design of the system is built, the method user is encouraged through the AS3 method to start thinking about setting up a strategy for how the targeted SS needs to evolve and improve. At this level, the improvement process can target the design of the SS and thus trigger its remodeling, or it can target the user to trigger some personalization actions. The runtime performance of the system can also be targeted on the long run through optimization by searching for the best configuration of SS.

To showcase how the AS3 method can be used to design and develop a targeted SS, we've used the *SMARTROAD* case study as a basis to build a *SMARTROAD* SS. We've identified and proposed a *Happy Path* representing the sections of the process model of the AS3 method that we were going to focus on. The identified path covers the most crucial elements of the AS3 method. Hence, we have followed the different intentions presented in the *Happy Path* of the AS3 method to identify the elements and design the *SMARTROAD* SS. This design was used to develop a simulated version of the system and to retrieve the execution traces of the system, thus making a log. This log was then used as a basis to improve two aspects of the *SMARTROAD* system, namely the response time and throughput as *QoS* properties and the predefined *Abstract Service Models* through the use of two recommendation techniques that we proposed to that end.

7.1.2 Limitations and challenges

Throughout the development and the writing of this thesis, we had some time and opportunity to reflect back on some of the design principle and decisions that went into developing our contributions. This reflection has led us to identify some potential limitations and threats to the validity of the conducted research. These limitations and challenges can be summarized as follows:

- The fact that *C2IoT* is based on CC and its service models makes it hard to evaluate solutions that do not explicitly use CC in their architecture (from a *Layer* perspective). However, this can be remedied by a finer analysis at the *View* perspective through the specification of which components of the IoT and/or the Big Data stacks are used in the solution.

- The development of the simulation of the *SMARTROAD* SS showcased the usefulness and relevance of the AS3 method. However, it is not sufficient to prove that its use is beneficial to system engineers and architects. Also, the system did not have the ability to improve over time and thus we were not able to collect real data of its usage based on real user experience. This had limited our ability to investigate and prove how adaptable systems built using the AS3 method can be. Hence, there is a need for a real-world case study and feedback from engineers on the evaluation of the method's usefulness.
- There is a lack of datasets that cater to composition of services with consideration to QoS attributes. To date, there exists no dataset or log files recording the behavior of composite services. The Service Computing community could highly benefit from such data. The data could be used to explore the effect of the composition on the selection algorithm. Meaning that considering two services s_1 and s_2 offering the same task t_1, where s_1 is better than s_2 and two other services s_3 and s_4 offering the same task t_2, where s_3 is better than s_4; would the composition of $s_1 \rightarrow s_3$ be better than the composition of $s_2 \rightarrow s_3$? and if not, then the research should maybe focus on predicting QoS over service patterns rather than individual services.

7.2 Perspectives

As we've stated in the limitations section earlier, more application domains need to be targeted to validate the generality of the proposed approach to build and operate smart environments. To that end, we are working on the design of a smart home solution through the application of the AS3 approach. While the smart home use case provides a smaller scope for the use AS3 as a method that initially targets System of Systems (SoS)s, the implementation and experimentation procedures are less complicated and thus provide a more accessible and controlled setting to conduct thorough experiments, especially on the ability of the developed system to improve according to end-user's behavior.

In this thesis, we've presented improvement as both a design and run-time process. While, we've developed two tools that can be used to improve SSs as part of the AS3 method, more specific tools to improve and optimize different aspects of SS are required to provide complete tool support for the AS3 method. Hence, we are working on developing more tools and techniques to cover the whole scope of the possible improvements discussed in section 6.1. Particularly:

- As exemplified in the improvement of *Quality of Service* properties of the SS (i.e., response time and throughput), recommender systems can be used to

improve the functional properties of SSs according to the preferences and behaviors of the end users. We argue that the same logic can be used to improve the other reasoning elements identified in the initial SS design (i.e., Function, Scope, Task, Quality of Context). This can be mainly achieved through the creation of new or modified instances of these concepts by analyzing the SS's execution data collected at runtime.

- In the same way and demonstrated in the improvement of the *Goal*, recommender systems can be used to improve the design of SSs. We argue that the same logic can be used to improve the other analysis elements identified in the initial SS design (i.e., Context, Situation, User, Service). This can be mainly achieved through the discovery of new instances of these concepts by analyzing the SS's execution data collected at runtime and the created/-modified instances of the reasoning elements.

Aside from connectedness, another important characteristic of the smartification paradigm is sustainability. Indeed, smart systems are supposed to alleviate and mitigate the resource scarceness problem by proactively dedicating just the necessary amount of resources to achieve or perform a task. In this direction, two research avenues are of particular interest to us.

1. First, it is worth investigating the relationships between (i) the resources implicated in a service's task, (ii) the quality of the service it exposes and (iii) the quantity of demand on the service. A clear correlation between these factors would allow the optimization of the resources according to the service task taking into account the demand at different contexts.

2. Second, as we have identified key technology enablers for smart systems (i.e., Cloud Computing, Internet of Things and Big Data) in the C2IoT framework (Faieq, Saidi, et al., 2017b), we aim at investigating the possibility of facilitating the configuration and deployment of the resulting systems over technological platforms (e.g., Containers like Docker, Single Board Computers like Raspberry Pi, etc.). This is particularly important as it allows to consider the resources used by the system as finite resources forcing the innovative efforts to be focused on the logic (algorithmic) side of the systems rather than on their hardware (computing) infrastructure.

Deep Learning frameworks have shown their efficiency and effectiveness on many application domains. The application of deep learning algorithms in Recommender Systems is supposed to enhance their performance even further (S. Zhang et al., 2019). Research has shown the improvements in performance when

applied to standard datasets such as MovieLens. Hence, it would be beneficial to explore the use of such frameworks to build new models for QoS prediction for Service Recommendation, as they can potentially provide better performances.

Another research direction that is worth investigating in recommender systems in general, and in service recommendation in particular, is the impact of these systems on the performance of the recommended services. Indeed, recommending the same service to a myriad of users could potentially be dangerous, as the service could be overloaded with multiple requests at the same time, which we assume could result in a serious drop in its QoS. Having the knowledge about the computing environment of the services, could help investigate how many requests can a service handle before it suffers QoS drops. New recommender systems would need to take this information into account in the selection process to avoid negative effects on the service.

Service composition is another problem that can benefit greatly from the advances in the field of AI. Many approaches have been adopted and/or adapted to solve the service composition problem. Today, and despite the advances in technology, static and manual approaches are still the ones in use in companies, especially for the execution of part or all of their business processes. This is despite many attempts to address this problem coming from artificial intelligence such as SHOP2 (D. Wu et al., 2003) or the semantic web such as OVWSC (Slaimi et al., 2014). These efforts assume that we already know the existing relationships between services either through their tasks or their interfaces (i.e., their signatures). These relationships are specified at design time by experts through specific structures (e.g., ontologies, hierarchical task networks, etc.). This limitation makes the solution space to be explored remain stable, which does not reflect the user's preferences, needs and innovations. Using and adapting Machine Learning (ML)-based approaches like Sequence-aware Recommenders (Quadrana et al., 2018) to solve the composition problem in an open world where the relationships between services are not known a priori as in the case of smart environments (e.g., smart roads, smart homes, etc.) is certainly worth exploring.

Another research direction that we are interested in concerns 'green design', which focuses on the consideration of computing resources in the design and development of smart systems. The objective here is to minimize the computing resources used for the implementation of a SS without compromising the quality of the solution. This is applicable at three levels (1) energy consumption related to data (storage), (2) energy consumption related to processing and (3) energy consumption related to communications (networks). A green design approach will thus have to perform a strong coupling between data models and behavior

models with resource models. Thus, a trade-off must be made with respect to the choice of the data to store, process and communicate, the algorithms to use and the communication medium against the available computing resources. We argue that can lead to a greener and more sustainable computing landscape and mitigate some of the energy consumption trends mentioned in the seminal work of Andy Hooper (Hooper, 2008).

Bibliography

Abdelhafidh, M., M. Fourati, L. C. Fourati, A. B. Mnaouer, and Z. Mokhtar (2018). "Cognitive internet of things for smart water pipeline monitoring system." In: *2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE, pp. 1–8. DOI: 10.1109/DISTRA.2018.8600999.

Abowd, G. D., A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles (1999). "Towards a Better Understanding of Context and Context-Awareness." In: *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*. HUC '99. Karlsruhe, Germany: Springer-Verlag, pp. 304–307. ISBN: 3540665501.

Ait-Cheik-Bihi, W., A. Nait-Sidi-Moh, M. Bakhouya, J. Gaber, and M. Wack (Dec. 2012). "TransportML platform for collaborative location-based services." In: *Service Oriented Computing and Applications* 6.4, pp. 363–378. ISSN: 1863-2394. DOI: 10.1007/s11761-012-0114-2.

Alansari, Z., N. B. Anuar, A. Kamsin, S. Soomro, M. R. Belgaum, M. H. Miraz, and J. Alshaer (2018). "Challenges of internet of things and big data integration." In: *International Conference for Emerging Technologies in Computing*. Springer, pp. 47–55. DOI: 10.1007/978-3-319-95450-9_4.

Andreesseen, M. (2011). "Why Software Is Eating The World." In: *Wall Street Journal* 8.20. URL: https://a16z.com/2011/08/20/why-software-is-eating-the-world/.

Angelopoulos, C. M., S. Nikoletseas, and G. C. Theofanopoulos (2011). "A smart system for garden watering using wireless sensor networks." In: *Proceedings of the 9th ACM international symposium on Mobility management and wireless access*, pp. 167–170. DOI: 10.1145/2069131.2069162.

Armando, N., A. Rodrigues, V. Pereira, J. S. Silva, and F. Boavida (Aug. 2018). "An Outlook on Physical and Virtual Sensors for a Socially Interactive Internet." In: *Sensors* 18.8, p. 2578. ISSN: 1424-8220. DOI: 10.3390/s18082578.

Arnold, C., D. Kiel, and K.-I. Voigt (Dec. 2016). "How The Industrial Internet Of Things Changes Business Models In Different Manufacturing Industries." In: *International Journal of Innovation Management* 20.08, p. 1640015. DOI: 10.1142/S1363919616400156.

Atzori, L., A. Iera, and G. Morabito (2010). "The Internet of Things: A survey." In: *Computer Networks* 54.15, pp. 2787–2805. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2010.05.010.

Augusto, J. C., V. Callaghan, D. Cook, A. Kameas, and I. Satoh (2013). "Intelligent environments: a manifesto." In: *Human-centric Computing and Information Sciences* 3.1, pp. 1–18.

Ballagny, C., N. Hameurlain, and F. Barbier (2009). "Mocas: A state-based component model for self-adaptation." In: *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, pp. 206–215. DOI: 10.1109/SASO.2009.11.

Baresi, L., E. Di Nitto, C. Ghezzi, and S. Guinea (Apr. 2007). "A framework for the deployment of adaptable web service compositions." In: *Service Oriented Computing and Applications* 1.1, pp. 75–91. ISSN: 1863-2394. DOI: 10.1007/s11761-007-0004-1.

Barsalou, L. W. (1982). "Context-independent and context-dependent information in concepts." In: *Memory & cognition* 10.1, pp. 82–93.

Baumgartner, N., W. Gottesheim, S. Mitsch, W. Retschitzegger, and W. Schwinger (2010). "Improving situation awareness in traffic management." In: *Proc. Intl. Conf. on Very Large Data Bases*.

Benkahla, N., H. Tounsi, S. Ye-Qiong, and M. Frikha (2019). "Enhanced ADR for LoRaWAN networks with mobility." In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, pp. 1–6. DOI: 10.1109/IWCMC.2019.8766738.

Bieberstein, N., S. Bose, L. Walker, and A. Lynch (Oct. 2005). "Impact of Service-oriented Architecture on Enterprise Systems, Organizational Structures, and Individuals." In: *IBM Syst. J.* 44.4, pp. 691–708. ISSN: 0018-8670. DOI: 10.1147/sj.444.0691.

Biswas, P., P. Langdon, J. Umadikar, S. Kittusami, and S. Prashant (2014). "How interface adaptation for physical impairment can help able bodied users in situational impairment." In: *Inclusive Designing*. Springer, pp. 49–58. DOI: 10.1007/978-3-319-05095-9_5.

Bleiholder, J. and F. Naumann (Jan. 2009). "Data Fusion." In: *ACM Comput. Surv.* 41.1. ISSN: 0360-0300. DOI: 10.1145/1456650.1456651.

Bosch, J. (1999). "Superimposition: A component adaptation technique." In: *Information and software technology* 41.5, pp. 257–273. DOI: 10.1016/S0950-5849(99)00007-5.

Botta, A., W. De Donato, V. Persico, and A. Pescapé (2016). "Integration of cloud computing and internet of things: a survey." In: *Future generation computer systems* 56, pp. 684–700. DOI: 10.1016/j.future.2015.09.021.

Bouguettaya, A., M. Singh, M. Huhns, Q. Z. Sheng, H. Dong, Q. Yu, A. G. Neiat, S. Mistry, B. Benatallah, B. Medjahed, M. Ouzzani, F. Casati, X. Liu, H. Wang, D. Georgakopoulos, L. Chen, S. Nepal, Z. Malik, A. Erradi, Y. Wang, B. Blake, S. Dustdar, F. Leymann, and M. Papazoglou (Mar. 2017). "A Service Computing Manifesto: The Next 10 Years." In: *Commun. ACM* 60.4, pp. 64–72. ISSN: 0001-0782. DOI: 10.1145/2983528.

Bourque, L. B. and V. A. Clark (1992). *Processing data: The survey example*. 85. Sage Publications, Inc. ISBN: 9780585217079.

Boyd, J. (1995). "OODA loop." In: *Center for Defense Information, Tech. Rep*.

Brdiczka, O., J. L. Crowley, and P. Reignier (2008). "Learning situation models in a smart home." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.1, pp. 56–63. DOI: 10.1109/TSMCB.2008.923526.

Cabrera, C. and S. Clarke (2019). "A Self-Adaptive Service Discovery Model for Smart Cities." In: *IEEE Transactions on Services Computing*. DOI: 10.1109/TSC.2019.2944356.

Candillier, L., F. Meyer, and M. Boullé (2007). "Comparing State-of-the-Art Collaborative Filtering Systems." In: *Machine Learning and Data Mining in Pattern Recognition*. Ed. by P. Perner. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 548–562.

Card, S. K., A. Newell, and T. P. Moran (1983). *The Psychology of Human-Computer Interaction*. USA: L. Erlbaum Associates Inc. ISBN: 0898592437.

Cardellini, V., E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola (2011). "Moses: A framework for qos driven runtime adaptation of service-oriented systems." In: *IEEE Transactions on Software Engineering* 38.5, pp. 1138–1159. DOI: 10.1109/TSE.2011.68.

Casati, F. and M.-C. Shan (2001). "Dynamic and adaptive composition of e-services." In: *Information Systems* 26.3. 12th International Conference on Advanced Systems Engineering, pp. 143–163. ISSN: 0306-4379. DOI: 10.1016/S0306-4379(01)00014-X.

Cela, O., M. Cortes-Cornax, A. Front, and D. Rieu (2019). "Methodological Framework to Guide the Development of Continual Evolution Methods." In: *Advanced Information Systems Engineering*. Ed. by P. Giorgini and B. Weber. Cham: Springer International Publishing, pp. 48–63. ISBN: 978-3-030-21290-2. DOI: 10.1007/978-3-030-21290-2_4.

Chang, C. K., H.-y. Jiang, H. Ming, and K. Oyama (2009). "Situ: A situation-theoretic approach to context-aware service evolution." In: *IEEE Transactions on Services Computing* 2.3, pp. 261–275. DOI: 10.1109/TSC.2009.21.

Chang, J., W. Yao, and X. Li (2017). "The design of a context-aware service system in intelligent transportation system." In: *International Journal of Distributed Sensor Networks* 13.10, p. 1550147717738165. DOI: 10.1177/1550147717738165.

Chen, M., Y. Ma, J. Song, C.-F. Lai, and B. Hu (2016). "Smart clothing: Connecting human with clouds and big data for sustainable health monitoring." In: *Mobile Networks and Applications* 21.5, pp. 825–845. DOI: 10.1007/s11036-016-0745-1.

Chen, X., X. Liu, Z. Huang, and H. Sun (July 2010). "RegionKNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation." In: *2010 IEEE International Conference on Web Services*, pp. 9–16. DOI: 10.1109/ICWS.2010.27.

Chen, X., L. Wang, J. Ding, and N. Thomas (2016). "Patient Flow Scheduling and Capacity Planning in a Smart Hospital Environment." In: *IEEE Access* 4, pp. 135–148. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2015.2509013.

Chen, X., Z. Zheng, Q. Yu, and M. R. Lyu (July 2014). "Web Service Recommendation via Exploiting Location and QoS Information." In: *IEEE Transactions on Parallel and Distributed Systems* 25.7, pp. 1913–1924. ISSN: 1045-9219. DOI: 10.1109/TPDS.2013.308.

Christensen, I. A. and S. Schiaffino (2011). "Entertainment recommender systems for group of users." In: *Expert Systems with Applications* 38.11, pp. 14127–14135. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.04.221.

Cicirelli, F., A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, A. Iera, and G. Ruggeri (Aug. 2018). "Edge Computing and Social Internet of Things for Large-Scale Smart Environments Development." In: *IEEE Internet of Things Journal* 5.4, pp. 2557–2571. ISSN: 2372-2541. DOI: 10.1109/JIOT.2017.2775739.

Cook, D. J. and S. K. Das (2007). "How smart are our environments? An updated look at the state of the art." In: *Pervasive and Mobile Computing* 3.2. Design and Use of Smart Environments, pp. 53–73. ISSN: 1574-1192. DOI: https://doi.org/10.1016/j.pmcj.2006.12.001.

Coutaz, J., J. L. Crowley, S. Dobson, and D. Garlan (Mar. 2005). "Context is Key." In: *Commun. ACM* 48.3, pp. 49–53. ISSN: 0001-0782. DOI: 10.1145/1047671.1047703.

Cubo, J., N. Gamez, L. Fuentes, and E. Pimentel (2013). "Composition and self-adaptation of service-based systems with feature models." In: *International Conference on Software Reuse*. Springer, pp. 326–342. DOI: 10.1007/978-3-642-38977-1_25.

Cubo, J. and E. Pimentel (2011). "DAMASCo: A framework for the automatic composition of component-based and service-oriented architectures." In: *European Conference on Software Architecture*. Springer, pp. 388–404. DOI: 10.1007/978-3-642-23798-0_41.

David, W. (2011). "Smart cities, smart places, smart democracy: Form-based codes, electronic governance and the role of place in making smart cities." In: *Intelligent Buildings International* 3.3, pp. 198–218. DOI: 10.1080/17508975.2011.586670.

Demirkan, H. (Sept. 2013). "A Smart Healthcare Systems Framework." In: *IT Professional* 15.5, pp. 38–45. ISSN: 1520-9202. DOI: 10.1109/MITP.2013.35.

Derdour, M., P. Roose, M. Dalmau, N. G. Zine, and A. Alti (2010). "An adaptation approach for component-based software architecture." In: *2010 IEEE 34th Annual Computer Software and Applications Conference*. IEEE, pp. 179–187. DOI: 10.1109/COMPSAC.2010.24.

Dijkstra, E. W. (1982). "On the role of scientific thought." In: *Selected writings on computing: a personal perspective*. Springer, pp. 60–66.

Djemame, K., R. Bosch, R. Kavanagh, P. Alvarez, J. Ejarque, J. Guitart, and L. Blasi (2017). "PaaS-IaaS inter-layer adaptation in an energy-aware cloud environment." In: *IEEE Transactions on Sustainable Computing* 2.2, pp. 127–139. DOI: 10.1109/TSUSC.2017.2719159.

Dumas, B., B. Signer, and D. Lalanne (2012). "Fusion in multimodal interactive systems: an HMM-based algorithm for user-induced adaptation." In: *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, pp. 15–24. DOI: 10.1145/2305484.2305490.

Durães, D., D. Carneiro, J. Bajo, and P. Novais (2018). "Modelling a smart environment for nonintrusive analysis of attention in the workplace." In: *Expert Systems* 35.5, e12275. DOI: 10.1111/exsy.12275.

Erl, T. (2009). *SOA Design Patterns*. 1st. USA: Prentice Hall PTR. ISBN: 0136135161.

Esfahani, N., E. Yuan, K. R. Canavera, and S. Malek (Feb. 2016). "Inferring Software Component Interaction Dependencies for Adaptation Support." In: *ACM Trans. Auton. Adapt. Syst.* 10.4. ISSN: 1556-4665. DOI: 10.1145/2856035.

Faieq, S., A. Front, R. Saidi, H. El Ghazi, and M. D. Rahmani (Oct. 2019). "A context-aware recommendation-based system for service composition in smart environments." In: *Service Oriented Computing and Applications* 13.4, pp. 341–355. ISSN: 1863-2394. DOI: 10.1007/s11761-019-00277-7.

Faieq, S., R. Saidi, H. Elghazi, and M. D. Rahmani (2017a). "A Conceptual Architecture for a Cloud-Based Context-Aware Service Composition." In: *Advances in Ubiquitous Networking 2*. Singapore: Springer Singapore, pp. 235–246.

— (2017b). "C2IoT: A framework for Cloud-based Context-aware Internet of Things services for smart cities." In: *Procedia Computer Science* 110, pp. 151–158. ISSN: 1877-0509. DOI: 10.1016/j.procs.2017.06.072.

Fang, J., S. Hu, and Y. Han (Sept. 2004). "A service interoperability assessment model for service composition." In: *IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004*, pp. 153–158. DOI: 10.1109/SCC.2004.1358002.

Force, I. R. S.
bibinitperiod C. M. T. (2014). *An Overview Report on the Current Status and Implications of Road Safety & Connected Mobility*. Tech. rep. IRU, Michelin Group, pp. 15–16.

Foundation, N. S. (2014). "Partnerships for Innovation: Building Innovation Capacity (PFI: BIC)." In: *Program Solicitation*, pp. 1–16. URL: https://www.nsf.gov/pubs/2014/nsf14610/nsf14610.pdf.

Garcin, F., C. Dimitrakakis, and B. Faltings (2013). "Personalized News Recommendation with Context Trees." In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys '13. Hong Kong, China: ACM, pp. 105–112. ISBN: 978-1-4503-2409-0. DOI: 10.1145/2507157.2507166.

Garg, K. and D. Kumar (2013). "Comparing the performance of frequent pattern mining algorithms." In: *International Journal of Computer Applications* 69.25.

Gasson, S. (Mar. 1999). "A Social Action Model of Situated Information Systems Design." In: *SIGMIS Database* 30.2, pp. 82–97. ISSN: 0095-0033. DOI: 10.1145/383371.383377.

Grady, B., A. M. Robert, W. E. Michael, J. Y. Bobbi, C. Jim, and A. H. Kelli (2007). *Object-oriented analysis and design with applications*. 3rd ed. The Addison-Wesley object technology series. Addison-Wesley. ISBN: 9780201895513.

Gu, Q. and P. Lago (Sept. 2009). "Exploring service-oriented system engineering challenges: a systematic literature review." In: *Service Oriented Computing and Applications* 3.3, pp. 171–188. ISSN: 1863-2394. DOI: 10.1007/s11761-009-0046-7.

Gu, R., C. Li, P. Shu, C. Yuan, and Y. Huang (2019). "Adaptive cache policy scheduling for big data applications on distributed tiered storage system." In: *Concurrency and Computation: Practice and Experience* 31.15. DOI: 10.1002/cpe.5138.

Gudivada, V. N., D. Rao, and V. V. Raghavan (2014). "NoSQL systems for big data management." In: *2014 IEEE World congress on services*. IEEE, pp. 190–197.

Gullà, F., S. Ceccacci, R. Menghi, and M. Germani (2016). "An adaptive smart system to foster disabled and elderly people in kitchen-related task." In: *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pp. 1–4. DOI: 10.1145/2910674.2910678.

Haeckel, S. H. (1999). *Adaptive enterprise: Creating and leading sense-and-respond organizations*. Harvard business press.

Hall, W. and T. Tiropanis (2012). "Web evolution and Web Science." In: *Computer Networks* 56.18. The WEB we live in, pp. 3859–3865. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2012.10.004.

Han, J., J. Pei, and Y. Yin (2000a). "Mining Frequent Patterns Without Candidate Generation." In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: ACM, pp. 1–12. ISBN: 1-58113-217-4. DOI: 10.1145/342009.335372.

— (2000b). "Mining frequent patterns without candidate generation." In: *ACM sigmod record* 29.2, pp. 1–12.

Hashem, I. A. T., V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma (2016). "The role of big data in smart city." In: *International Journal of Information Management* 36.5, pp. 748–758. ISSN: 0268-4012. DOI: https://doi.org/10.1016/j.ijinfomgt.2016.05.002.

Hashem, I. A. T., I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan (2015). "The rise of "big data" on cloud computing: Review and open research issues." In: *Information systems* 47, pp. 98–115. DOI: 10.1016/j.is.2014.07.006.

He, P., J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu (June 2014). "Location-Based Hierarchical Matrix Factorization for Web Service Recommendation." In: *2014 IEEE International Conference on Web Services*, pp. 297–304. DOI: 10.1109/ICWS.2014.51.

Hennebert, C. and J. D. Santos (2014). "Security Protocols and Privacy Issues into 6LoWPAN Stack: A Synthesis." In: *IEEE Internet of Things Journal* 1.5, pp. 384–398.

Holland, J. H. (2006). "Studying complex adaptive systems." In: *Journal of systems science and complexity* 19.1, pp. 1–8.

Hooper, A. (2008). "Green computing." In: *Communication of the ACM* 51.10, pp. 11–13.

Jara, A. J., Y. Sun, H. Song, R. Bie, D. Genooud, and Y. Bocchi (Mar. 2015). "Internet of Things for Cultural Heritage of Smart Cities and Smart Regions." In: *2015 IEEE 29th International*

Conference on Advanced Information Networking and Applications Workshops, pp. 668–675. DOI: 10.1109/WAINA.2015.169.

Jin, J., X. Ma, and I. Kosonen (2017). "A stochastic optimization framework for road traffic controls based on evolutionary algorithms and traffic simulation." In: *Advances in Engineering Software* 114, pp. 348–360.

JTC-1, I. (2014). *Smart cities. Preliminary Report*. Tech. rep. ISO/IEC JTC 1 — Information Technology. URL: https://www.iso.org/files/live/sites/isoorg/files/developing_standards/docs/en/smart_cities_report-jtc1.pdf.

Jung, G., M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu (2010). "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures." In: *2010 IEEE 30th International Conference on Distributed Computing Systems*. IEEE, pp. 62–73. DOI: 10.1109/ICDCS.2010.88.

Kabir, M. H., M. R. Hoque, H. Seo, and S.-H. Yang (2015). "Machine learning based adaptive context-aware system for smart home environment." In: *International Journal of Smart Home* 9.11, pp. 55–62. DOI: 10.14257/ijsh.2015.9.11.07.

Kantere, V. and M. Filatov (2015). "A Workflow Model for Adaptive Analytics on Big Data." In: *2015 IEEE International Congress on Big Data*. IEEE, pp. 673–676. DOI: 10.1109/BigDataCongress.2015.106.

Karat, J. (July 1997). "Evolving the Scope of User-Centered Design." In: *Commun. ACM* 40.7, pp. 33–38. ISSN: 0001-0782. DOI: 10.1145/256175.256181.

Kazhamiakin, R., S. Benbernou, L. Baresi, P. Plebani, M. Uhlig, and O. Barais (2010). "Adaptation of service-based systems." In: *Service research challenges and solutions for the future internet*. Springer, pp. 117–156. DOI: 10.1007/978-3-642-17599-2_5.

Kephart, J. O. and D. M. Chess (2003a). "The vision of autonomic computing." In: *Computer* 36.1, pp. 41–50. DOI: 10.1109/MC.2003.1160055.

— (Jan. 2003b). "The Vision of Autonomic Computing." In: *Computer* 36.1, pp. 41–50. ISSN: 0018-9162. DOI: 10.1109/MC.2003.1160055.

Khaddam, I., N. Mezhoudi, and J. Vanderdonckt (2015). "Adapt-first: A MDE transformation approach for supporting user interface adaptation." In: *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*. IEEE, pp. 1–9. DOI: 10.1109/WSWAN.2015.7209080.

Khan, Z., A. Anjum, and S. L. Kiani (2013). "Cloud based big data analytics for smart future cities." In: *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE, pp. 381–386.

Khatoun, R. and S. Zeadally (July 2016). "Smart Cities: Concepts, Architectures, Research Opportunities." In: *Commun. ACM* 59.8, pp. 46–57. ISSN: 0001-0782. DOI: 10.1145/2858789.

Kling, R. (1977). "The Organizational Context of User-Centered Software Designs." In: *MIS Quarterly* 1.4, pp. 41–52. ISSN: 02767783. URL: http://www.jstor.org/stable/249021.

Laplante, P. A., M. Kassab, N. L. Laplante, and J. M. Voas (2017). "Building caring healthcare systems in the Internet of Things." In: *IEEE Systems Journal* 12.3, pp. 3030–3037.

Lee, H. and J. Lee (2018). "Development Concepts of Smart Service System-based Smart Factory (4SF)." In: *INCOSE International Symposium* 28.1, pp. 1153–1169. DOI: 10.1002/j.2334-5837.2018.00540.x.

Lee, K., K. C. Kang, and J. Lee (2002). "Concepts and guidelines of feature modeling for product line software engineering." In: *International Conference on Software Reuse*. Springer, pp. 62–77. DOI: 10.1007/3-540-46020-9_5.

Legner, C. and R. Heutschi (June 2007). "SOA Adoption in Practice - Findings from Early SOA Implementations." In: *Relevant rigour - rigorous relevance : 15th European Conference on Information Systems ; ECIS 2007*. Ed. by H. Österle, J. Schelp, and R. Winter. AIS Electronic Library (AISeL): Association for Information Systems, pp. 1643–1654.

Levina, A. I., A. S. Dubgorn, and O. Y. Iliashenko (2017). "Internet of Things within the Service Architecture of Intelligent Transport Systems." In: *2017 European Conference on Electrical Engineering and Computer Science (EECS)*. IEEE, pp. 351–355.

Lewis, M., B. Young, L. Mathiassen, A. Rai, and R. Welke (2007). "Business process innovation based on stakeholder perceptions." In: *Information Knowledge Systems Management* 6.1, 2, pp. 7–27.

Li, Q., S. P. Gochhayat, M. Conti, and F. Liu (2017). "EnergIoT: A solution to improve network lifetime of IoT devices." In: *Pervasive and Mobile Computing* 42, pp. 124–133. DOI: 10.1016/j.pmcj.2017.10.005.

Li, S., L. Da Xu, and S. Zhao (2015). "The internet of things: a survey." In: *Information Systems Frontiers* 17.2, pp. 243–259. DOI: 10.1007/s10796-014-9492-7.

Lim, C. and P. P. Maglio (2019). "Clarifying the Concept of Smart Service System." In: *Handbook of Service Science, Volume II*. Springer International Publishing, pp. 349–376. ISBN: 978-3-319-98512-1. DOI: 10.1007/978-3-319-98512-1_16.

Lo, W., J. Yin, S. Deng, Y. Li, and Z. Wu (June 2012). "Collaborative Web Service QoS Prediction with Location-Based Regularization." In: *2012 IEEE 19th International Conference on Web Services*, pp. 464–471. DOI: 10.1109/ICWS.2012.49.

López-Jaquero, V., J. Vanderdonckt, F. Montero, and P. González (2007). "Towards an extended model of user interface adaptation: the Isatine framework." In: *IFIP International Conference on Engineering for Human-Computer Interaction*. Springer, pp. 374–392. DOI: 10.1007/978-3-540-92698-6_23.

Luberg, A., T. Tammet, and P. Järv (2011). "Smart City: A Rule-based Tourist Recommendation System." In: *Information and Communication Technologies in Tourism 2011*. Ed. by R. Law, M. Fuchs, and F. Ricci. Vienna: Springer Vienna, pp. 51–62. ISBN: 978-3-7091-0503-0.

Lyytinen, K. (1987). "A Taxonomic Perspective of Information Systems Development: Theoretical Constructs and Recommendations." In: *Critical Issues in Information Systems Research*. USA: John Wiley & Sons, Inc., pp. 3–41. ISBN: 0471912816.

Machado, A., A. M. Pernas, L. K. Wives, and J. P. M. de Oliveira (2014). "Situation-Aware Smart Environment Modeling." In: *Advances in Conceptual Modeling*. Ed. by J. Parsons and D. Chiu. Cham: Springer International Publishing, pp. 139–149. ISBN: 978-3-319-14139-8.

Madria, S., V. Kumar, and R. Dalvi (2014). "Sensor Cloud: A Cloud of Virtual Sensors." In: *IEEE Software* 31.2, pp. 70–77.

Marquezan, C. C., F. Wessling, A. Metzger, K. Pohl, C. Woods, and K. Wallbom (2014). "Towards exploiting the full adaptation potential of cloud applications." In: *Proceedings of the 6th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems*, pp. 48–57. DOI: 10.1145/2593793.2593799.

McIlroy, M. D., J. Buxton, P. Naur, and B. Randell (1968). "Mass-produced software components." In: *Proceedings of the 1st international conference on software engineering, Garmisch Pattenkirchen, Germany*, pp. 88–98.

Mell, P. and T. Grance (2011). "The NIST definition of cloud computing." In: 800-145.

Minerva, R., A. Biru, and D. Rotondi (2015). "Towards a definition of the Internet of Things (IoT)." In: *IEEE Internet Initiative* 1.1, pp. 1–86.

MOF, O. (2014). *Meta Object Facility*. URL: https://www.omg.org/spec/MOF/2.4.2 (visited on 12/26/2018).

Mohanty, S. P., U. Choppali, and E. Kougianos (2016). "Everything you wanted to know about smart cities: The internet of things is the backbone." In: *IEEE Consumer Electronics Magazine* 5.3, pp. 60–70.

Montenegro, G., N. Kushalnagar, J. Hui, and D. Culler (Sept. 2007). *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC 4944 (Proposed Standard). Internet Engineering Task Force (IETF). DOI: 10.17487/RFC4944.

Mshali, H., T. Lemlouma, and D. Magoni (2018). "Adaptive monitoring system for e-health smart homes." In: *Pervasive and Mobile Computing* 43, pp. 1–19. DOI: 10.1016/j.pmcj.2017.11.001.

Nakahara, F. A. and D. M. Beder (2018). "A context-aware and self-adaptive offloading decision support model for mobile cloud computing system." In: *Journal of Ambient Intelligence and Humanized Computing* 9.5, pp. 1561–1572. DOI: 10.1007/s12652-018-0790-7.

Narasimhan, G., B. G. Ephrem, S. Cheriyan, and N. Balasupramanian (2017). "Predictive analytics of road accidents in Oman using machine learning approach." In: *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. IEEE, pp. 1058–1065.

Nations, U. (2014). *World urbanization prospects*, p. 32. DOI: https://doi.org/https://doi.org/10.18356/527e5125-en. URL: https://www.un-ilibrary.org/content/publication/527e5125-en.

Nechkoska, R. P. (2020). *Tactical Management in Complexity: Managerial and Informational Aspects*. Springer, Cham. ISBN: 978-3-030-22804-0. DOI: https://doi.org/10.1007/978-3-030-22804-0.

Nettsträter, A., J. R. Nopper, C. Prasse, and M. ten Hompel (2010). "The internet of things in logistics." In: *European Workshop on Smart Objects: Systems, Technologies and Applications*. VDE, pp. 1–8.

Newman, S. (2015). *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc."

Obinikpo, A. A. and B. Kantarci (2019). "Big data aggregation in the case of heterogeneity: a feasibility study for digital health." In: *International Journal of Machine Learning and Cybernetics* 10.10, pp. 2643–2655.

Oltean, V. E., T. Borangiu, M. Dragoicea, and I. Iacob (2013). "Approaches and Challenges in Viable Service Systems Development." In: *4th International Conference on Exploring Service Science*. URL: http://www.inseed.cimr.pub.ro/documents/prezentari/Approaches%5C%20and%5C%20Challenges%5C%20in%5C%20Viable%5C%20Service[Oltean%5C%20feb%5C%202013].pdf.

OMG (2016). *OMG Meta Object Facility (MOF) Core Specification. Version 2.5.1. October 2016*. URL: https://www.omg.org/spec/MOF/.

Oteafy, S. M. and H. S. Hassanein (2016). "Resilient IoT architectures over dynamic sensor networks with adaptive components." In: *IEEE Internet of Things Journal* 4.2, pp. 474–483. DOI: 10.1109/JIOT.2016.2621998.

Palanca, J., E. d. Val, A. Garcia-Fornes, H. Billhardt, J. M. Corchado, and V. Julián (Feb. 2018). "Designing a goal-oriented smart-home environment." In: *Information Systems Frontiers* 20.1, pp. 125–142. ISSN: 1572-9419. DOI: 10.1007/s10796-016-9670-x.

Papazoglou, M. P. and D. Georgakopoulos (Oct. 2003). "Introduction: Service-Oriented Computing." In: *Commun. ACM* 46.10, pp. 24–28. ISSN: 0001-0782. DOI: 10.1145/944217.944233.

Park, C. Y., K. B. Laskey, S. Salim, and J. Y. Lee (July 2017). "Predictive situation awareness model for smart manufacturing." In: *2017 20th International Conference on Information Fusion (Fusion)*, pp. 1–8. DOI: 10.23919/ICIF.2017.8009849.

Peng, X., B. Chen, Y. Yu, and W. Zhao (2012). "Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop." In: *Journal of Systems and Software* 85.12, pp. 2707–2719. DOI: 10.1016/j.jss.2012.04.079.

Perera, C., A. Zaslavsky, P. Christen, and D. Georgakopoulos (2014). "Context Aware Computing for The Internet of Things: A Survey." In: *IEEE Communications Surveys & Tutorials* 16.1, pp. 414–454.

Petrolo, R., V. Loscrì, and N. Mitton (2017). "Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms." In: *Transactions on Emerging Telecommunications Technologies* 28.1, e2931. DOI: 10.1002/ett.2931.

Pinheiro, M. K. and C. Souveyet (2018). "Supporting context on software applications: a survey on context engineering." In: *Modélisation et utilisation du contexte* 2.1. ISSN: 2514-5711. DOI: 10.21494/ISTE.OP.2018.0275.

Prevention of Traffic Accidents (Morocco), N. C. for the (2016). *Provisional assessment of road accidents for 2017*. URL: http://cnpac.gov.ma/wp-content/uploads/2017/08/Bilan2017_Stat-prov..pdf (visited on 12/24/2018).

Puzicha, J., J. M. Buhmann, Y. Rubner, and C. Tomasi (1999). "Empirical evaluation of dissimilarity measures for color and texture." In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. IEEE, pp. 1165–1172. DOI: 10.1109/ICCV.1999.790412.

Quadrana, M., P. Cremonesi, and D. Jannach (2018). "Sequence-aware recommender systems." In: *ACM Computing Surveys (CSUR)* 51.4, pp. 1–36.

Reinsel, D., J. Gantz, and J. Rydning (Nov. 2018). *The Digitization of the World - From Edge to Core*. Tech. rep. US44413318. IDC. URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.

Ribino, P., C. Lodato, A. Cavaleri, and M. Cossentino (2016). "A Norm-Based Approach for Personalising Smart Environments." In: *Intelligent Interactive Multimedia Systems and Services 2016*. Ed. by G. D. Pietro, L. Gallo, R. J. Howlett, and L. C. Jain. Cham: Springer International Publishing, pp. 659–670. ISBN: 978-3-319-39345-2.

Rodrigues, G. S., F. P. Guimarães, G. N. Rodrigues, A. Knauss, J. P. C. de Araújo, H. Andrade, and R. Ali (2019). "GoalD: A Goal-Driven deployment framework for dynamic and heterogeneous computing environments." In: *Information and Software Technology* 111, pp. 159–176. DOI: 10.1016/j.infsof.2019.04.003.

Rolland, C. (2007). "Capturing System Intentionality with Maps." In: *Conceptual Modelling in Information Systems Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 141–158. ISBN: 978-3-540-72677-7. DOI: 10.1007/978-3-540-72677-7_9.

Ryan, N. S., J. Pascoe, and D. R. Morse (1998). "Enhanced reality fieldwork: the context-aware archaeological assistant." In: *Computer applications in archaeology*. Tempus Reparatum.

Ryu, S., M. Kim, and H. Kim (2020). "Denoising Autoencoder-Based Missing Value Imputation for Smart Meters." In: *IEEE Access* 8, pp. 40656–40666.

Santana, E. F. Z., A. P. Chaves, M. A. Gerosa, F. Kon, and D. S. Milojicic (Nov. 2017). "Software Platforms for Smart Cities: Concepts, Requirements, Challenges, and a Unified Reference Architecture." In: *ACM Comput. Surv.* 50.6, 78:1–78:37. ISSN: 0360-0300. DOI: 10.1145/3124391.

Santofimia, M. J., D. Villa, O. Aceña, X. del Toro, C. Trapero, F. J. Villanueva, and J. C. Lopez (2018). "Enabling smart behavior through automatic service composition for Internet of Things–based Smart Homes." In: *International Journal of Distributed Sensor Networks* 14.8.

Sarwar, B., G. Karypis, J. Konstan, and J. Riedl (2000). "Analysis of Recommendation Algorithms for e-Commerce." In: *Proceedings of the 2Nd ACM Conference on Electronic Commerce*. EC '00. Minneapolis, Minnesota, USA: ACM, pp. 158–167. ISBN: 1-58113-272-7. DOI: 10.1145/352871.352887.

Schilit, B. N. and M. M. Theimer (1994). "Disseminating active map information to mobile hosts." In: *IEEE Network* 8.5, pp. 22–32.

Schwartze, V., M. Blumendorf, and S. Albayrak (2010). "Adjustable context adaptations for user interfaces at runtime." In: *Proceedings of the International Conference on Advanced Visual Interfaces*, pp. 321–324. DOI: 10.1145/1842993.1843051.

Sedhain, S., A. K. Menon, S. Sanner, and L. Xie (2015). "Autorec: Autoencoders meet collaborative filtering." In: *Proceedings of the 24th International Conference on World Wide Web*. ACM, pp. 111–112.

Shafer, T. (2017). "The 42 V's of big data and data science." In: *Elder Research*.

Sivamani, S., N. Bae, and Y. Cho (2013). "A Smart Service Model Based on Ubiquitous Sensor Networks Using Vertical Farm Ontology." In: *International Journal of Distributed Sensor Networks* 9.12. DOI: 10.1155/2013/161495.

Slabicki, M., G. Premsankar, and M. Di Francesco (2018). "Adaptive configuration of LoRa networks for dense IoT deployments." In: *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, pp. 1–9. DOI: 10.1109/NOMS.2018.8406255.

Slaimi, F., A. B. Hassine, and M. Tagina (2014). "Ontology based vertical web service composition." In: *International Journal of Knowledge-based and Intelligent Engineering Systems* 18.1, pp. 1–9.

Soh, H., S. Sanner, M. White, and G. Jamieson (2017). "Deep sequential recommendation for personalized adaptive user interfaces." In: *Proceedings of the 22nd international conference on intelligent user interfaces*, pp. 589–593. DOI: 10.1145/3025171.3025207.

Sood, S. K., R. Sandhu, K. Singla, and V. Chang (2018). "IoT, big data and HPC based smart flood management framework." In: *Sustainable Computing: Informatics and Systems* 20, pp. 102–117. DOI: 10.1016/j.suscom.2017.12.001.

Souabni, R., I. B. Saâdi, Kinshuk, and H. B. Ghezala (2018). "Context Weighting for Ubiquitous Learning Situation Description: Approach Based on Combination of Weighted Experts' Opinions." In: *International Journal of Information Technology & Decision Making* 17.01, pp. 247–309. DOI: 10.1142/S0219622017500407.

Stone, V. M. (Sept. 2008). "The auto-associative neural network - a network architecture worth considering." In: *2008 World Automation Congress*, pp. 1–4.

Sultan, M. and K. N. Ahmed (2017). "SLASH: Self-learning and adaptive smart home framework by integrating IoT with big data analytics." In: *2017 Computing Conference*. IEEE, pp. 530–538. DOI: 10.1109/SAI.2017.8252147.

Sun, H., Z. Zheng, J. Chen, and M. R. Lyu (Oct. 2013). "Personalized Web Service Recommendation via Normal Recovery Collaborative Filtering." In: *IEEE Transactions on Services Computing* 6.4, pp. 573–579. DOI: 10.1109/TSC.2012.31.

Sundar, S. S., X. Dou, and S. Lee (2013). "Communicating in a ubicomp world: interaction rules for guiding design of mobile interfaces." In: *IFIP Conference on Human-Computer Interaction*. Springer, pp. 730–747.

Tang, M., Y. Jiang, J. Liu, and X. Liu (June 2012). "Location-Aware Collaborative Filtering for QoS-Based Service Recommendation." In: *2012 IEEE 19th International Conference on Web Services*, pp. 202–209. DOI: 10.1109/ICWS.2012.61.

Truong, H. and S. Dustdar (Mar. 2015). "Principles for Engineering IoT Cloud Systems." In: *IEEE Cloud Computing* 2.2, pp. 68–76. ISSN: 2325-6095. DOI: 10.1109/MCC.2015.23.

Tzafilkou, K., N. Protogeros, and A. Koumpis (2017). "User-centred cloud service adaptation: an adaptation framework for cloud services to enhance user experience." In: *International Journal of Computer Integrated Manufacturing* 30.4-5, pp. 472–482. DOI: 10.1080/0951192X.2015.1030697.

Uhlig, R., G. Neiger, D. Rodgers, A. L. Santoni, F. C. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith (2005). "Intel virtualization technology." In: *Computer* 38.5, pp. 48–56.

Varadan, R., K. Channabasavaiah, S. Simpson, K. Holley, and A. Allam (2008). "Increasing business flexibility and SOA adoption through effective SOA governance." In: *IBM Systems Journal* 47.3, pp. 473–488.

Villa, D., O. Aceña, F. J. Villanueva, M. J. Santofimia, S. Escolar, X. del Toro Garca, and J. C. Lopez (Oct. 2017). "IDM: An inter-domain messaging protocol for IoT." In: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 8355–8360. DOI: 10.1109/IECON.2017.8217467.

Walker, M. and B. Burton (2015). *Hype Cycle for Emerging Technologies, 2015*. URL: https://www.gartner.com/en/documents/3100227/hype-cycle-for-emerging-technologies-2015.

Wang, H., L. Wang, Q. Yu, Z. Zheng, and Z. Yang (2018). "A proactive approach based on online reliability prediction for adaptation of service-oriented systems." In: *Journal of Parallel and Distributed Computing* 114, pp. 70–84. DOI: 10.1016/j.jpdc.2017.12.006.

Wang, S. and S. Dey (2013). "Adaptive mobile cloud computing to enable rich mobile multimedia applications." In: *IEEE Transactions on Multimedia* 15.4, pp. 870–883. DOI: 10.1109/TMM.2013.2240674.

Ward, J. S. and A. Barker (2013). "Undefined by data: a survey of big data definitions." In: *arXiv preprint arXiv:1309.5821*.

WHO (2018). "Global status report on road safety 2018." In: *Geneva: World Health Organization*. URL: https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/.

Winkler, V. J. (2011). *Securing the Cloud: Cloud computer Security techniques and tactics*. Elsevier.

Wolter, D. and A. Kirsch (2017). "Smart Environments: What is it and Why Should We Care?" In: *KI - Künstliche Intelligenz* 31.3, pp. 231–237. DOI: 10.1007/s13218-017-0498-4.

WSDREAM (2011). *WS-DREAM: A Package of Open Source-Code and Datasets to Benchmark QoS Prediction Approaches of Web Services*. URL: https://github.com/wsdream (visited on 12/30/2018).

Wu, D., B. Parsia, E. Sirin, J. Hendler, and D. Nau (2003). "Automating DAML-S web services composition using SHOP2." In: *International semantic web conference*. Springer, pp. 195–210.

Wu, J., L. Chen, Y. Feng, Z. Zheng, M. C. Zhou, and Z. Wu (Mar. 2013). "Predicting Quality of Service for Selection by Neighborhood-Based Collaborative Filtering." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 43.2, pp. 428–439. DOI: 10.1109/TSMCA.2012.2210409.

Yan, Y., L. Zhang, Q. Z. Sheng, B. Wang, X. Gao, and Y. Cong (2019). "Dynamic Release of Big Location Data Based on Adaptive Sampling and Differential Privacy." In: *IEEE Access* 7, pp. 164962–164974. DOI: 10.1109/ACCESS.2019.2951364.

Yu, D., Y. Liu, Y. Xu, and Y. Yin (June 2014). "Personalized QoS Prediction for Web Services Using Latent Factor Models." In: *2014 IEEE International Conference on Services Computing*, pp. 107–114. DOI: 10.1109/SCC.2014.23.

Zhang, S., L. Yao, A. Sun, and Y. Tay (Feb. 2019). "Deep Learning Based Recommender System: A Survey and New Perspectives." In: *ACM Comput. Surv.* 52.1. ISSN: 0360-0300. DOI: 10.1145/3285029.

Zhang, Y., Z. Zheng, and M. R. Lyu (Oct. 2011). "Exploring Latent Features for Memory-Based QoS Prediction in Cloud Computing." In: *2011 IEEE 30th International Symposium on Reliable Distributed Systems*, pp. 1–10. DOI: 10.1109/SRDS.2011.10.

Zhang, Y. and Y.-M. Cheung (2014). "Discretizing numerical attributes in decision tree for big data analysis." In: *2014 IEEE International Conference on Data Mining Workshop*. IEEE, pp. 1150–1157.

Zhao, T., W. Zhang, H. Zhao, and Z. Jin (2017). "A reinforcement learning-based framework for the generation and evolution of adaptation rules." In: *2017 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, pp. 103–112. DOI: 10.1109/ICAC.2017.47.

Zheng, Z., H. Ma, M. R. Lyu, and I. King (Apr. 2011). "QoS-Aware Web Service Recommendation by Collaborative Filtering." In: *IEEE Transactions on Services Computing* 4.2, pp. 140–152. DOI: 10.1109/TSC.2010.52.

— (July 2013). "Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization." In: *IEEE Transactions on Services Computing* 6.3, pp. 289–299. DOI: 10.1109/TSC.2011.59.

Zheng, Z., Y. Zhang, and M. R. Lyu (Jan. 2014). "Investigating QoS of Real-World Web Services." In: *IEEE Trans. Serv. Comput.* 7.1, pp. 32–39. ISSN: 1939-1374. DOI: 10.1109/TSC.2012.34.

Zhong, R. Y., G. Q. Huang, and Q. Dai (2014). "A big data cleansing approach for n-dimensional RFID-Cuboids." In: *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, pp. 289–294.