Designing, Aligning, and Visualizing Service Systems

THÈSE N^{O} 8502 (2018)

PRÉSENTÉE LE 30 AVRIL 2018

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE MODÉLISATION SYSTÉMIQUE
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Gorica TAPANDJIEVA

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

Abstract

Service is a concept that separates the concerns of an organization into (1) the value created for users and (2) the way the organization manages its resources to provide this value. The discipline of management of information technology (IT) uses services to coordinate and to optimize the use of IT resources (servers, applications, databases, etc.) in a way that brings value to users.

The concrete application of the service concept is challenging due to its abstract, interdependent and recursive nature. We experienced this challenge while collaborating with the IT department of our university (École Polytechnique Fédérale de Lausanne, EPFL) when the IT department adopted the IT Infrastructure Library (ITIL) best-practices framework for IT service management. As researchers, we have the goal of improving the understanding of services as a means to structuring what people and organizations do. In the context of the IT department, we studied how to apply the service concept internally within the IT department, and externally (as business services) in the overall organization.

In this thesis, we model services by using systems thinking principles. In particular, we use and improve SEAM, the systemic service-modeling method developed in our laboratory.

Our main result is an ontology for SEAM service modeling. Our contributions are the heuristics that define how the ontology relates to a perceived reality: for example, the heuristics focus on behavior rather than organization and they put an emphasis on service instances rather than service types. We also define alignment between service systems, based on the properties of the systems' behavior.

We show how to model an organization by implementing the concept of service as defined by our ontology. This ontology supports the design of service systems that align across both IT and business services. During our work with over one hundred IT services, we developed several visualization prototypes of a service cartography; we use these prototypes to describe and to relate the different views required for managing services.

Our results offer a concrete way to implement the abstract concept of services. This way could be of interest for any organization willing to embark on a large-scale service project.

Keywords: service science, systems thinking, ITIL, SEAM, service design, service modeling, service ontology, business/IT alignment, service visualization, service cartography

Résumé

Le service est un concept qui sépare les enjeux d'une organisation en (1) la valeur créée pour les utilisateurs et (2) la manière dont l'organisation gère ses ressources pour fournir cette valeur. La discipline du management des technologies de l'information ("information technology", IT) utilise les services pour coordonner et optimiser l'utilisation de ressources IT (serveurs, applications, bases de données, etc.) de telle manière à apporter de la valeur aux utilisateurs.

L'application concrète du concept de service est délicate en raison de la nature abstraite, interdépendante et récursive de ce concept. Nous avons fait face à ce défi en collaborant avec le département IT de notre université (École Polytechnique Fédérale de Lausanne, EPFL) quand le département IT a adopté l'ITIL ("IT Infrastructure Library", gestion des bibliothèques d'infrastructure informatique), qui est un cadre de bonnes pratiques pour le management des services IT. Comme chercheurs, notre objectif est d'améliorer la compréhension des services en tant que moyen de structurer ce que les gens et les organisations font. Dans le contexte du département IT, nous avons étudié comment appliquer le concept de service à l'intérieur du département informatique et à l'extérieur (comme services métier) dans l'ensemble de l'organisation.

Dans cette thèse, nous modélisons des services en utilisant des principes de pensée systémique. En particulier, nous utilisons et améliorons SEAM, la méthode systémique de modélisation qui a été développée dans notre laboratoire.

Notre résultat principal est une ontologie pour la modélisation de services avec SEAM. Nos contributions sont les heuristiques qui définissent comment l'ontologie est liée à la réalité perçue : par exemple, les heuristiques se concentrent sur le comportement plutôt que l'organisation et mettent l'accent sur les instances de services plutôt que sur les types de services. Nous définissons aussi un alignement basé sur les propriétés du comportement des systèmes.

Nous démontrons qu'il est possible de modéliser une organisation en utilisant le concept de service tel que défini par notre ontologie. Cette ontologie permet la conception de systèmes de services alignés qui s'étendent de l'IT au business. Pendant notre travail avec plus d'une centaine de services IT, nous avons développé plusieurs prototypes de visualisation d'une cartographie des services; nous utilisons ces prototypes pour décrire et lier les différentes vues nécessaires à la gestion des services.

Nos résultats offrent une manière concrète de mettre en œuvre le concept abstrait de services. Cette manière pourrait présenter un intérêt pour toutes les organisations souhaitant commencer un projet de service à grande échelle.

Mots clefs : science des services, pensée systémique, ITIL, SEAM, conception des services, modélisation des services, ontologie des services, alignement informatique/métier, visualisation des services, cartographie des services

Резиме

Сервисот е концепт кој ги дели одговорностите на една организација на два дела: (1) вредноста која се создава за корисникот и (2) начинот на кој организацијата управува со своите ресурси за да ја обезбеди оваа вредност. Полето за управување на информациските технологии (на англиски: information technology, IT) користи сервиси за координација и оптимизација на информациски ресурси (сервери, апликации, бази на податоци, итн.) на начин кој им носи одредена вредност на корисниците.

Конкретната примена на концептот на сервиси претставува вистински предизвик поради природата на самите сервиси, која е апстрактна, меѓузависна и рекурзивна. Ние се соочивме со овој предизвик додека соработувавме со одделот за информациски технологии на нашиот универзитет (на француски: École Polytechnique Fédérale de Lausanne, EPFL) кога одделот ја усвои библиотеката за инфраструктура на информациските технологии (на англиски: IT Infrastructure Library, ITIL) која претставува рамка со добри практики во полето на управување на овие технологии. Како истражувачи, имаме цел да го подобриме разбирањето на сервисите како начин на структурирање на работата на луѓето и организациите. Во случајот на информацискиот оддел, изучувавме како да се примени концептот на сервиси во рамки на одделот, но и надвор од него.

Во оваа докторска теза, сервисите се моделирани користејќи принципи на системско мислење. Конкретно, го користиме и подобруваме SEAM системскиот метод за моделирање на сервиси кој е развиен во нашата лабораторија.

Главниот резултат на нашата работа е онтологија за моделирање на сервиси со SEAM. Наш придонес се хевристиките кои дефинираат како онтологијата се поврзува со перцепираната реалност: на пример, хевристиките се фокусираат на активностите место на организацијата и ги нагласуваат инстанците наместо типовите на сервиси. Исто така, го дефинираме усогласувањето меѓу системите од сервиси врз база на својствата на активностите на системите.

Преку имплементирање на концептот на сервиси онака како што го дефинираме со нашата онтологија, се прикажува како да се моделира една организација. Онтологијата поддржува дизајнирање на системи од сервиси кои истовремено се усогласени и со информациските, и со бизнис сервисите. Во нашата работа вклучивме повеќе од сто информациски сервиси, што резултираше со развивање на повеќе прототипи за визуелизација на картографијата на тие сервиси; овие прототипи се корисни за опишување и поврзување на различните погледи потребни за управување со сервиси.

Постигнатите резултати нудат конкретен начин на имплементација на апстрактниот концепт на сервиси. Овој начин на имплементација може да биде корисен за секоја организација која е подготвена да започне проект за сервиси од големи размери.

Клучни зборови: наука за сервиси, системско мислење, ITIL, SEAM, дизајн на сервиси, моделирање на сервиси, онтологија на сервиси, порамнување на бизнисот со информациските технологии, визуелизација на сервиси, картографија на сервиси

Acknowledgements

My journey in pursuing a PhD thesis was a constant reflection on the academic, practical and personal aspects of the unique experiences I had during these past six years spent in the Systemic Modeling Laboratory (LAMS); first as a master's, and afterwards as a PhD student. With these acknowledgements, I present my reflections on all the people who positively influenced me and my work.

First and foremost, I would like to express my profound gratitude to my thesis advisor, Prof. Alain Wegmann, for his criticism, support, enthusiasm, and guidance during my PhD journey. The questions he raised in meetings were always puzzling yet motivating and stimulated my curiosity for further inquiry. Thank you, Alain, for believing in me from the moment you met me and for being a mentor in every aspect of my life. Despite my resistance, you made me confident that my work is truly valuable in practice.

I am thankful to the members of my thesis committee. I truly appreciate Prof. Karl Aberer, Prof. Yves Pigneur, and Prof. Irina Rychkova for evaluating my thesis and providing constructive feedback. I am grateful and honored for having Prof. Martin Odersky as the president of the committee.

My name might be under the title, but the work I present in this PhD thesis is as much mine as that of the people of the IT department at EPFL. I was extremely fortunate to have collaborated with Giorgio Anastopoulos, Maude Grossan and Didier Rey during my PhD research. I am indebted to them for providing me valuable information about their work, which was crucial for the completion of my thesis.

LAMS is an extraordinary place to work thanks to the people who are and were part of it. I am thankful to Gil for steering me in the right research direction through our fruitful discussions. He helped me learn how to use my practical experience in order to ground my work and my writing, despite my stubbornness for preferring abstractions. He is still the only person with whom I feel comfortable speaking French. *Merci Gil pour toutes nos discussions, en français et en anglais.* The last two years in LAMS were marvelous thanks to time spent with my dear friend and colleague Betty, in the office and outside of it. Thank you Betty for always being open for discussions and motivated to collaborate with me. Pierre, thank you for the cheerful conversations at lunch, in the office, and at the apéros in BC, and of course for enlightening me with the music of Les Cowboys Fringants. A huge thanks goes to Arash for showing me how to live a heroic journey of bringing more musical riffs to life and of following his teaching passion. I wish to thank Lucien for sharing the practical and business perspectives with LAMS researchers. Thanks to Anshuman for his countless academic and professional advice. Special thanks to Aarthi for being abundantly positive during her time in LAMS and to Blaise for SeamCAD and organizing the best outing. I would also like to thank all the students that worked on projects with me, especially the two students with whom I co-authored papers: Georgios Piskas and Matteo Filipponi.

Conducting PhD research is an easy endeavor when surrounded and helped by an amazing administrative staff, Angela and Patricia. There were no IT issues that could not be taken care of by Yves and Marc-André. Holly proofread and corrected the English mistakes in this thesis and in a few of my papers. A huge thanks to all of you!

I am overwhelmed by the support I received from all the Macedonians I met in Switzerland. Blagica and Brane are the people I have known from the first day I came to Switzerland. Among many other things, I would like to thank them for keeping me from being homesick for Macedonian food. The list of Macedonians is of course incomplete without my friends Ana M., Darko, Dino, Filip, Igor, Jovanche, Kristina, Maja, Maki, Mladen, Natasha, Nikolche, Paulina, Stefan, Vase, and many others. I genuinely enjoyed and had an amazing time during all of our lunches, dinners, board-game nights, birthdays, barbeques, and other events.

Very special and warm thanks go to all the people I met while in Switzerland who became my close friends and made my life more joyful – Vijay, Sergio, Rita, Eray, Vassilis, Aleksei, Vagia, Deniz, Miroslav, Teodora and many others.

A special gratitude goes to Ana, for her kindness and friendship. I feel very lucky to have her as one of my closest and dearest friends. Towards the end of my PhD journey, I had the pleasure to meet Ana's son Pavle who, with his smiles, brightened the days when I was writing this thesis.

A very particular appreciation goes to Ana, Adi, Josif, Koljo, Ole, Elena, and Perica, the people I have known since my childhood and teenage years. I could not thank them enough for the love, friendship, and support they have shown, not only while I was pursuing my PhD degree, but even before.

I am especially thankful to my boyfriend Emilijan who decided to accompany me on this journey. From the moment he joined me in Lausanne, he made my days joyful, exciting, lazy, easy and fun. I could not thank him enough for his care, patience, encouragements, understanding, support, and most of all love, especially when I was in deadline mode.

Finally, I would like to thank the people for whom there are no words that can fully capture how much they mean to me. An exceptional thank you to my nephews, Andrej and Viktor, who always keep me positive. My parents, Violeta and Mitko, and my sister, Tatjana, have been the greatest support in all my endeavors. They always believe in me, give me unconditional love, and teach me real values in life. They always motivated me to pursue my dreams and made me the person I am today.

Мама, тато, Тања, фала ви за неизмерната љубов и поддршка. Оваа дисертација Ви ја посветувам Вам.

Lausanne, 2 April 2018 Gorica Tapandjieva

Contents

List of Figures

List of Tables

1 Introduction

IT service management

People in many domains and disciplines have widely recognized and accepted the notion of a service for structuring business interactions [Bardhan et al., 2010]. This lead to an increasing importance of the service concept that is captured in the abstraction called service system [Maglio et al., 2009]. One field of research and application of services is within the information technology (IT) management field, called IT service management, where the IT Infrastructure Library (ITIL) [Arraj, 2013; Cartlidge et al., 2012] is a commonly adopted framework of best practices. The promise of ITIL is that it can help IT people to effectively manage their IT resources in order to provide value to the business. Although ITIL focuses on processes for managing the daily activities around services, it does not provide means to conceptualize services as a separation of concerns into the value a service brings and into the service implementation.

Services difficult to conceptualize

Despite their wide acceptance, services are difficult to conceptualize due to their nature of being (1) abstract [Maglio et al., 2009], (2) interdependent and (3) recursive [Spohrer et al., 2007]. For example, it is easy to know the functionalities of an IT application, but it is more challenging to conceptualize the value (service) provided by the application without any references to the application itself.

Thesis subject

The subject of this thesis is designing, aligning, and visualizing service systems in the context of an IT department that adopted an IT service management approach. More concretely, the practical context is the extended use of services, both within and outside the IT department of our university, for the management and organization of human and IT resources, with the goal of providing value to users.

Thesis work as part of a wide-scope action research project

We began this PhD research in September 2012 and it was part of a wide-scope longitudinal collaboration between the IT department of our university and the Systemic Modeling Laboratory (LAMS) [LAMS]. The collaboration lasted between 2010 – 2017 and it was conceptualized as an action research project [Avison et al., 1999] led by Prof. Wegmann, the thesis advisor and the head of LAMS. Initially, in 2010, the goal of this action research project was to help the IT department to transition into a service organization that provides over one hundred services. Later, in 2015, the goal was to coordinate all IT resources of the university around the concept of service. The scope of this thesis is smaller than the overall action research project. We structure this thesis around three projects that provide foundations for conceptualizing services:

(1) designing a service-modeling ontology,
(2) aligning low-level IT services with a segment of users and/or a mission, and
(3) providing prototypes for visualizing and communicating on services.

Each project is presented separately in a chapter of this thesis.

Application of the SEAM systemic paradigm

In this thesis, we use SEAM [Wegmann, 2003], a systemic service-modeling method developed at LAMS. SEAM is based on systems thinking and it is used for service design and modeling. Applying SEAM means adopting the SEAM systemic paradigm. A paradigm explains a person's underlying philosophical assumptions that in turn define the approach in addressing research challenges. The SEAM systemic paradigm is based on interpretivist and constructivist epistemology. The SEAM epistemology influences us to recurrently reflect on people's perspectives and to understand (interpret) how a perspective changes a service conceptualization.

Action research – source of relevant research problems and information

The overall action research project, i.e., collaboration with the IT department, was instrumental in this thesis as we needed concrete experience to be in contact with members of the IT department (practitioners) to interpret what they do. Our interpretations of the challenges and issues within the IT department inspired us to initiate the three research projects we present in this thesis. In addition, the overarching action research project enabled us to collect relevant information from the field (meeting notes, interviews, documents, etc.).

Design-science research method

We design artifacts in all three research projects that we present in this thesis. Two of the three projects conform to the design science for IS research framework [Hevner et al., 2004]. In the third project, members of the IT department actively participated in the design and development of the artifact. This required a method that formally combines design with action research, we therefore used the action design research [Sein et al., 2011].

Thesis contributions

Our main contributions in this thesis are the results of each project. The service science discipline and the SEAM modeling method benefit from our artifacts and heuristics that help in the conceptualization and alignment of services across the whole organization, starting from a low-level IT service to one of the university's missions. Unlike most of the existing literature that treats alignment as an abstract concept, we provide concrete heuristics based on epistemic reflections, and we demonstrate them on a real project. We offer a new perspective on the dynamic nature of service systems with a heuristic that behavior defines the structure of service systems, thus making the service systems independent of organizational hierarchies and boundaries. We present another heuristic explaining that as the service behavior changes depending on the context (time, location, users, etc.), the same service implementation yields as many service instances as contexts in which the service is used. Our final contribution is a visualization tool for describing, connecting and communicating many service instances and perspectives.

Dependencies between chapters

The three projects of this thesis are described in Chapters 4 – 6. Figure 1.1 illustrates the thesis organization with the dependencies between chapters (projects). Next, we give a short description of the Chapters 2 – 6 that are depicted in the figure.

Foundations

In **Chapter 2**, we present the foundations of our work and we cover the themes upon which we base our work. These include service science, systems thinking and SEAM. As mentioned, SEAM is the systemic service-modeling method developed in our research laboratory; this method synthesizes the research and application of systems thinking in IS and business related disciplines, including service science.

Research method

In **Chapter 3**, we give an overview of the set of research methods we considered, based on our philosophical assumptions: design science in IS research, action research and action design research.

These two chapters are the core of our research and influence our approach, therefore we depict them in the center of Figure 1.1 to denote that Chapters 4 – 6 depend on them.

Designing an ontology for service systems modeling

In **Chapter 4**, we present an ontology for modeling service systems with the SEAM systemic method. As part of the ontology, we provide

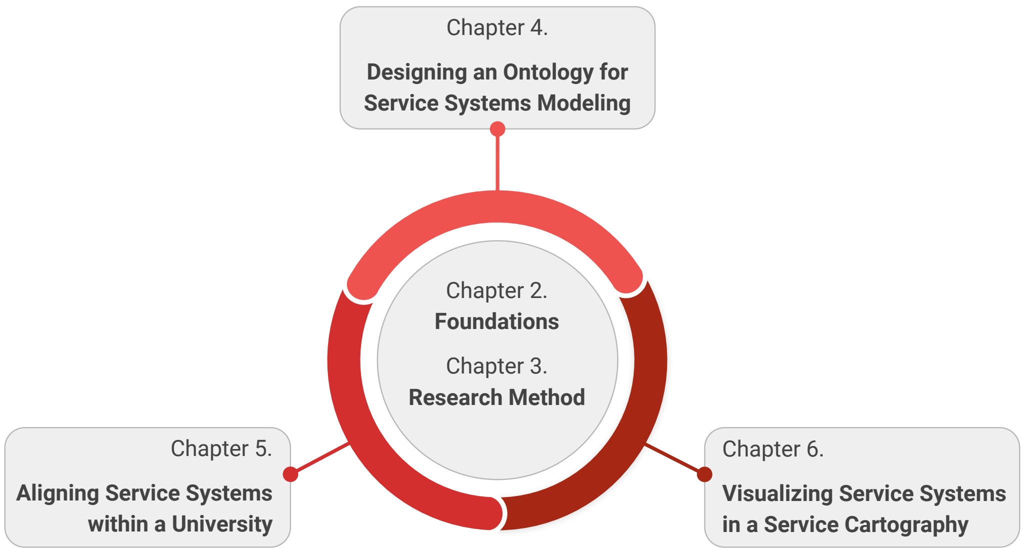

Chapter 4.
Designing an Ontology for Service Systems Modeling
Chapter 2.
Foundations
Chapter 3.
Research Method
Chapter 5.
Aligning Service Systems within a University
Chapter 6.
Visualizing Service Systems in a Service Cartography

Figure 1.1 – Thesis organization.

a meta-model, well-formedness rules and a formalization in the Alloy declarative language for modeling structures [Jackson, 2002] based on first-order logic that we use for constraint solving. The ontology we propose represents an updated and simplified version of the existing SEAM modeling language ontology. To complement the ontology, we also propose a heuristic that puts an emphasis on the behavior of the systems. We use our ontology artifact for building the alignment of services (Chapter 5) and for building the service cartography tool (Chapter 6).

Aligning service systems within a university

In **Chapter 5**, we explain in more detail the context of the collaboration and the challenges the IT department was facing, i.e., the action research project. Within this context, we illustrate how to use the proposed ontology for aligning services. Then, we take an example of a single case study from the action research project to demonstrate how to align a low-level virtualization service with one of the university's missions. An important lesson that we learned while working on this case study was that the alignment requires introducing new services. We propose three heuristics that aid in finding and managing aligned services and, in Appendix B, we formally define and automatically verify the service alignment based on properties of services. In addition, the findings from Chapter 5 influenced the development of various visualizations presented in Chapter 6.

Visualizing Service Systems in a Service Cartography

In **Chapter 6**, we show the design and development of a visualization tool we call service cartography. The idea for such a tool came from the need to model and visualize a map for approximately one hundred services. The basis of this tool is the SEAM service-modeling ontology and some of the visualizations were inspired from the heuristics for aligning services.

The structure of Chapters 4 – 6

In Chapters 4 – 6, We present the three key projects and we organize them to follow the same structure:

- *Context and Motivation*: We recapture the specificities of the collaboration with the IT department that steered our research.
- *Related Work*: We lay out the relevant work in the context of the chapter.
- *Research Method*: We instantiate the research method we follow for the work presented in the chapter.
- *Artifact*: We illustrate the designed artifact on a specific case study/situation from the action research project. In Chapter 6, the artifact was explained within the research method section.
- *Heuristics*: We present the emergent heuristic techniques we propose that complement the artifact.
- *Future Work*: We present directions for future research and practice.
- *Conclusion*: We discuss the practical and theoretical implications of the proposed artifact and heuristics.

Thesis conclusions, future work and appendices

Finally, in the last chapter of this thesis, we present the conclusions and future work. We also provide additional information that complements the thesis in appendix. More concretely, Appendix A contains an extended meta-model of the ontology in Chapter 4 and contains several visualization prototypes that we created during the tool development we describe in Chapter 6. Next, in Appendix B, we present the formal and automated version of the alignment of SEAM service models based on quantitative properties. Finally, in Appendix C, we present a visual and detailed account of the activities constituting this thesis.

2 Foundations

"The systems view is a way of thinking and acting."
— Béla H. Bánáthy

Concrete project with service systems

The work presented in this thesis is about the study of and collaboration with people from the university's IT department. During this collaboration, we studied how to transform the perception of the IT department's work from managing/maintaining technical infrastructure and applications to providing services to users. The IT department adopted the IT Infrastructure Library (ITIL) framework [Cartlidge et al., 2012] for service management to systematically use services across the whole organization. Any organization that provides services is called a service system and the academic discipline that studies services is called service science. In Section 2.1, we present an overview of this discipline.

The SEAM systemic approach to studying service systems

Systems thinking is the core competency of our research laboratory and it is embodied in the SEAM approach. SEAM synthesizes the research and application of systems thinking in IS and business related disciplines, including service science. This thesis illustrates the application of the SEAM method in the context of service science: We used the SEAM method to address the challenges of the IT department's transformation towards a service system. In Section 2.2, we present an overview systems thinking, including the systemic paradigm composed of epistemology, ontology, axiology and methodology. A paradigm helps to describe and understand a person's underlying philosophical assumptions and beliefs. In Section 2.3, we present the main SEAM concepts and the SEAM systemic paradigm. This paradigm (1) illustrates the underlying beliefs we have when using SEAM and (2) explains our constant reflections on the epistemological issues we encounter. In Chapters 4 – 6, we express our reflections as research findings that we call heuristics.

2.1 Service Science

Service concept

The service concept has been studied for approximately forty years in the context of various disciplines [Hill, 1977; Zeithaml et al., 1985], such as operations [Johnston and Clark, 2008], marketing [Wilson et al., 2012] and software design [Krafzig et al., 2005]. Services are encountered within the domains of service-oriented thinking [Demirkan et al., 2009], service-dominant (S-D) logic [Vargo and Lusch, 2004], service science [Chesbrough and Spohrer, 2006], service systems [Spohrer et al., 2007], servitization [Vandermerwe and Rada, 1988] and service-oriented architecture [Krafzig et al., 2005]. The distinction between products and services was present already in the 1970s [Gummesson, 1979], but the paradigm shift that emerged from services was initiated by Vargo and Lusch [2004] with the introduction of the service-dominant logic and the eight foundational premises of this logic. They define services as “the application of specialized competences (knowledge and skills) through deeds, processes, and performances for the benefit of another entity or the entity itself” [Vargo and Lusch, 2004]. Simply put, a service defines how resources (technology, suppliers, people) are organized to bring value to users. The user does not need to know details about how the service is implemented. This enables us to distinguish between the user viewpoint and the implementation viewpoint. All the complexities of the service implementation is captured in the abstraction called service systems.

Service systems

Service systems “are value-creation networks composed of people, technology, and organizations” [Maglio et al., 2006]. They also include “other internal and external *service systems*, and shared information” [Spohrer et al., 2007]. In addition, service systems enable the separate analysis of the value created for the user, from the internal organizational and architectural issues. Note that such analysis includes the systemic perspective of services because the service value is seen as emergent from (1) the internal organization of the provider and (2) the interaction between the provider and the user. For example, in the context of a university, learning can be seen as a co-creation process [Edvardsson et al., 2011, p.328] among a student and a professor, with the support of other resources (librarians, books and IT systems). The value that a student receives during the learning emerges from the interactions and cannot be attributed to one actor or resource.

Service science

The study of services and service systems was formalized as a specific discipline called service science. Such a discipline was necessary to address the growth of services and the expansion of technology, in both business and IT [Spohrer et al., 2007]. Service science can be seen as a collection of efforts aimed at developing, understanding, improving, and innovating service systems that provide value to people [Maglio et al., 2006]. Bardhan et al. [2010] note that: "The goal of service science is to provide a foundation to advance our ability to design, refine, and scale service systems for practical business and societal purposes". Scholars from various disciplines contribute to service science by providing different perspectives.

Different Perspectives on Services

Different functional perspectives

According to the service-dominant logic, services are the fundamental basis of exchange [Vargo and Lusch, 2004, 2008]. Due to the ubiquity of services, it is difficult to make a strict differentiation between the types of services exchanged between provider and consumer; it depends on the perspective of the person observing the service and the context of the service exchange. Within marketing, services represent a shift from tangibles towards intangibles (skills, information, and knowledge), and toward interactivity, connectivity and ongoing relationships between a provider and consumer [Vargo and Lusch, 2004]. In IT development the services approach is widely known as service-oriented architecture and it deals with the development of integration technologies, such as web services and application programming interfaces (APIs), that are combined or reused for higher-level business purposes. In IT management, a widely accepted services approach is the IT Infrastructure Library (ITIL) best-practices framework [Cartlidge et al., 2012] that focuses on the alignment of IT services with the business objectives and that provides support within core processes of a company.

Customer-dominant logic

Heinonen et al. [2013] identify that service-dominant logic has the view of the service provider only where it is sufficient to create value propositions that orchestrate the value creation. In the customer-dominant logic the focus is on how customers embed service providers in their ecosystem [Heinonen and Strandvik, 2015]. The implication for service providers of adopting this logic is the striving towards a better understanding of the customers' activities, experiences and practices within the customers' context. This understanding will ultimately influence the providers' strategies for service design and provision [Heinonen and Strandvik, 2015]. The customer-dominant logic can be considered as systemic, i.e., holistic, because it looks at the customers in their context and not in isolation. One move towards customer-dominant logic would be the integration of customer experience techniques in the service design process as a support for understanding the customer's complex ecosystem where the provider intends to be included [Teixeira et al., 2012].

External perspective

Instead of focusing solely on "the interaction between the firm and the customer" [Lusch et al., 2008], the customer-dominant logic advocates that the service perspective must be expanded outwards, towards the service user. From a systemic standpoint, adopting a customer-dominant logic is justified by the fact that the service user does not exist in a vacuum, he is also a service provider to someone. In [Checkland, 1999, p. 318], Checkland states that a service system is "a system which is conceived as serving another". He then continues: "A conceptual model of such a human activity [service] system cannot be built unless there exists a conceptual model of the system served, since this will dictate the structure and activity of the service system" [Checkland, 1999, p. 318].

Internal perspective

Adopting only the external perspective does not ensure a successful service provision. Scholars, such as [Ahmed and Rafiq, 2003] and [Bruhn, 2003], discuss the logic of employees as customers of internal services. This logic can be called an internal perspective on services, where "organizational units and their employees are seen as service providers, the users of these services as internal customers, and the activities as internal services" [Stauss, 1995]. It has already been shown that the internal customer satisfaction is essential for delivering high-quality services to external customers [Braun et al., 2017]. It is supposed that the internal and external service management are similar [Stauss, 1995], but the complexity of adopting such a model poses a challenge and is still being researched [Braun et al., 2017].

2.2 Systems Thinking

Emergence of systems thinking

Similarly to other pioneers of the systems movement (such as Ashby, Boulding and Wiener) von Bertalanffy [1968, p.30] noticed that there are similar problems and conceptions that have independently appeared in very different fields. Systems thinking emerged as a powerful scientific movement that unifies the principles encountered in specialized disciplines. System thinkers observed the need in science to not only investigate the parts, but to look at the relations and the results from the parts' interaction [von Bertalanffy, 1950]. It is through the focus on relationships in systems that an observer has a chance to truly understand emerging patterns and structures. Banathy and Jenlink [2003] review the key ideas of Bertalanffy and Boulding to describe the emergence and evolution of systems thinking as an "expansionist, nonlinear dynamic, and synthetic mode of thinking".

General Systems Theory

System thinkers create systems theories to study the complexities found in the surrounding world [Banathy and Jenlink, 2003]. One example is the General Systems Theory (GST) that describes an integrated way of observing, interpreting, understanding, and taking action upon the complex surrounding phenomena [von Bertalanffy, 1950]. Some system thinkers, such as Miller [1978] and Beer [1984], use analogies of concrete systems to formulate systems theories and to better grasp the complexities encountered. We present two of them in Appendix A.1.2.

Paradigm

The development of systems theories often follows a systematic study of systems; this is called systems inquiry [Banathy and Jenlink, 2003]. As in every scientific inquiry, no researcher should conduct systems inquiry without clarifying what paradigm guides his approach [Guba et al., 1994, p. 116]. As a concept, the word *paradigm* has been used by Thomas Kuhn, in his book on scientific revolutions, to describe a set of beliefs scientists share about the nature of knowledge (epistemology) and what there is to know (ontology), as a mean to explain the development of science [Kuhn, 1962]. For von Bertalanffy [1968, p. xxi] the concept of a system is a new paradigm. Following the paradigmatic analysis framework in [Iivari, 1991] and the systems inquiry approach defined by [Banathy and Jenlink, 2003], we describe the underlying systemic beliefs through the aspects of ontology, epistemology, axiology and methodology [Wegmann, 2003].

2.2.1 Systemic Paradigm

Epistemology

Epistemology is the study of knowledge and its relationship to beliefs, truth and justification [Steup, 2017]. Epistemology is concerned with the "means by which we may have and express knowledge of the world" [Checkland, 1999, p. 314]. In systems philosophy, epistemological aspects address *how thinking is done*, not on the justification of the correctness of the thinking [Weinberg, 2001, p. 31]. Such reflection shapes the principles of conducting systems inquiry, because people base their inquiries on preexisting fundamental ideas about knowledge: what it is, where it comes from and how it can be captured or created [Nelson and

Stolterman, 2012, p. 219-220].

Ontology

Ontology deals "with whether or not a certain thing, or more broadly entity, exists" [Hofweber, 2017]. Von Bertalanffy explains that systems ontology is concerned "with what is meant by 'system' and how systems are realized at various levels of the world of observation" [von Bertalanffy, 1968, p. xxi]. There is no easy answer to the question, *what systems are there?* As with epistemology, one person might define a distinction between systems that are real, conceptual, abstract, whereas another person might choose to distinguish only between living and artificial systems.

Axiology

The axiological aspect of systems philosophy is focused on the study of value, ethics, and aesthetics [Banathy and Jenlink, 2003]. It is the people conducting the systems inquiry who have to ask themselves about the implications of the actions they take. After all, it is up to people to judge what is moral, ethical, beautiful, etc. Traditional science is distant from these considerations, whereas systems thinking includes them in the inquiry.

Methodology

The word methodology is derived from two Greek words *methodos* (a way of doing something) and *logos* (word, speech, statement, discourse). Bunge [1999, p. 178] simply defines methodology as the study of methods. In the traditional scientific inquiry, methodology is concerned with critically exploring arguments for choosing one method over another. Thus, methodologies might differ across separate disciplines. However, systems methodology encourages system thinkers to choose the approach that is the most appropriate for the system studied [Banathy and Jenlink, 2003], depending on the context, the problem encountered, etc. Within systems inquiry, system methodology deals with (1) the study of methods that help us to obtain knowledge about complex systems, and (2) the description of manners for choosing appropriate tools, methods and models in the systems thinking process [Banathy and Jenlink, 2003].

Different systemic paradigms

Different systems thinkers have different sets of underlying beliefs, hence they apply different systemic paradigms. Checkland and Holwell [1997] give an example of how two different systemic paradigms form two distinct schools of thought in systems thinking, namely *hard* and *soft*. Both of these schools share the holistic thinking that the system is more than the sum of its parts, and that knowledge comes from observation. The difference is mainly in the epistemological position. The hard systems thinkers take a positivist stance and consider "scientific knowledge to be obtainable only from sense data that can be directly experienced and verified between independent observers" [Susman and Evered, 1978, p. 583]. As a consequence, positivist (hard) approaches are free of context, whereas soft system thinkers take an interpretivist stance and consider reality to be a social construction. Within this soft view, making sense of the reality is an agreed subjectivity and not objectivity [Checkland and Holwell, 1997, p. 22]. For interpretivists, knowledge is person-dependent, bound to the situational context.

Modeling is a way to describe the differences within a paradigm. We depict the systems modeling process within the systemic paradigm as a means to better explain and understand the differences between distinct schools of thought.

2.2.2 Systems Modeling

Importance of models

As [Weinberg, 2001, p. 51] suggests that "a system is a way of looking at the world", we find that every systems thinker has his way of defining, describing and using systems. In systems thinking, as well as in other disciplines such as mathematics and physics, people construct and rely on models to better understand the part of the reality they represent with systems. Meadows explains that "everything we think we know about the world is a model" [Meadows, 2008, p. 86]. Sharing this model improves people's understanding and ability to make decisions about the modeled reality. Given this importance of models, let us first describe the way we see a modeling process. We will use our description of this modeling process as a framework for further analysis of epistemological and ontological differences in the hard and soft systems schools of thought.

Modeling process

The modeling process, presented in Figure 2.1, begins when the modeler[1], the person who creates the model, observes part of a reality, known as the universe of discourse (UoD). Hence for different modelers, a system, or more concretely a system model, is considered to be true if there is an agreement about the relation with the part of reality observed[2]. This relationship between the reality and the model is called *conceptualization*. The modeler then interprets his observation by applying his set of conceptualizations on the UoD to distinguish systems and relationships among them. Finally, to share his conceptualization of the UoD, the modeler creates a model in a representation domain

[1]We use observer and modeler interchangeably.

[2]This comes from Tarski's description of correspondence theory, that "the truth of a sentence consists in its agreement with (or correspondence to) reality" [Tarski, 1944, p. 343].

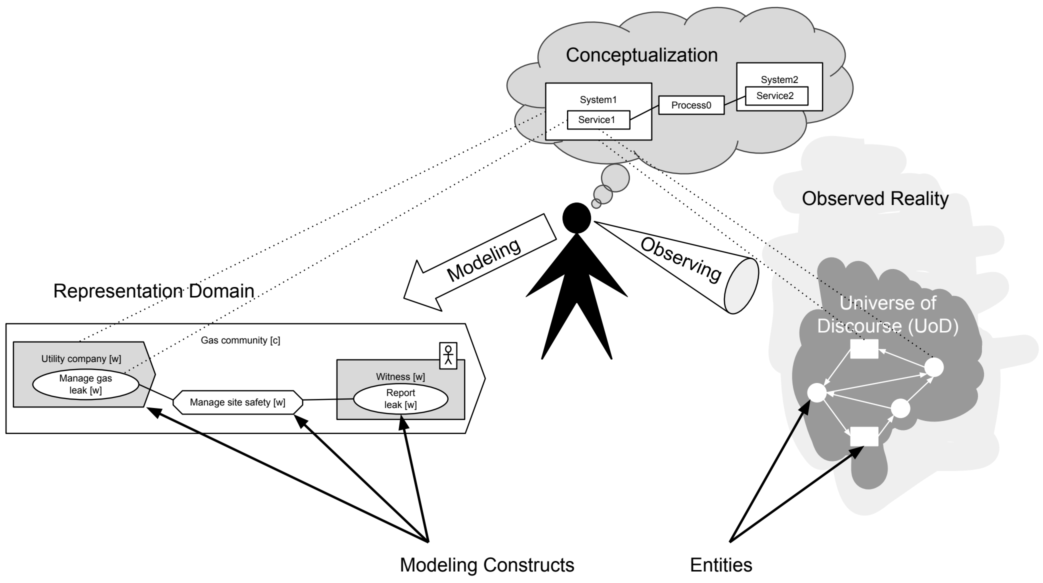


Figure 2.1 – Modeling process: from perception to models. Adapted from [Regev, 2003, p. 66], [Preiss, 2004, p. 55], [Golnam, 2013, p. 54]

in which he translates his conceptualization into modeling constructs [Regev, 2003, p. 66], [Preiss, 2004, p. 55], [Golnam, 2013, p. 54].

Importance of observer (modeler)

Mingers [2006, p. 87] notes that: "Systems thinking is only an epistemology, a particular way of describing the world. It does not tell us what the world is. Hence, strictly speaking, we should never say of something in the world: *It is a system*, only: *It may be described as a system*" by the *observer* of the world. Models are essentially one kind of descriptions done by someone – the observer (modeler). As a consequence, epistemological and ontological discussions depend on the observer (modeler).

Hard vs soft systems thinking schools of thought

The hard systems thinking school of thought assumes that the real world contains systems that can be engineered. It also considers that gathered experiential data on systems can be quantified and verified between independent observers where the use of powerful tools and techniques is oriented towards problem solving and goal seeking, without leaving room for interpretations. The soft systems thinking assumes that systems are intellectual constructs, differently interpreted by different observers. By focusing on interpretations, this school of thought shifts systemicity from the real world to the process of inquiry about the world [Checkland and Holwell, 1997, p. 41]. Thus, it is oriented towards learning, without producing final answers [Ison, 2008, p. 147]. Taking our modeling process from Figure 2.1, we can say that independent modelers belonging to the hard systems thinking school, as positivists, see the UoD as an objective reality of systems. This would not be the case for modelers from the soft systems thinking school, as for them what is actually in the UoD is not as important as the way the UoD is observed, because the reality is subjective and systems interpretations are found in the modeler's conceptualizations.

As noted before, the systems thinking approach we use in this thesis is SEAM.

2.3 SEAM

SEAM introduction

Over the past twenty years, the Systemic Modeling Laboratory (LAMS) has developed the SEAM approach for conducting systems inquiry. SEAM started as a Systemic Enterprise Architecture Methodology [Wegmann, 2003], but through the years it has grown to be a concrete application of the systemic principles, theories and concepts (Section 2.2) in the disciplines of service science, business and IT alignment, enterprise architecture, requirements engineering and strategic thinking. The specific systems view of the world a person takes when applying the SEAM approach is defined in the SEAM philosophy and is communicated via SEAM models.

SEAM modeling

In the SEAM approach, the modeling process is an essential part of the systems inquiry in which system thinkers create and simulate one or more models to gain better insight into the part of reality modeled. In order to create SEAM models, LAMS researchers have developed tools, SeamCAD [LAMS, n.d.b] and the Trade Tour Mind platform [TYM, n.d.]. When available, requirements and constraints are put into SEAM models, and then, these models can be automatically simulated and analyzed by theorem provers or model checkers [Bajić-Bizumić et al., 2013; Rychkova et al., 2008; Tapandjieva and Wegmann, 2015].

2.3.1 SEAM Concepts

System definition

We apply the following definition of a system: *A system is an observer's conceptualization of a set of interrelated entities in an observed reality. This conceptualization can have two views: whole and composite.*

Systems concepts as used in this thesis

In the numerous books and articles on systems, scholars dedicate whole chapters describing fundamental concepts around systems because without additional explanations, one sentence describing a system is of little use. We consider the discussion of these concepts an important but separate topic, hence we list a glossary of the concepts we find essential in SEAM. This glossary should give part of the vocabulary used by SEAM modelers. The descriptions of concepts are adapted from [Checkland, 1999; Gharajedaghi, 2011; Ison, 2008; Klir, 2001; Meadows, 2008; Regev and Wegmann, 2005; Wegmann et al., 2008] and as SEAM takes a constructivist stance, we emphasize that all these concepts are defined by an observer.

Whole	An observer's view of a system where the focus is on the services offered in the environment.
Composite	An observer's view of a system where the component systems and their relationships are visible.
Property	A concept belonging to the system that can have one value at a given moment in time and another value at a different moment in time.
Behavior	Two types of behavior:
Service	A concept belonging to the system seen as a whole, which changes the system's property from one value to another.
Process	A concept belonging to the system seen as a composite, which depends on the properties of the component systems.
Boundary	A concept that encompasses all elements of a system (properties, emergent properties, subsystems, behaviors).
Environment	A concept that denotes the 'outside' of the system boundary. It affects and is affected by the system behavior. It is itself a system that represents the context.

2.3.2 SEAM Systemic Paradigm

SEAM epistemology

SEAM relies on constructivist and interpretive [Checkland and Holwell, 1997; Mintzberg et al., 2005; Wegmann, 2003] epistemology principles. Constructivism is the view that things, such as ideas, entities, relationships and the observed world [Bunge, 1999, p. 48], are human constructions, whereas interpretivism is mostly concerned with understanding the changing reality [Burrell and Morgan, 2017]. Both constructivism and interpretivism are anti-positivist and contextual, meaning that the knowledge represented in a model is relative to the observer. In SEAM, we define a system as "a set of interrelated elements that describes an entity in the (observed) reality as defined by an *observer*" [Regev and Wegmann, 2005] (italics added). Most of the epistemological discussions in SEAM are observer-dependent and are around the relationship between reality, i.e., the universe of discourse (UoD), and the model.

When using SEAM, people from different disciplines are encouraged to use their specific vocabulary and modeling elements that represent systems in the observed UoD. One feature of SEAM is that it enables people to conceptualize the observed reality in a model with several hierarchical, i.e., abstraction levels. Depending on the context and discipline where SEAM is applied, these levels represent different viewpoints of the observed reality: Software engineers observe a hierarchy of software components and IT systems; service managers observe a hierarchy of service systems, and so on.

SEAM ontology

The SEAM ontology corresponds with the computer science definition proposed by [Gruber, 1995] stating that ontology is as an explicit specification of a conceptualization. The SEAM ontology is based on the Reference Model of Open Distributed Processing (RM-ODP) [Wegmann, 2003], that is an ISO/ITU standard. In its current version, the SEAM ontology used in the observer's conceptualization includes *systems*, *services*, *processes*, *properties*, and the *exchange relationships* and *refinement relationships*. These constructs have a different name in the modeling domain, such as *working object* for a system and *localized action* for a process. But in practice, the names from the conceptualization are most often used. The SEAM ontology relies on a meta-model that enables the development of the SEAM tools used for modeling and simulation.

SEAM axiology

The SEAM axiology deals with all the choices a person makes when representing the observed reality in a model. For example: How many and which levels do we conceptualize? Who is the model for? Which observed entities should we include in the model? At which level? All these choices are mostly driven by the values of the observer, hence people using SEAM try to make all the heuristics used in making the choices explicit [Regev et al., 2013; Wegmann, 2003]. One typical example of SEAM axiology (when modeling the UoD) is the preference of declarative semantics over imperative.

SEAM methodology

SEAM facilitates the activities of specialists in the different disciplines with the choice of methods and tools. This choice has to be justified by the SEAM philosophy. Therefore, in line with the constructivist and interpretive epistemology, the SEAM methodology uses methods that put the observer in the context of the stakeholders, such as action research or contextual inquiry, so that the observer constructs his knowledge together with the stakeholders. It is epistemologically important to keep the relationship between the UoD and the model, so when using the SEAM ontology, the modeling elements are labeled according to the specific vocabulary of stakeholders. Furthermore, the ontology can be extended to include specialized entities corresponding to the domain of the SEAM application. Axiologically, SEAM supports both bottom-up and top-down methods for model creation and the observation of reality. SEAM uses simulation tools and techniques that support declarative descriptions of structure and behavior, which is in line with the axiological preference of declarative over imperative semantics. Here are a few examples of approaches that resulted from applying the SEAM systems methodology in different disciplines:

- Wegmann et al. [2008] present a systemic conceptualization approach, based on SEAM modeling, for the Zachman enterprise architecture framework.
- Golnam et al. [2013] combine SEAM with a multi-criteria decision-making method as a requirements engineering technique for a tool selection.
- An integration of SEAM, contextual inquiry and an adoption methodology used in industry for the definition of business requirements is described in [Regev et al., 2011].
- The TradeYourMind web-based platform [TYM, n.d.] used for creating systemic business models to represent eco-systems and stakeholders, to analyze motivations, features, benefits, etc.
- Researchers developed tools that are used in the SEAM approach for a formal and automatic verification of the alignment between the business and IT [Rychkova et al., 2008; Tapandjieva and Wegmann, 2015].

Impact on the research method

In the next chapter we describe how our application of the SEAM systemic paradigm, being interpretive and constructivist, influences the choice of research method. Based on our philosophical assumptions, we use a method rooted in action research and design-science research.

3 Research Method

"Have the courage to use your own reason!"
— Immanuel Kant

Research based on concrete experience

Our quest for knowledge is guided by experiential learning. Through the focus on the learning process of an individual, Kolb [2014] explains that "knowledge is created through transformation of experience". Looking at Kolb's experiential learning cycle in Figure 3.1, we can say that we start working on research projects by first experiencing the potential practical implications of the research. The experience guiding us in this PhD research begun with the master's thesis project titled "Enterprise Bus Selection for EPFL Central Services" [Tapandjieva, 2012] and continued with our collaboration with the university's IT department in their multiple projects towards service-orientation. Through our collaboration with the IT department, we experienced first-hand the challenges IT people face in providing and organizing services that bring value to users.

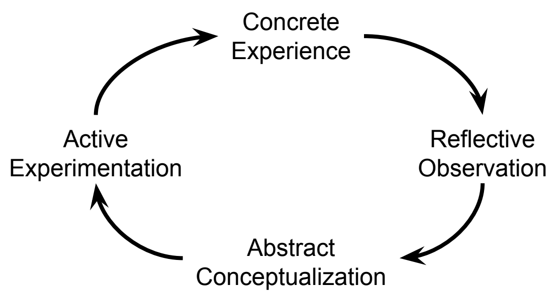


Figure 3.1 – Kolb's experiential learning cycle.

Philosophical assumptions

Research on services through collaborative experiential learning implies studying human phenomena, an endeavor that can be approached from a multitude of perspectives. By conforming to the constructivist and interpretive SEAM systemic paradigm, we make our perspective explicit in terms of the philosophical assumptions that shape and define our way of conducting research. Recall that philosophical assumptions reflect the positions in ontology, epistemology and axiology, which in turn steer the methodology. Let us revise the SEAM systemic paradigm in terms of these three dimensions.

- Ontology explains what reality is made of. In a research method, ontology distinguishes between (1) objective reality (social and physical) that exists independently of humans, and (2) subjective reality that exists only through human action [Orlikowski and Baroudi, 1991]. According to the SEAM systemic paradigm we assume the second position.
- Epistemology explains the way knowledge is constructed and the criteria for it [Mingers, 2003; Orlikowski and Baroudi, 1991]. Applying the constructivist and interpretive SEAM systemic paradigm requires understanding social reality through experiencing the practices and tacit norms shared by people in that reality [Orlikowski and Baroudi, 1991].
- Axiology explains the values. Within interpretivism, and SEAM, there is no value-neutral stance, as researchers' prior assumptions, beliefs and values influence the research [Orlikowski and Baroudi, 1991].

Action research – source of information

Applying the SEAM systemic paradigm, combined with our active involvement in understanding and solving practical problems in the IT department's situational context, perfectly fits the *action research* method. The scope of the collaboration was much wider than the scope of this thesis, it covered the strategic and long-term aspects of the IT department. In our work we use the action research as a method to find relevant practical problems and to gather data for the case studies we present and use as validation throughout this thesis.

Design science research

To tackle the practical problems arising within the case studies from the action research project, we found it necessary to design and to build artifacts that address the problems we observed. We already had experience in using the *design science in information systems research* framework introduced by [Hevner et al., 2004], hence by default we chose it to be our research method. Every chapter presents a designed artifact for service systems modeling: SEAM ontology (Chapter 4), SEAM alignment models and heuristics (Chapter 5), and visualization tool (Chapter 6).

Action design research (ADR)

For the development of the visualization tool, we found it necessary to formally combine design research with action research, as the practitioners with whom we collaborated were directly involved in the artifact development. In our literature review, we came across the *action design research* method, which we adopted for this particular project (see Chapter 6 for details).

In the following sections, we give an overview of design science in information systems research (Section 3.1), action research (Section 3.2) and action design research (Section 3.3).

3.1 Design Science in Information Systems Research

Designing is a human activity that results with an artifact that solves a problem in a perceived reality for which it is created. The observation of this reality is crucial in recognizing the opportunistic and contextual emergence of the newly created things. Let's consider that "humans did not discover fire – they designed it" [Nelson and Stolterman, 2012, p. 11]. Such a claim is incomplete without the findings of Gowlett [2016] who points out that the humans first interaction with fire is most probably after a lightning (opportunistic) that left a burnt landscape with some cooked food, i.e., animals and plants, making the food easy to pick and eat (contextual). The design eventually came from the need to control fire, as fire helped solve various problems, such as nutrition and protection. Human-made creations are the main focus of "The Sciences of the Artificial" [Simon, 1996], a book in which Herbert A. Simon set the foundations for engineering and design disciplines, including design science in IS.

Assumptions of design-science research in IS

One way of understanding design science as a discipline within IS is to analyze several underlying assumptions. First, the word 'design' is a dichotomy: a verb (set of activities, process) and a noun (product, artifact) [Hevner et al., 2004]. Therefore, design-science research encompasses the designed artifact and the set of activities that have to do with the systematic creation of this artifact. Second, within IS, there are continuous discussions over what exactly an IT artifact is: the executing code or the concept behind this code [Baskerville, 2008]. We would argue that it is both, because as a problem-solving research paradigm, design-science research develops innovative artifacts, such as constructs, methods, models and instantiations [March and Smith, 1995], that solve the problem when introduced in the reality. Third, design science is seen as a research paradigm that is different from natural sciences, with its

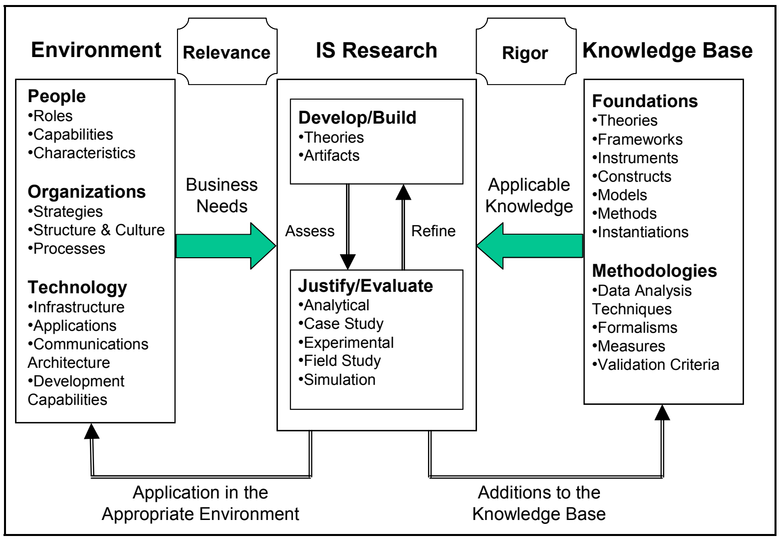


Figure 3.2 – IS research framework. Figure from [Hevner et al., 2004, p. 80].

own philosophical groundings. For example, when two people conduct an experiment in the setting of natural sciences, they should always obtain the same results. Whereas, it is impossible, undesirable and even “wrong” to obtain the same solution to a problematic situation from two designers [Garcia, 2012]. In this thesis, we do not analyze the philosophical groundings of design-science research, rather we present and apply (1) the framework for IS research and (2) the guidelines for design science in IS research, both proposed by Hevner et al. [2004].

IS research framework

Hevner et al. [2004] combine behavioral-science and design-science paradigms in a conceptual framework for IS research (see Figure 3.2). Their framework describes the IS research as an iterative process for the creation, refinement and assessment of an artifact. The business needs that researchers observe in their environment are the basis of the research and assure the *relevance* of the artifact being created. The IS research achieves *rigor* by applying existing foundations and methodologies relevant to the context of research.

Guidelines for design science in IS research

The activities of researchers and practitioners in IS often include designing a “purposeful organization of resources to accomplish a goal” [Hevner et al., 2004, p. 78]. To help with the activities of conducting, evaluating and presenting design research, [Hevner et al., 2004] present seven guidelines for IS researchers.

Guideline 1: *Design as an Artifact* – Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.

Guideline 2: *Problem Relevance* – The objective of design-science research is to develop technology-based solutions to important and relevant business problems.

Guideline 3: *Design Evaluation* The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.

Guideline 4: *Research Contributions* – Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.

Guideline 5: *Research Rigor* – Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.

Guideline 6: *Design as a Search Process* – The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.

Guideline 7: *Communication of Research* – Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Hevner et al. [2004] do not present a complete method for designing an artifact, but we find their guidelines sufficient for steering our research. We used them in the research projects we describe in Chapter 4 and Chapter 5.

A method for design-science research

In their conduct of design-science research in IS, some researchers choose to be supported by methods that include principles, practices and procedures. One widely adopted method is the design-science research methodology (DSRM) for information systems research developed by [Peffers et al., 2007]. DSRM for IS provides formally defined guidance, in the form of a process that researchers should follow, and it gives a mental model used for the presentation of the research outcomes. We find DSRM restrictive because it does not explicitly involve interaction with the practitioners. Consequently, we sought an existing method that includes practitioners. We found action research fits our needs the best, so we incorporated it in the IS research framework of Hevner et al. [2004]. In the research project described in Chapter 5 we used action research in an informal way within the IS research framework proposed by [Hevner et al., 2004, p. 80]. The concrete organizational context of the IT department made our research practice-inspired and relevant.

3.2 Action Research

Collaboration with practitioners

In most positivist approaches to research, researchers act as passive observers detached from the practitioners' reality. Contrary to such approaches, action research is undertaken in a real organizational context, taking part in solving an immediate problem situation in collaboration with practitioners. Coined by Kurt Lewin [Lewin, 1946, 1947], action research is social research combined with "generation of theory with changing the social system through the researcher acting on or in the social system" [Susman and Evered, 1978]. In action research, researchers collaborate with the people being studied [Berg, 2004, p. 197].

Going out in the field

Every organization is a complex whole, composed of interacting entities such as people, other organizations, information systems. We find it impossible to completely analyze and affect these interactions relying solely on quantitative and qualitative information. With "going out in the field", the action researcher observes practitioners and their interactions as a whole entity, thus enriching the quantitative information with qualitative insights about the practice. According to Avison et al. [1999]; Coghlan and Brannick [2014], the researchers' goals for such collaboration are

1. understanding the problem, in the context of a complex organizational setting, and
2. collaborating with practitioners, to introduce a change to a desirable situation, by
3. using existing or developing new theories to be tested.

Multiple iterations

But, action research is not only observation. It is an iterative process where researchers and practitioners act to change an organization and reflect on the effects of their actions [Avison et al., 1999]. The action researcher "is viewed as a key participant in the research process, working collaboratively with other concerned and/or affected actors to bring about change in the problem context" [McKay and Marshall, 2001]. Hence, the action researcher becomes a practitioner who also reflects on his practice. Such a role is also known as participant-as-observer position [Regev et al., 2015]. As described by [Avison et al., 1999], "in action research, the researcher wants to try out a theory with practitioners in real situations, gain feedback from this experience, modify the theory as a result of this feedback, and try it again." Recall that this iterative nature of action research has strong similarities with Kolb's experiential learning cycle (see Figure 3.1).

Patton [1990] categorizes action research as “action-oriented, problem-solving research”, with informal data collection and research publications different from those in basic and applied research. For example, our experience shows that we produced documents that circulated internally among members of the IT department and we also produced academic publications around our collaboration with specific people from the IT department [Popescu et al., 2013; Tapandjieva and Wegmann, 2014; Tapandjieva et al., 2013, 2014, 2016, 2017a].

Results communicated via informal documents

Conducting action research in the context of investigating information systems is not new. For example, Baskerville published a tutorial on an action research of information systems [Baskerville, 1999], where the soft systems methodology of [Checkland and Holwell, 1997] and the ETHICS method of [Mumford, 1993] are rooted in action research. Despite the relevance of action research as a research method in the information systems field, a recent study shows that existing barriers for the publishing of action research in good journals are due to various misconceptions [Avison et al., 2017]. These include views about action research being inappropriate for PhD students and as being less scientific than other research methods [Avison et al., 2017].

Action research in information systems research

Issues with Conducting Action Research

Action research is interpretative in nature [Checkland and Holwell, 1997]. Interpretivism is not widely accepted. For example, some management scientists have a view that the problems addressed by researchers are independent of the observer, meaning there is a clear distinction between the context and the problem [Pidd, 2003, p. 292]. Such a view is called a declaration of independence, which is usually absent when researchers apply what Checkland and Holwell [1997] categorized as soft approaches. Soft systems methodology (SSM) and action research are soft approaches that require the researcher to be mindful of the change he brings to the system he studies [Checkland and Holwell, 1997].

Declaration of independence

Another challenge is the validation of findings obtained during an action research project. The knowledge gained by using action research is difficult to validate in terms of positivist science [Baskerville, 1999]. Note, action research has a different epistemology from positivist science because it creates knowledge dependent on a specific situation. Thus, it is difficult to know ahead of time the consequences of actions taken. Nevertheless, action research is relevant because it solves organizational problems and generates theories grounded in action; actions

Relevance of action research

taken are guided by theory and the theory is supported or revised based on the evaluation of the action's consequences [Baskerville, 1999; Susman and Evered, 1978].

3.3 Action Design Research

Järvinen [2007] identified substantial similarities between action research and design science research. His view about the total overlap between the two is strongly opposed by Iivari and Venable [2009]. The nature of one of our projects (see Chapter 6) required us to combine action research with design science in IS research, hence our quest for a research method that formally and harmonically combines the two, without equaling them resulted in the action design research method proposed by Sein et al. [2011].

A synthesis of action research and design-science research

Sein et al. [2011] point out that "traditional design science does not fully recognize the role of organizational context in shaping the design, as well as shaping the deployed artifact". However, they also mention there are a few researchers who acknowledge "a view of artifacts as emergent from organizational context", hence they propose a design research method that does not separate the IT artifacts from the interaction with the organizational context. Their method is called action design research (ADR), a combination of action research with design science research. This synthesis was a response to the debate about the similarities [Cole et al., 2005; Järvinen, 2007] and differences [Iivari and Venable, 2009] between design-science and action research.

ADR stages

Action design research is a method useful for conducting practice-inspired research, resulting in both practical and theoretical contributions. The advantage of ADR over AR is the production of research results that are generalized outside the organizational context where the research takes place. The ADR process is organized in four stages (see Figure 3.3), where each stage contains a set of principles. We briefly explain the stages, without focusing on the principles.

1. **Problem Formulation** is a stage in which researchers identify, articulate and scope a problem inspired by practitioners, researchers, end-users, technologies or prior research. The challenges in this stage are to secure a long-term commitment for the complete project between researchers and the organization, and to define the research problem as an instance of a class of problems.
2. **Building, Intervention and Evaluation (BIE)** is a stage that is car-

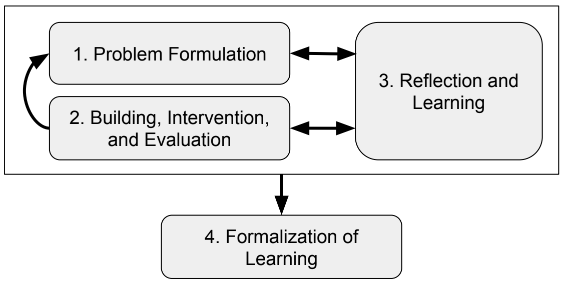

1. Problem Formulation
2. Building, Intervention, and Evaluation
3. Reflection and Learning
4. Formalization of Learning

Figure 3.3 – Stages in action design research. Figure adapted from [Sein et al., 2011].

ried out as an iterative process interweaving "the *building* of an IT artifact, *intervention* in the organization and *evaluation*" [Sein et al., 2011]. In each iteration within this stage, both the problem and the artifact are continuously evaluated. There are two types of BIE, depending on the realized design artifact: IT-dominant BIE and organization-dominant BIE.

3. **Reflection and Learning** occurs in parallel with the first two stages: researchers reflect on the problem formulated, and on the theories and tools chosen to develop a particular solution. The learning from this reflection leads to a refined problem formulation and solution, as both researchers and practitioners gain a better understanding of the emerging artifact. During this reflection, researchers ensure the existence of the knowledge contributions and adjust the research process according to the updated understanding.
4. **Formalization of Learning** is a stage where the learning from the ADR project should result in generalized solution concepts for a class of field problems. This is the most challenging stage, as the ADR output is situated in the concrete organizational context.

Two types of BIE

The form of BIE iterations is customized based on the ADR project. Two types of end-points are identified for the design continuum of BIE:

- An IT-dominant BIE is used when the ADR focuses on the creation of an innovative technological artifact. A prototypical schema of such BIE is depicted in Figure 3.4, where the initial design, a lightweight intervention, is released as an alpha version in the limited context of the ADR team. An updated artifact emerges from the continuous interaction among ADR team members. When the team decides the artifact has matured, a beta version is released into a wider organizational setting, where feedback and evaluation is obtained from a wider audience. The results of this feed-

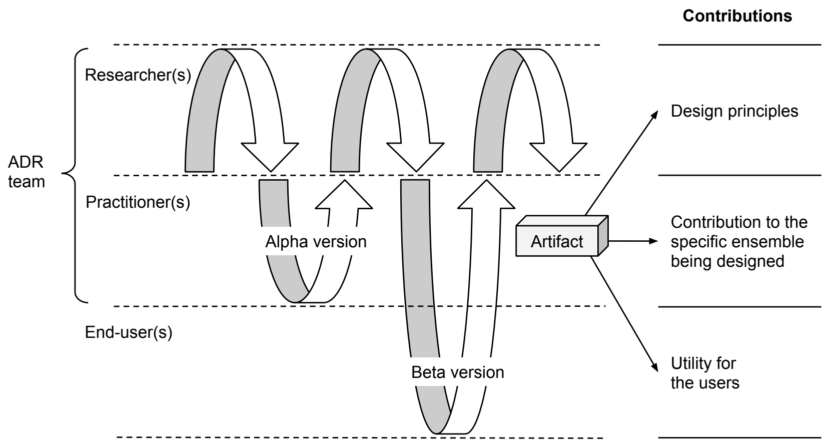


Figure 3.4 – Generic schema for an IT-dominant Building, Intervention and Evaluation (BIE) stage. Figure adapted from [Sein et al., 2011].

back determine the ADR future: completed BIE, or new BIE cycle. These BIE cycles shape the research contributions, organized on three levels (see Figure 3.4): (1) design principles relevant for the researchers, (2) a designed solution to a specific problem relevant for the practitioners, and (3) the utility relevant for the users.

- An organization-dominant BIE is used for innovation by creating design knowledge about organizational intervention. Compared to the IT-dominant BIE, the initial, i.e., alpha, version of the designed artifact is released early for a wide number of organizational participants. New versions are deployed until the organization decides to adopt or reject the artifact.

Research method instantiated in Chapters 4 – 6

In this chapter, we have presented a general overview of the research methods that follow our philosophical assumptions. We use these methods in our three projects that are described in Chapters 4 – 6. The result of each project is an artifact that solves a particular problem we observed. For the development of the artifact we use a separate set of theories, methods and tools. This required us to describe the concrete instance of the research method per artifact built, hence a designated section for a research method per chapter. We instantiate the design science for IS research framework [Hevner et al., 2004] for two projects, described in Chapter 4 (SEAM modeling ontology artifact) and Chapter 5 (SEAM alignment models and heuristics artifact). In both of these chapters, we use action research as a source of information and as a relevant environment for building and evaluating the artifacts. In the third project, described in Chapter 6 (visualization tool artifact), we needed a more formal intertwining of action research with design research, as members of the IT department directly participated in the artifact development. The appropriate method for this project was the action design research method.

4 An Ontology for Service Systems Modeling

Structure of service systems

The inspiration for our work in this chapter came from being involved in modeling the existing and new services of the IT department with the SEAM modeling method. In the process of explaining and describing our service models, we realized we were not following the organizational structure and boundaries [Tapandjieva et al., 2014]. Moreover, we recurrently used only a subset of the existing SEAM ontology [Lê and Wegmann, 2013]. Our exploration of what defines the structure of service systems and our reflection on how people conceptualize service systems, implied reconsidering a new meta-model of the SEAM modeling ontology. The result was a generic, scalable, yet rigorous ontology for the SEAM modeling method and a heuristic about how the structure of the service systems depends on the behavior.

Chapter organization

In Section 4.1, we present the context and our motivation for the work presented in this chapter. In Section 4.2, we present the related work in modeling ontologies. We describe the instance of the research method we follow for designing the ontology in Section 4.3. In Section 4.4, we present an example and formalize the constructs used to build SEAM service models (meta-model and well-formedness rules). We describe the evaluation in Subsection 4.4.5, and we present service-modeling heuristics in 4.5. Finally, we conclude in Section 4.7.

4.1 Context and Motivation

Research question

Through modeling real services in real projects as part of our collaboration with the IT department, we noticed that service systems often do not relate to a pre-defined entity in the reality, such as a department or an organizational unit. For example, what is the service system that embodies an unofficial collaboration between two departments that provide services? What if the collaboration is between two different companies? When the collaboration becomes official, how will the service system be called? This brought us to the research question: *What defines a service system?*

Ontology - the entities and their relations in the perceived reality

Pidd [2003] explains that "a model is an external and explicit representation of a part of reality as seen by the people who wish to use that model to understand, change, manage, and control that part of reality". Before building the model, people first conceptualize the reality they perceive. A conceptualization is formed of all objects, concepts, entities and the relationships among them that are assumed to exist in the reality perceived (see Figure 2.1). [Gruber, 1995, p. 908] defines ontology as "an explicit specification of a shared conceptualization". More concretely, the ontology is "a set of representational primitives with which to model a domain of knowledge or discourse" [Gruber, 2009], thus a modeling language conforms to an ontology.

Context – the SEAM modeling language

The SEAM modeling language [LAMS, n.d.a; Wegmann, 2003] we use was developed in our laboratory LAMS [LAMS] and is a result of over 15 years of research. As suggested by S-D logic and service science, SEAM (1) considers services as the fundamental basis of exchange and (2) models the observed reality as a recursive hierarchy of service systems. When using SEAM, the entities we choose to perceive are systems, where

Definition 1. A *service* is the behavior of a system, observed from the system's environment, that brings value to another system in the same environment.

We find all perceived systems to be service systems, hence we use these two terms interchangeably.

SEAM ontology

In our practice of SEAM modeling, we noticed we did not use all elements defined in the existing meta-model [Lê and Wegmann, 2013]. In addition, this existing meta-model did not provide an answer to our research question. We then sought answers by designing a simplified ontology of the SEAM service-modeling language. In this process, we found that in service systems the emphasis is on the behavior. One contribution of our ontology is that service modelers are encouraged to consider the system structure as being emergent from the perceived or desired behavior of the service system. Note that our goal was not to develop a universal ontology of what exists, rather to develop an ontology that can be used to design and to describe services in a model.

4.2 Related Work

Many different perspectives can be adopted for the management of services [Bardhan et al., 2010]. A service-modeling language is adapted not only to the perspective, but to the motivation and the needs for representing the information around services. In this section, we give an overview of the modeling languages in two of the five proposed perspectives by Bardhan et al. [2010]: computer science and marketing. We find that the second perspective can be further refined into a customer and provider perspective.

4.2.1 Computer Science Perspective on Service Systems

The computer science perspective of services is widely known as service-oriented architecture (SOA) [Rosen et al., 2008] and applies mostly to the usage of software solutions to facilitate the interaction of value co-creation between providers and consumers. Consequently, SOA research and application focuses on technical architecture that orchestrates software services, such as WS-* web services, APIs and RESTful services, in heterogeneous and distributed environments. As a service approach, SOA tries to separate the concern between the service description and implementation [Arsanjani, 2004]. In an SOA context, services are loosely coupled, platform-independent, abstract the implementation and enable interoperability among systems. This enables services to “be combined and used by business processes that may span multiple service providers and organizations” [Georgakopoulos and Papazoglou, 2008].

SoaML

The Service Oriented Architecture Modeling Language (SoaML) [SoaML v1.0.1] is specification project from the Object Management Group (OMG) [OMG] that provides a standard way to design, create and model services within an SOA. The SoaML specification describes (1) a meta-model and (2) a set of extensions to the basic UML model elements, called a UML profile. As SoaML is based on UML, it can be used with existing UML modeling tools. Besides services architectures, showing how services are implemented and used, SoaML models show the encapsulation of interactions between service participants [Amsden, 2014]. Modeling with SoaML fits the model-driven development approach.

XML-based modeling

The Service Modeling Language (SML) [Popescu et al., 2009] and the Web Service Description Language (WSDL) [Christensen et al., 2001], are XML-based modeling languages. The XML files describing the service contain information about the service configuration, deployment, monitoring, etc. These kinds of modeling languages are not graphical, hence not used in people's communication, but they are suitable for task automation, implementing interoperability, communication and exchange between applications, etc. The Unified Service Description Language (USDL) [Cardoso et al., 2010] is also based on XML and aims at unifying the technical and the business perspectives of a given service. The Web Services Business Process Execution Language (WS-BPEL, only BPEL for short) [Rosen et al., 2008] is an XML-based language used to define the coordination and integration of Web Services within higher-level business processes of a company. BPEL is platform independent and provides independence and flexibility by enabling the separation of the business process interaction from the web services [Rosen et al., 2008].

BPMN

The visual representation of the company's processes that are exposed as business services is done with Business Process Modeling Notation (BPMN) diagrams [Rosen et al., 2008]. BPMN is the IT automation of the business and has a notation that is understandable by business analysts, managers and technical developers. It is possible to compile BPMN diagrams into executable BPEL, and [White, 2005] has demonstrated how.

Enterprise Architecture

Enterprise Architecture (EA) is another discipline where services are modeled. We give an overview of ArchiMate®, a widely adopted EA modeling language. The ArchiMate language is based on a meta-model that describes the various concepts and relationships used in modeling an EA, and it also comes with a standard notation for these concepts. As with any EA modeling language, ArchiMate aids technology-integration by creating models that describe the enterprise within and between different domains. The models are used for visualization and analysis. ArchiMate expresses service-orientation with the so-called service layers and service implementation layers that realize the services. Services from a higher layer are linked with and typically use services from the lower layers. ArchiMate identifies three main layers: business, application and technology [Lankhorst, 2009], but the most recent specification includes a strategy and physical layers [Josey et al., 2016]. The concepts used for modeling in each layer have three dimensions: structural, behavioral and internal/external [Lankhorst, 2009, pg. 89].

4.2.2 Marketing Perspective on Service Systems

Even before the introduction of SOA, services existed in the marketing discipline [Hill, 1977] with the focus on the service consumers, such as customers, users and clients. In our context, the marketing/management perspective of services always takes into consideration the value of using IS and IT in the service customization, and uses modeling languages that express service offerings, without over-analyzing the technical implementation. The focus is mostly on understanding the needs of consumers and designing service offerings consumers will value.

e3service

The e3service is an approach for generating service bundles, by using the notion of functional consequences to match the customer perspective and the supplier perspective [Razo-Zapata et al., 2015]. The approach includes an ontology of constructs for service marketing and belongs to the e3family of ontologies for building service networks [Razo-Zapata et al., 2012]. Automated reasoning can be used for finding service bundles that match the customer needs with the service outcomes from the supplier perspective [Razo-Zapata et al., 2015].

Service blueprinting

The service blueprint [Shostack, 1984] is a flowchart of all interactions that belong to the service delivery. Inspired by the design blueprints, it contains four levels of information that help service designers to think and ask questions concerning

- the physical evidence the customer will see and experience,
- the line of interaction that defines interfaces through which the customer-initiated interactions with the service will occur,
- the line of visibility that defines where the service abstraction occurs, thus activities below this line are not visible to the customer, and finally
- the line of internal interaction that defines the support processes.

The service blueprint represents a precise definition enabling service designers to explicitly depict roles of consumers, service providers, and supporting services. Such a definition facilitates the understanding of the subjective ideas around the service concept, through questions about the actions on all four levels of the delivery process.

Canvases

Business models are used for the analysis and design of the business logic of a company [Osterwalder, 2004]. There are several service approaches that have been inspired by the widely cited and broadly applied Business Model Canvas (BMC) [Osterwalder and Pigneur, 2010].

The Service Logic Business Model Canvas (SLBMC) [Ojasalo and Ojasalo, 2015] includes the original nine blocks from the BMC and considers a provider viewpoint ("From our point of view") and a customer viewpoint ("From customer point of view") in each of the blocks. In this manner, [Ojasalo and Ojasalo, 2015] incorporate the service logic into the business-logic principles of BMC. Another approach is the Service Business Model Canvas (SBMC) [Zolnowski et al., 2014] that modifies the BMC to represent co-creation in the model. Similarly to the SLBMC, the SBMC adds a customer perspective. Finally, there is another canvas visualization approach, the Service Model Canvas (SMC) [Turner, 2015a,b], designed by a user-experience professional. The SMC is intended to be used in an early exploration of a service by asking starter questions to organize and document service design thoughts.

4.3 Research Method: Design Science in Information Systems Research

The research method we used in the development of the SEAM service-modeling ontology conforms to the design science in information systems research framework proposed by [Hevner et al., 2004] (see Section 3.1). In Figure 4.1 we depict an instance of this framework to present the environment driving our research cycle and the knowledge base we used. All our activities followed the guidelines presented by [Hevner et al., 2004, p. 82] which have assisted us in understanding the requirements for an effective design-science research. Next, we elaborate how we fulfilled the recommendations in each guideline.

Guideline 1: Design as an Artifact
The SEAM service-modeling ontology artifact is the result of our research that is an updated, flexible, abstract, yet easy to use in constructing service models. The ontology artifact releases the constraint of not having a cycle between hierarchical levels and it emphasizes the behavior in service models.

Guideline 2: Problem Relevance
The business needs we address with our research come from people that belong to two broad categories: academics and practitioners.

- *Academics* – LAMS researchers and LAMS students, are in a constant search of tools and methods that solve problems practitioners have in different domains. Their approach is always based on SEAM by applying a systems thinking perspective with a service-

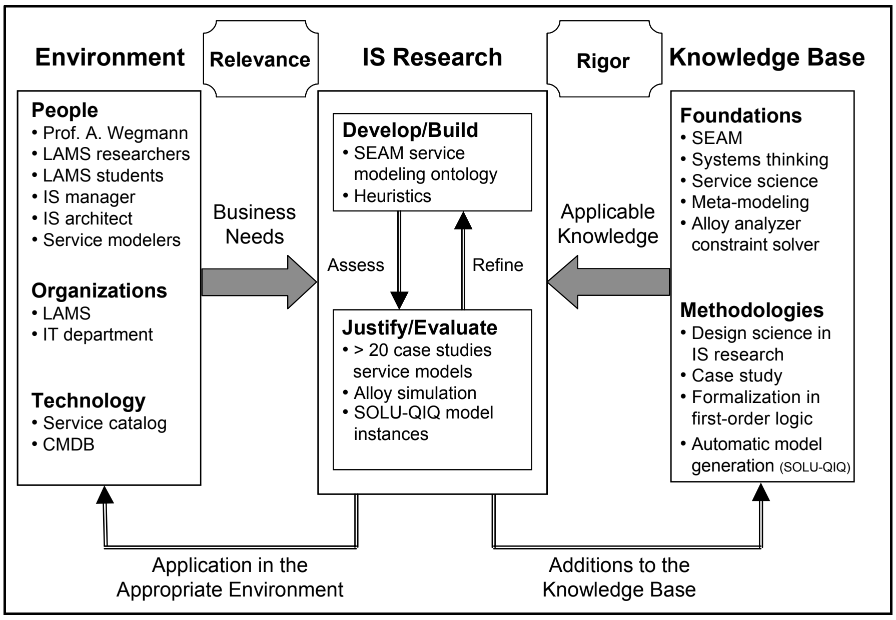


Figure 4.1 – The IS research framework used for building the SEAM ontology artifact. The figure is adapted from [Hevner et al., 2004] to show the concrete environment, knowledge base and the research cycle.

oriented solution. The simplified SEAM service-modeling ontology helps LAMS researchers apply with ease the SEAM method in projects that span multiple domains. It also helps them to define the service system as a collaboration between actors that strive towards the same behavior.

- *Practitioners* – Our research is based on a collaboration with practitioners, an IS manager and an IS architect. We study their perspectives to address their concrete needs:
  - The *IS manager*, has the concerns about how services are provided to the external customers and which entities are involved in the service implementation.
  - The *IS architect*, has the concerns about how to organize service architectures and how architectural designs can be effectively shared among all IT employees, including the IS managers.

  Both of these people were familiar with the SEAM modeling language for internal communication but, before our artifact, they were never exposed to the SEAM meta-model.

Guideline 3: Design Evaluation

We used three techniques to evaluate and justify our artifact (see Section 4.4.5 for details).

1. *Case studies*: Through the course of our research, we interviewed and collected information on around 20 case studies. Each of the cases involved solving a different problem that included service systems where our artifact was used in conceptualizing and modeling the problem situation and solution. In Section 4.4, we show one of them.
2. *Formal model simulation*: We formalized our artifact in a first-order logic language (Alloy [Jackson, 2002]) and simulated it with the Alloy analyzer tool [Alloy, n.d.] to analyze and derive modeling rules.
3. *Automatic model generation*: We implemented our ontology as a meta-model in a tool that generates models automatically as instances of the meta-model. With such an evaluation, we proved the scalability of our artifact. Please refer to [Tapandjieva and Wegmann, 2014] for more details.

Guideline 4: Research Contributions
The main contribution of our artifact is the shift from focusing on systems structure towards focusing on the system behavior, when modeling in SEAM. In the knowledge base, service science and SEAM benefit from a simplistic ontology applicable on all organizational levels. Both domains benefit from the reflection about naming service systems.

Guideline 5: Research Rigor
Our artifact is based on and conforms to the *SEAM systemic paradigm*. Consequently, we applied theories and concepts coming from *systems thinking* and *service science*. To be able to design and evaluate the ontology artifact, we used *meta-modeling* and the *Alloy constraint solver* tool.

Guideline 6: Design as a Search Process
Our search for an effective artifact had a main constraint of being capable of creating SEAM models. To have such an artifact, we built and evaluated our meta-model with LAMS researchers. We also revised the previously created SEAM meta-models [Lê and Wegmann, 2013; Rychkova, 2008]. The modeling rules we list capture the constraints for the correct SEAM instances of the meta-model we propose.

Guideline 7: Communication of Research
The initial artifact was communicated to the academic community via a publication and a poster presentation at a conference (see [Tapandjieva and Wegmann, 2014]). Technology-oriented audiences, namely IS architects, read the informal documentation and description of the arti-

fact. We did not communicate the artifact to the management-oriented audiences in the same form in which it is presented in this chapter. According to our understanding, management-oriented audiences benefit from the visual model that represents an abstraction of their UoD, hence they do not need an additional abstraction in the form of an ontology or meta-model.

4.4 Artifact: An Ontology for SEAM Service Models

In this section, we present the SEAM modeling language, through an example, and then we show the designed ontology in a more formal way. The meta-model with the constructs that comprise the ontology for SEAM service models is presented in Section 4.4.2. Next, the well-formedness rules that are not captured in the meta-model are listed in Section 4.4.3. Afterwards, both the meta-model and the rules are formalized using a declarative language called Alloy [Jackson, 2002] in Section 4.4.4. Alloy generates instances of the meta-model to check if we have over or under-constrained the meta-model with our rules. With Alloy, we verify the correctness of the meta-model and the modeling rules.

4.4.1 Service Modeling with SEAM

SEAM

SEAM is the service-modeling method we use throughout this thesis. It enables us, as modelers, to explicitly show different viewpoints of an organization. Over 15 years of research and practical application of SEAM resulted in modeling principles, heuristics and constructs for representing and analyzing different abstraction levels of systems and behavior. In this section, we present details of the SEAM modeling technique on a concrete example. The theoretical foundations of SEAM are presented in Section 2.3.

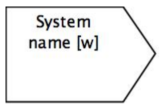
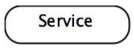
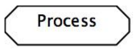
Hierarchy (levels) of service systems in SEAM

SEAM is used to represent an organization as a *hierarchy of systems* (from business down to IT) that provide *services*, where a system refers to entities in the reality perceived: a department, a person, an IT system, or an application [Wegmann et al., 2008].

An abstract view and a concrete view of a system

In the aforementioned hierarchy, we model systems as a whole, also denoted as '[w]' (black boxes) or as a composite, denoted as [c] (white boxes). By modeling a system as a whole, we ignore the system's components and we focus only on the services offered by the system. The system as a whole represents an abstraction in which the focus is on what is relevant for the purpose for which we build the model [ISO/IEC 19506:2012, en]. When we model a system as a composite, the components and their relationships are visible, so we see the implementation of the service and understand the responsibility of each component. In the composite view of a system, we focus on the interaction of the components and on what emerges from combining the components' behavior. A brief overview of the SEAM modeling constructs used in this paper is presented in Table 4.1.

Table 4.1 – SEAM modeling language visual vocabulary.

System	
	*System* – an entity in the perceived reality, such as a company, a department, an organization. A system has two views, whole and composite. A small letter in square brackets, [w] or [c], is added as a suffix to the system name to denote which view is shown.
System behavior [Wegmann et al., 2008]	
	*Service* – the behavior of a system as a whole representing *a service offered by a system.*
	*Process* – the behavior of a system as a composite defining *a service implementation.*
Links	
———	*Use (invoke)* link between services and processes, in a system as a composite. This link means that the process uses (invokes) the connected services.
•———•	*Refinement* (decomposition) link, connecting the abstract and concrete view of a system, [w] and [c] respectively. The services from the system as a whole have corresponding processes in the system as a composite. These processes show the implementation of services.

In the next subsection, we present the Infoscience example where we apply SEAM concepts for creating a two-level service model. The purpose of this model is to illustrate the SEAM modeling process on a real scenario. Note that to keep the model simple, we omit many details.

Example: SEAM Service Model of the Infoscience Tool

Description of the Infoscience example

Imagine reading our university's annual report in which there is a section about the research performance in terms of scientific publications.

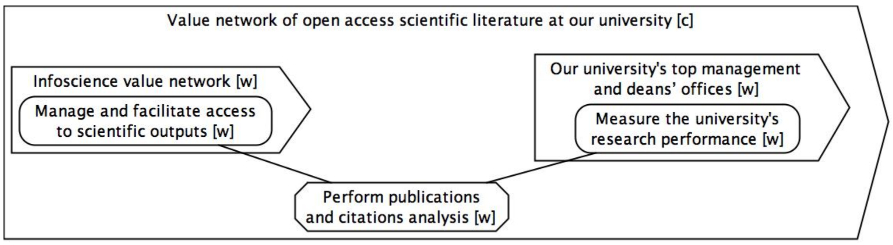

Value network of open access scientific literature at our university [c]
Infoscience value network [w]
Manage and facilitate access to scientific outputs [w]
Our university's top management and deans' offices [w]
Measure the university's research performance [w]
Perform publications and citations analysis [w]

Figure 4.2 – An example of a SEAM service model. The model shows the service of management and access to scientific outputs, as it is used by our university's top management and deans' offices.

How does the university's management measure such performance? The answer lies in Infoscience, a tool that enables managing and accessing scientific outputs. This tool is the result of a partnership between the university librarians and the IT department. The main users of Infoscience are researchers: they use it in the process of archiving their scientific outputs, organizing them, and afterwards, if needed, reclaiming the content. Besides researchers, Infoscience is used by the university's top management and deans' offices in the analysis of publications and citations.

Identifying service systems – first level

In SEAM, such a scenario is conceptualized as follows. Every actor in the case description corresponds to a service system. We do not know the details of the partnership around Infoscience, so it is a black box (system as a whole), that we name *Infoscience value network*. In this case, the name is the choice of the modeler because no such department or organization (Infoscience value network) exists in real-life. The concrete output of this partnership is the service that we call *Manage and facilitate access to scientific outputs*. Similarly, we conceptualize *Our university's top management and deans' offices*, existing entities, as a black box, with the action *Measure the university's research performance*. The actions (services) of these two systems are used in the process that we conceptualize as *Perform publications and citations analysis*. They are also the components of the composite system (the white box) that we name *Value network of open access scientific literature at our university*. Again, this value network is not recognized as an official entity. The resulting model is depicted in Figure 4.2.

Usefulness of SEAM models

We use SEAM to conceptualize the reality into models that are useful. Creating a SEAM model for the Infoscience tool is helpful to the university's top management for understanding how and by whom are publication metrics gathered, processed. As we present the SEAM modeling process, we will omit many scenarios and will develop a generic

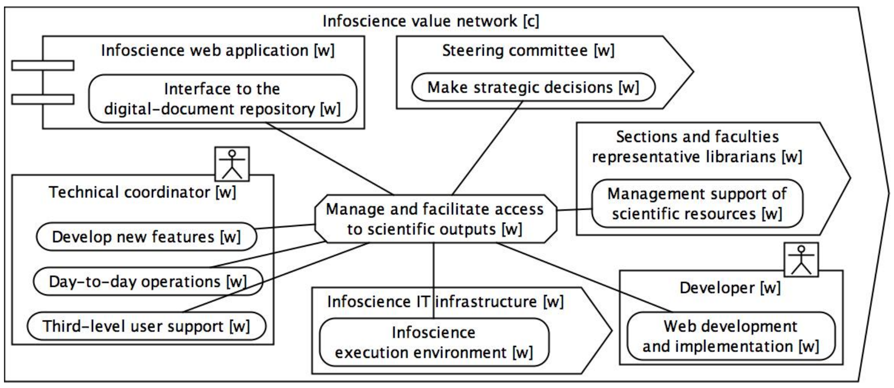

Infoscience value network [c]
Infoscience web application [w]
Interface to the digital-document repository [w]
Steering committee [w]
Make strategic decisions [w]
Sections and faculties representative librarians [w]
Management support of scientific resources [w]
Technical coordinator [w]
Develop new features [w]
Day-to-day operations [w]
Third-level user support [w]
Manage and facilitate access to scientific outputs [w]
Infoscience IT infrastructure [w]
Infoscience execution environment [w]
Developer [w]
Web development and implementation [w]

Figure 4.3 – SEAM model for the second organizational level of the Infoscience project.

model.

Infoscience implementation

Who is involved in the Infoscience value network?

- A **steering committee** that consists of high-level stakeholders and experts, such as the dean of research, the head of the library, the head of an IT unit, the IT systems coordinator, etc. This committee is responsible for *making strategic decisions* concerning the Infoscience project.
- **Representative librarians** for the university sections and faculties: they are responsible for providing *support* to the researchers of the corresponding section.
- The Infoscience **technical coordinator** is responsible for the *day-to-day operations, third-level user support and the development of new features.*
- The Infoscience **developer** is responsible for the *web development and implementation of new Infoscience features.*
- The **Infoscience web application** serves as a *web interface* to the digital-document repository.
- A group of IT infrastructure resources and people provide the execution environment for the Infoscience application.

The *Infoscience value network* – second level

In the second level, we show this *Infoscience value network* system as a composite with a process that implements the mentioned *Manage and facilitate access to scientific outputs.* This process uses all the services from all of the composing systems that, seen as a whole, are part of the *Infoscience value network* system. There is one person, the *Developer,* and one application, the *Infoscience web application,* among these systems. In Fig. 4.3, we illustrate the SEAM service model by showing the interactions and the service exchange between the systems that collaborate in the implementation of the *Manage and facilitate access to scientific outputs.*

No predefined number of levels in the systems hierarchy

In a service-oriented environment, there is no definite number of levels between two systems. We can systematically continue the modeling in the lower levels. For example, the *Infoscience IT infrastructure* system can be expanded to show the service implementation and the composing systems, most of them are IT systems. In some modeling approaches, such as ArchiMate [Josey et al., 2016], there is only one application layer where all applications must be modeled, so our example with IT systems at the second (*Infoscience value network*) and third level (*Infoscience IT infrastructure*) would not be supported. As a consequence, we need a meta-model that allows for scalability in terms of levels. Hence, a taxonomy for the levels is not present in SEAM service models, but modelers are free to embed a taxonomy in their diagrams, as we can do with our SEAM models, and say that there is an end-user level (Fig. 4.2), and several application levels.

Previously developed meta-models for the SEAM modeling language [Bajić-Bizumić et al., 2013; Le and Wegmann, 2005] are complicated and used for the development of tools.

4.4.2 SEAM Meta-Model

Focus on collaboration

Since 2008, researchers have been developing meta-models[1] that cover a larger set of SEAM modeling constructs [Lê and Wegmann, 2013; Rychkova, 2008]. Throughout this thesis we use SEAM models that focus on collaboration among services from different systems, specifically people, IT systems, organizations, companies. Our interest is the value creation through such collaboration, from which new services emerge. Then, we study the surrounding context (upper level) where the new service is used, again, in collaboration with other systems. In a similar manner, we try to understand the upper context, until we have interest in doing so. The example in Subsection 4.4.1 illustrates how we model collaborations with SEAM. Our meta-model contains a subset of entities present in the existing meta-models [Lê and Wegmann, 2013; Rychkova, 2008], that are systems (whole and composite), behavior (service and process) and connections (decomposition and usage). We further explain that these entities are enough for creating correct SEAM models.

[1] In this chapter, we interchangeably use the terms meta-model and information model.

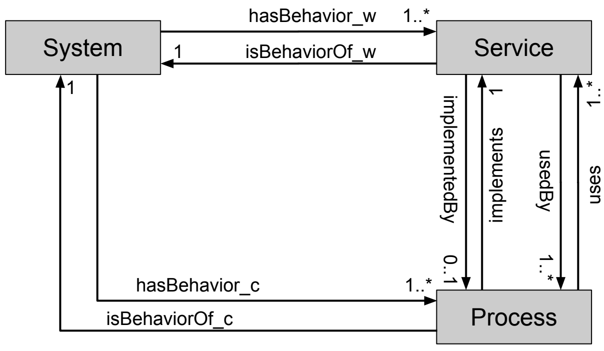


Figure 4.4 – Meta-model of SEAM service-modeling concepts used in this thesis.

SEAM ontology with three entities: System, Service and Process

We built an ontology with the entities: *System, Service* and *Process*. Our proposed meta-model for this ontology is depicted in Figure 4.4. Every relationship between entities has a symmetric pair. We consider the cardinality of the starting entity to be always one. For example, the one-to-many *hasBehavior_w* relationship between *System – Service* means that a system perceived as a whole can have one or more services, and the one-to-one *isBehaviorOf_w* relationship between *Service – System* means that one service can belong to only one system.

Capturing concepts in the meta-model

The concept of a system as a whole is captured in the *hasBehavior_w* relationship, with services being the behavior of systems as a whole. Similarly the *hasBehavior_c* means that the behavior of systems as a composite are processes. The two kinds of links from Table 4.1 are captured in the relationships between *Process – Service*. The *implementedBy* and *implements* stands for the refinement and implementation concept. The usage (invoke) link is captured with the *uses* relationship, telling us that a process can use multiple services. The *usedBy* relationship means that the same service can collaborate in different processes.

Omitting the concepts of a whole and a composite

In the meta-model, we omit entities for the concepts of a system as a whole and a system as a composite: they are represented with the relationships *hasBehavior_w* and *hasBehavior_c* respectively. We also do not include a *System – System* relationship to denote that a system as a composite contains processes and systems as a whole. In short, the systems as a whole that belong to the system as a composite must participate with their services in the process; they are not free-floating in the composite. Consequently, the following query, ***System*** – *hasBehavior_c* – ***Process*** – *uses* – ***Service*** – *isBehaviorOf_w* – ***System***, results in all systems (as a whole) belonging to the starting system (as a composite). Additional details are presented with the modeling rules (Subsection 4.4.3).

From a systems-thinking perspective, the distinction between the concepts of a system as a whole and a composite is based in epistemology; these concepts depend on the observer's (the modeler's) knowledge and relation to the reality when describing the systems. It is the observer who decides the viewpoint he takes when modeling the reality: the context that he knows (system as a composite) where many systems as a whole interact (services connected to a process). In addition, for newly created services, often based on collaboration, there is no formal entity that hosts the process execution. In such cases, the observer derives the system name from the process. In conclusion, we have two reasons for omitting the concepts of a whole and a composite from the meta-model, the first one being pragmatic (they can be computed), and the second philosophical.

Epistemological choice: whole or composite?

4.4.3 Well-Formedness Rules

The well-formedness rules of a modeling language complement the meta-model to ensure the consistency of models. They are also created to avoid the semantic ambiguities that might arise in the instances of the meta-model. We summarize the SEAM well-formedness rules in Table 4.2

Rules for consistency and ensuring the model semantics

Table 4.2 – Well-formedness rules for SEAM service modeling.

Rule	Description	Applies to concepts
R1	A service is unique to one system as a whole, hence two systems as a whole cannot contain the same service.	*Service*
R2	A process is unique to one system as a composite, hence two systems as a composite cannot contain the same process.	*Process*
R3	A process can implement only one service.	*Process* and *Service*
R4	A process must be connected to at least one service. Otherwise, there is no relationship among systems as a whole, and the overall observation of the system as a composite is put into question.	*Process* and *Service*
R5	The decomposition relationship shows the refinement from the abstract view of a system (the whole) to the concrete view of a system (the composite). The service present in the whole, becomes a process in the composite view. Consequently, the process and the service it implements must belong to the same system, hence a process cannot implement a service from a different system as a whole.	*Process* and *Service*
R6	Recursion between levels is permitted. A process can be connected with the service it implements. Such recursion does not have to be immediate, it can occur at any level in the organizational hierarchy.	*Process* and *Service*

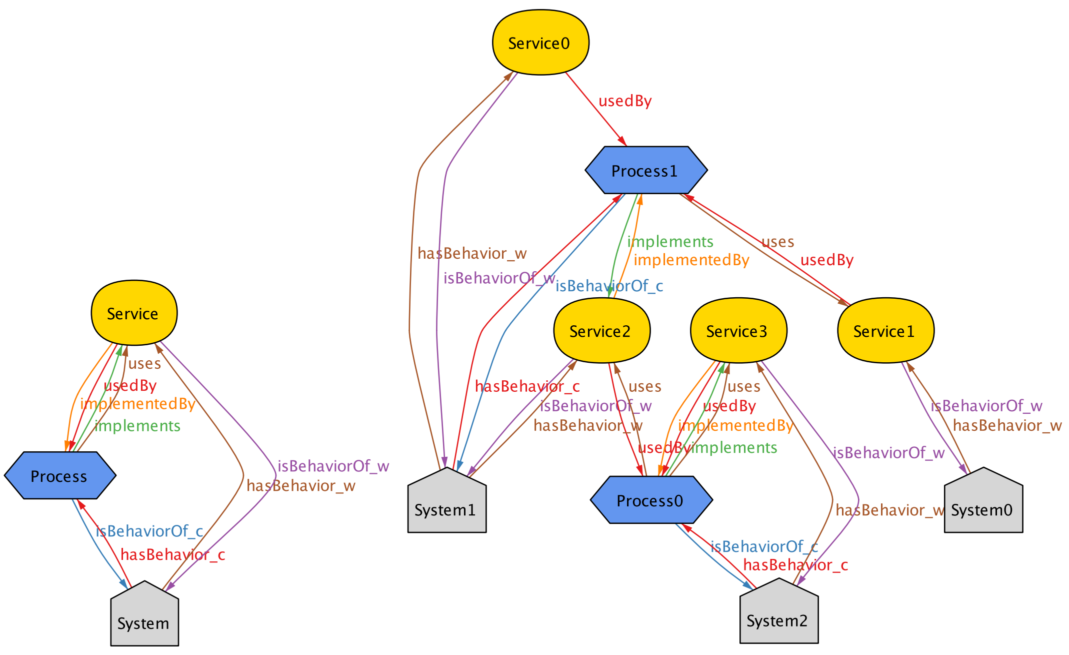
4.4.4 Formalization in Alloy

Alloy [Alloy, n.d.; Jackson, 2002] is a declarative language for modeling structures based on first-order logic. It comes with a constraint solver, the Alloy Analyzer that automatically finds models that satisfy the formulas written with the Alloy language. The Alloy code usually describes basic structures, called signatures, and has detailed constraints applied to the structures. Constraints are expressed in terms of facts, predicates, assertions or quantifiers.

Our use of Alloy

We use Alloy to formalize the SEAM meta-model concepts with signatures (*sig* keyword), and we use facts and quantifiers over the signatures to formalize the well-formedness rules. Then, we use the Alloy Analyzer to generate instance models or counter-examples in a domain we specify. Obtaining meaningful instances would indicate that our formalization is consistent in the domain we have set. The following code shows the complete formalization of the SEAM meta-model and the well-formedness rules.

```
sig System {
  hasBehavior_w: some Service,
  hasBehavior_c: set Process
}
sig Service {
  isBehaviorOf_w: one System,
  implementedBy: lone Process,
  usedBy: set Process
}
sig Process {
  isBehaviorOf_c: one System,
  implements: one Service,                          //R3
  uses: some Service                                //R4
}
fact uniqueServiceInSystem {                        //R1
  no ser: Service, s1: System, s2: System |
    ser in s1.hasBehavior_w and ser in s2.hasBehavior_w and s1!=s2
}
fact uniqueProcessInSystem {                        //R2
  no p: Process, s1: System, s2: System |
    p in s1.hasBehavior_c and p in s2.hasBehavior_c and s1!=s2
}
fact refinement {                                   //R5
  all s: Service, p: Process | p.implements=s =>
    s.isBehaviorOf_w=p.isBehaviorOf_c and s.implementedBy=p
  all p: Process, s: Service | s.implementedBy=p =>
    s.isBehaviorOf_w=p.isBehaviorOf_c and p.implements=s
}
fact symetry {
  all s: Service, p: Process | s in p.uses => p in s.usedBy
  all s: Service, p: Process | p.implements=s <=> s.implementedBy=p
  all sys: System, ser: Service |
    ser.isBehaviorOf_w = sys <=> ser in sys.hasBehavior_w
  all sys: System, p:Process |
    p.isBehaviorOf_c = sys <=> p in sys.hasBehavior_c
```



(a) Alloy generated meta-model **(b)** Alloy generated instance

Figure 4.5 – Alloy outputs based on the formalized SEAM meta-model. The first figure shows the Alloy automatically generated version of the meta-model. Note the similarity with the meta-model in Fig. 4.4. The second figure depicts one Alloy instance where all well-formedness rules are respected.

```
}
run {} for exactly 4 Service, exactly 2 Process, exactly 3 System
```

Listing 4.1 – Formalization of the SEAM service ontology in Alloy

Alloy code

There is a signature (*sig* keyword) for each concept from the meta-model: *System*, *Service* and *Process*. The cardinalities are coded with the corresponding keywords: 1..* is mapped to *some*, 0..1 is mapped to *lone*, and 1 is mapped to one. The well-formedness rules R3 and R4 are already captured with the cardinalities. The remaining rules are written as facts. For example, the R5 well-formedness rule is specified in the fact *refinement*. This fact states that for all services and processes for which a process implements a service, the service is implemented by that process, and both the service and the process are the behaviors of the same system. At the end of the code, we finally specify the domain for which we want the Alloy Analyzer to generate an instance model (the *run* command). The existence of an instance means that the Alloy code is correct and not over-constrained for the domain we have set.

Alloy instance

The Alloy instance in Fig. 4.5 is one of the many that satisfies the constraints written for the well-formedness rules for the domain *exactly 4 Service, exactly 2 Process, exactly 3 System.* Note the immediate cycle that exists between Service3 and Process0. Such cycles capture situations found in reality. We demonstrate this with an example.

Example: hosting service

Imagine the hosting service offered by a data center. Such a service is implemented by using a server room, racks, power supply and other technical resources. In addition, the data center uses a web site to display information about the status of the resources, for monitoring purposes. This website is hosted on a machine in the data center.

Caution with cycles and recursion

A process execution occurs in a specific context, defined by invariants that do not change during execution but depend on the properties of the services connected to the process. As one service has multiple properties, not all of them are used in every process. Going back to the data-center example, the web site showing the data-center status uses the data-center service in a different manner compared to, for example, a mail server. So the data-center implementation captures all the possible properties of its service, whereas processes using that service use a subset of them depending on the context. Hence, recursion is permitted, but it must be used with caution while modeling, as a service is not the 'same' in different contexts.

4.4.5 Evaluation

Formal verification

The Alloy checking is a formal evaluation of the ontology. It proves that correct SEAM models, as the one showed in the Infoscience example can be generated with our meta-model.

Modeled around 20 services

The numerous case studies are another form of an evaluation: They consist of approximately 20 services that we modeled in collaboration with practitioners using our ontology. The difficulties we encountered in naming the systems were easily overcome by focusing on the behavior, i.e., the process and the emerging service. We depict a miniature version of these models in Figure 4.6. Tapandjieva et al. [2014] illustrate in detail two more case studies modeled with our ontology.

Ontology implementation in a commercial tool

We also proved the compatibility of our ontology with the existing SEAM modeling language by implementing it in a commercial tool, called SOLU-QIQ. This tool automatically generates models from a database that conforms to a predefined meta-model [Tapandjieva and Wegmann, 2014]. The SEAM-like visual output shows that using our meta-model as a conceptualization of the reality observed yields correct SEAM models.

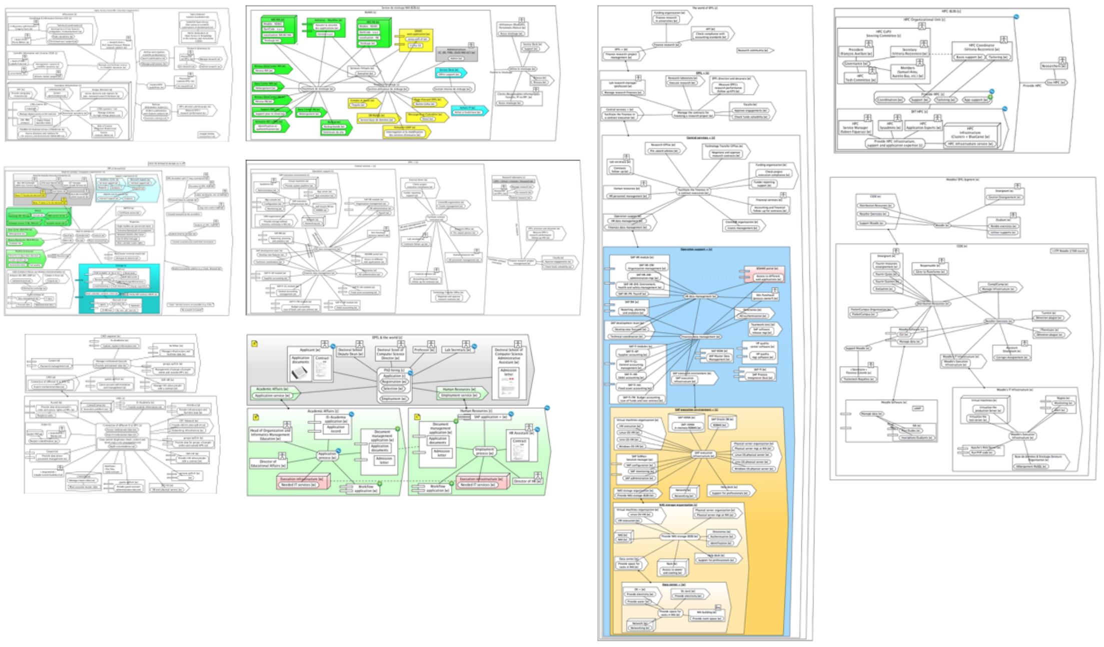


Figure 4.6 – Examples of SEAM service models. .

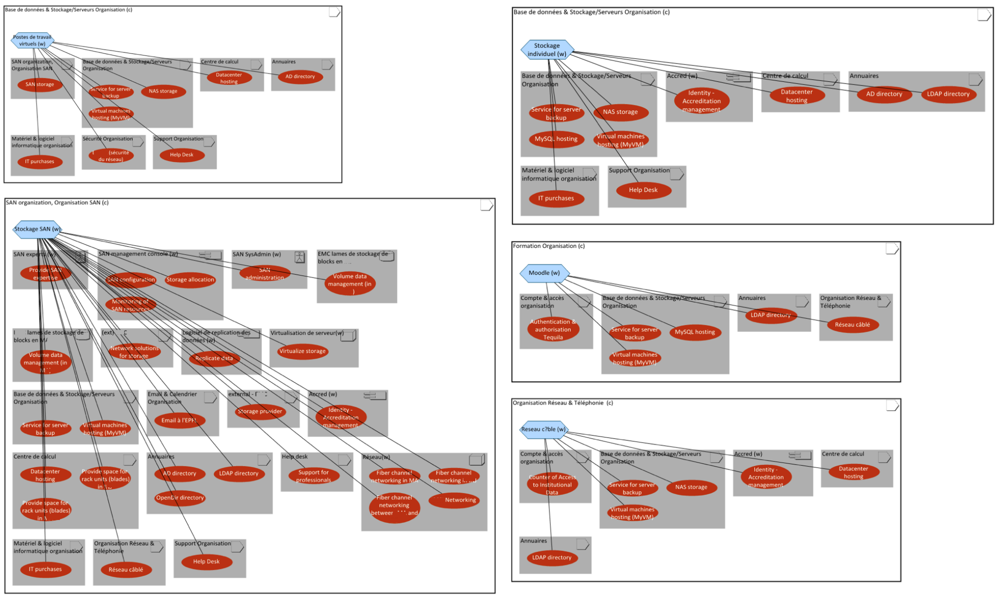

Base de données & Stockage/Serveurs Organisation (c)
Postes de travail virtuels (w)
SAN storage
Service for server backup
NAS storage
Virtual machines hosting (MyVM)
Datacenter hosting
AD directory
IT purchases
Help Desk
Stockage individuel (w)
MySQL hosting
Identity - Accreditation management
LDAP directory
SAN organization, Organisation SAN (c)
Stockage SAN (w)
Storage allocation
SAN administration
Volume data management
Virtualize storage
Replicate data
Email à l'EPFL
Storage provider
OpenLdap directory
Support for professionals
Networking
Réseau câblé
Formation Organisation (c)
Moodle (w)
Authentication & authorisation Tequila
Organisation Réseau & Téléphonie (c)
Réseau câble (w)

Figure 4.7 – SOLU-QIQ generated outputs of SEAM service models. We configured the meta-model in SOLU-QIQ to be the one from Figure 4.4.

4.5 A Heuristic for Designing Service Models Using the SEAM Ontology

Different perceptions of systems and services

SEAM modeling focuses not only on the user perception of the value that the service brings; but by applying the same ontology, it also gives details on the perception of the systems in the multiple organizational levels of the service provider. An implication of using the same ontology at any level of the organization is that the service exchange and value co-creation among systems are present everywhere across the whole organization, not only at the end-user level. Such conceptualization uses the first foundational premise of service-dominant logic: "service is the fundamental basis of exchange" [Vargo and Lusch, 2008].

The concepts of *system as a whole* and *system as a composite* only describe how we choose to perceive the reality, so they are not strictly represented in the meta-model of the ontology. Another reason is that the systems belonging to a system as a composite view, are only there because they contribute in some way to the system's behavior. We find this to be a heuristic technique and describe it next.

4.5.1 Heuristic 1: The behavior defines the service system's structure

During the course of our research, we realized that the existence of systems (more precisely the systems we choose to observe) is strongly dependent on our perception of the behavior of these systems, i.e., services and their implementations. From our involvement in industry projects, we learned that even the systems' names convey information about the behavior of the system (a service or a process). In SEAM, processes show collaboration and value co-creation, hence when we try do define the system with its boundaries where this collaboration occurs, the choice for the system name (1) expresses the function of the collaboration, or (2) relates to an organization, department, or an entity from the observed reality. In any case, it is dependent on the system behavior and it does not strictly follow predefined organizational structures.

4.6 Future Work

Expansion

For future work, the meta-model should be expanded to show properties and functional refinement. We present a first step of this expansion in Section A.2.1 of Appendix A. Additional extensions should include entities to enable various aggregations of services and systems. The need for such extension is justified in the next Chapter 5.

Functional refinement

Another aspect to consider for future work of extending this ontology is the functional hierarchy, more precisely the decomposition of the behavior into sub-services and sub-processes. This can be simply captured with a recursive relation for the Service and the Process entity in the meta-model, but a few more well-formedness rules have to be added to the ontology artifact.

State and properties

In addition, the meta-model does not capture temporal and sequential elements, such as changes of state over time or after an action. This can be done by enriching the meta-model with entities that show properties of systems, services and processes. Previous SEAM meta-models included these concepts [Lê and Wegmann, 2013], and in Appendix A.2.1 we briefly discuss how they are added in our meta-model.

4.7 Chapter 4 Conclusion

Implications for service modeling

SEAM was already used in service-modeling contexts with the previous meta-model that encompasses more concepts. However, in our collaboration with the IT department, we found this meta-model to be overwhelming, which made SEAM difficult to use in large scale projects. In addition, we lacked reasoning about where the structure of service systems comes from. In this chapter, we suggest to use a simplified version of SEAM. For this version, we propose an ontology that offers a new perspective, the focus on the behavior of the system, not the system itself. All the relationships among the Service and the Process concept in the meta-model shift the modeling focus around the behavior, i.e., what systems do together. In such a version, cycles are permitted and even external actors, such as regulators, are considered in a service implementation. We describe a case study where we conceptualized and modeled external actors into a service implementation in [Tapandjieva et al., 2014].

Implications for service organizations

Service science literature discusses mainly one level of value co-creation: the one with the customer or end user. Here, we propose to reuse the same principle inside an organization, by focusing only on what systems do together (behavior). With our ontology, we do not introduce a classification of service providers and consumers. We focus on the interaction, collaboration and co-creation among systems, regardless of their position in the organizational hierarchy. The analysis of the dynamics between a service provider and consumer are systematically applicable on the systems in the service provider, internally, across the whole organization. We can use the existing knowledge on services that is applied at the consumer level, to organize a team, a department, even an entire organization or corporation. To do so, we provide a simplified formalized ontology that we proved can be automated in visualization tools. We present more details on this, in the next chapters.

In Chapter 5, we illustrate the application of our ontology for the alignment of service systems, from IT services all the way to the organization's mission. Such alignment is directly possible from the enabled recursion and linkages between services and processes in the ontology. Then, in Chapter 6, we show the implementation of our ontology in visualization tools and prototypes that are destined for building and sharing people's perspectives on services.

5 Aligning Service Systems within a University

Alignment through the service concept

The idea we explore in this chapter is the use of services to achieve alignment in the functional integration axis. A service is a common concept used in both business and IT contexts. Services have different meanings for a business person (e.g., service economy [Normann and Ramirez, 1993]) and an IT person (e.g., web service and service-oriented architecture (SOA) [Rosen et al., 2008]). However, the fundamental principles behind the service concept, such as abstracting the service implementation from the user, are context-independent, which makes it possible to align the two social realities.

Alignment of perspectives

During our research project, the IT department applied ITIL for service management, and we were confronted with the challenge of extending the use of services beyond the IT boundaries. The project we present in this chapter describes how to use the concept of services across the overall organization, in many different contexts, not only within the IT department. ITIL and other service management approaches focus on procedural details oriented towards the provider's perspective. This hinders the benefits and the overall impact of adopting service-orientation. As SEAM relies on constructivist epistemology, building a SEAM service model means first choosing the reality of interest (e.g., a service provider, user or a collaborator), conceptualizing it and modeling it with the SEAM notation. Such models concretely show different perspectives and facilitate a discussion of alignment between perspectives, including business and IT.

Chapter organization

The rest of the chapter is organized as follows. First, in Section 5.1, we present the context and our motivation for this research collaboration project in which we participated in the building of a service organization. In Section 5.2, we present the related work. Then, in Section 5.3 we describe the research method employed and the sources of information. In Section 5.4, we give an example, specifically the IT department's organization around one service. Through this example, we describe the heuristics behind building a service organization aligned with the main missions. We discuss these heuristics are discussed separately in Section 5.5. Finally, in Sections 5.6 and 5.7, we present the future work and conclusions of this chapter.

5.1 Context and Motivation

IT for a wide community of users

The context of the work presented in this chapter are the information systems within one university that provides services to a wide community. As of 2016, the university had a student body of around 10500 students within seven schools and around 5500 staff members. The management of the university is divided among six main departments: education, research, innovation, human resources and operation, finances and, information systems. More specifically, in this chapter, we describe the alignment between the central IT department and all other departments that have a unit or a group managing any kind of IT resources.

Conflicting values

Although the university's regulation holds the IT department responsible for all IT resources, all units and groups managing IT resources distributed across the university did not report directly to the IT department's management. In addition, the values that motivate the people who belong to the IT groups are different and often conflicting. The central IT department, together with the IT groups in the human resources and finance departments, focuses on compliance and standardization. The IT groups associated with research laboratories focus on the research freedom and diversification.

Federated management

To deal with the conflicting values, while providing quality services, the university's IT management sought a federated way to manage the IT resources. This meant reconciling the conflicting goals of the various IT groups with the interests of the university community, the larger whole to which the IT groups belong. A federated IT organization is a more promising way to achieve alignment compared to centralized or decentralized organization [Luftman and Kempaiah, 2007].

Service – a uniting concept

The members of the university community use IT resources and services in their daily work and practice. According to our understanding, the IT management believed that the concept of a service could unite all IT groups. Services are about providing value to users, by abstracting the implementation details from them (see Section 2.1). It is in the university's interest for its community members to obtain value from the IT, without being exposed to the IT organizational complexities. To provide a coherent set of services that brings value to users, all individual groups that manage IT resources must collaborate, abstract the service implementation details from the users, and focus solely on providing the best value. Among these lines, the central IT department initiated the implementation of the ITIL's best practices for IT service management. First, a service catalog was developed, documenting all services the central IT department provides. Then, a single point of contact (i.e., help desk) was created for providing support to the users who have issues with the services referenced in the catalog.

Questions raised

After ITIL's adoption by the central IT department, the management raised the following questions: *Can services be used and developed across the whole university, within all IT groups? If yes, how do we align them?*

Scope – service to mission alignment

In this chapter, we show heuristics used to conceptualize new levels of services, across the university's IT groups. For aligning the services provided by the different groups, we advise extending the scope of interest beyond one department's or group's boundaries and create service organizations. In service organizations, the hierarchical structure of groups and departments is irrelevant and the focus is only on providing services. We did not develop a detailed step-by-step guide for adopting the service concept across many IT groups, but demonstrated that applying several service heuristics can help to organize resources, collaborate and follow the organization's mission. As Mintzberg [1994] pointed out: "Sometimes strategies must be left as broad visions, not precisely articulated, to adapt to a changing environment". We organized our work to represent only one part of our collaboration: the alignment of services by giving a concrete relationship between the IT services and the mission of the university. This was challenging due to the number of different perspectives that needed to be aligned. The second part of the project, the complete action research project, which we do not show, covered an IT strategy around services, with issues related to the management-reporting structure and the financial structure.

5.2 Related Work

Business and IT alignment

Although the subject of alignment, more concretely business and IT alignment, has been present in the literature for approximately thirty years [Gerow et al., 2015], it still remains at the top of IT management concerns [Kappelman et al., 2017]. Luftman [2003, p. 3] notes that there is no single answer or a silver bullet to ensure business and IT alignment. We find that organizations still strive for business/IT alignment because IT increases the company's competitive advantage through innovation, increased productivity and efficiency [Powell and Dent-Micallef, 1997].

Alignment, explained

Many other terms are used to refer to alignment, such as fit, bridge, integration, harmony, linkage and fusion [Aversano et al., 2016, p. 172]. The definition we adopt for the concept of alignment, or fit, is the "correspondence between a set of components" [Regev and Wegmann, 2004], where depending on the perspective taken, components might refer to choices, goals, behavior, needs, etc. More concretely, these components belong to or describe part of a department, organizational unit, company, IT system; with one word - a system.

Mis-alignment due to different epistemologies

An impediment in reaching this correspondence across systems is the different epistemologies people have that shape the actions about or within systems. In the context of business and IT alignment, Alaceva and Rusu [2015, p. 723] found that business and IT executives use different and unfamiliar languages, creating a barrier between them. This finding is an example of people's different social realities in which separate languages are used. Connecting realities through adopting a common language is one way to achieving alignment. Due to the acceptance of services in many disciplines [Bardhan et al., 2010], we adopt the service concept in order to connect the different realities.

In Section 5.2.1, we present the general concept of business and IT alignment (Section 5.2.1), as introduced by Henderson and Venkatraman [1993] with the Strategic Alignment Model (SAM). This model identifies two axis of alignment, strategic fit and functional integration. Services represent functionality hence, in Section 5.2.2, we focus on approaches belonging to the functional integration axis of alignment. Finally, in Section 5.2.3, we describe two IT service management approaches that can be used as means to achieve alignment.

5.2.1 Business and IT Alignment

Strategic alignment model (SAM)

A widely accepted and recognized description of alignment is provided by Henderson and Venkatraman [1993] with the strategic alignment model (SAM)[1] (See Figure 5.1). SAM identifies four fundamental compo-

[1]As of December 2017, Henderson and Venkatraman [1993] have 4604 citations on Google Scholar https://scholar.google.com/

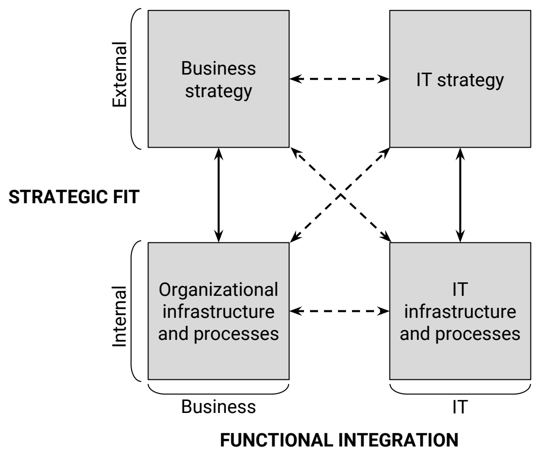


Figure 5.1 – Strategic alignment model (SAM). Adapted from [Henderson and Venkatraman, 1993]

nents, represented with rectangles, and two axes of alignment, strategic fit (vertical) and functional integration (horizontal). Luftman [2004, p. 220] explains that “strategic fit emphasizes the need to make choices that position the enterprise in an external marketplace and decide how to best structure internal arrangements to execute this market-positioning strategy”. The work we present in this thesis does not contain business strategy choices, but it includes the functional integration of infrastructure and processes with the help of services as means to adapt to the evolving strategies, both business and IT. For further overview of the evolution of alignment within the IS field, see Leonard and Seddon [2012].

5.2.2 Functional Alignment

The alignment via functional integration is for the correspondence between business processes and the IT infrastructure that implements them. Such alignment relies on modeling approaches, with the use of a notation understandable by both business and IT people. We present two approaches for modeling the business processes in an organization, an imperative and a declarative one.

Business processes

A business process is an organized and sequential collection of events and activities (automated or manual), decision points, actors (human actors, organizations, or software systems acting on behalf of human actors or organizations), and resources for the purpose of reaching a well-defined outcome [Dumas et al., 2013]. Business processes are first defined and explicitly modeled using the Business Process Model and Notation (BPMN) [BPMN v2.0.2]. Then, they are encoded, automated and executed in a workflow or a business process management system. Such systems enable the analysis, improvement, enactment and inclusion of organizational regulations. Human actors have no flexibility when using business process management systems as they are required to follow the defined steps for achieving the prescribed outcomes [BPMN v2.0.2; Marin, 2016].

Case management

Another, declarative complement to business processes is the case-management approach, represented with the Case Model Management and Notation (CMMN). In the OMG standard, a case is defined as “a proceeding that involves actions taken regarding a subject in a particular situation to achieve a desired outcome” [CMMN v1.1]. Instead of focusing on prescribing imperative detailed steps, case management empowers human actors to achieve their goals by providing access to all the necessary information that, together with what is allowed and disallowed, drive the actions taken in a case [Marin, 2016]. Human actors, who are case participants, direct the resolution of a case with their explicit contextual knowledge and their tacit knowledge from their organization or community.

Except for modeling, there are approaches to alignment where IT is considered as a business on its own, indirectly making IT people adopt the vocabulary of the business environment.

Managing the Business of IT: IT People Adopting the Business Vocabulary

IT4IT Reference Architecture

Inspired from the value chain concept introduced by Porter [2008], the IT4IT™Reference Architecture [IT4IT v2.0] is a standard that introduces a business concept, namely the IT value chain, to describe the work of an IT department. The IT value chain defines a sequence of activities performed by IT which add value to a business service. These activities are “required to design, produce, and provide a specific good or service, and along which information, materials, and worth flows” [Josey et al., 2017, p. 2]. IT4IT also defines five reference architecture levels, providing reference architecture models for only three of them. Services are used within this standard as means to align the IT value chain with the end-to-end overview of the reference architecture model (the service model backbone). IT4IT suggests managing the function of IT as a business in a holistic manner, indirectly applying a business perspective.

Technology Business Management (TBM)

A community of CIOs, CTOs, and other technology leaders from companies including AOL, Cisco, Marriott and MasterCard, formed a council that created a framework to manage the business of IT in an organization by providing transparency of costs, consumption and performance [TBM-Council]. This framework, called Technology Business Management (TBM), is based on a hierarchical taxonomy for describing various cost sources, including hardware, software and services [TBM Taxonomy v2.0]. This taxonomy and distribution of costs is understandable by both IT and non-IT people, assisting the communication across business units and the IT department, ultimately leading to collaboration on business and IT alignment decisions. An instance of a TBM model shows the mapping of costs and allocations of resource consumption across the organizations, thus giving a comprehensive overview to all audiences [TBM Taxonomy v2.0]. TBM is an example of achieved alignment by adopting an intermediate common language that everyone understands, the language of costs.

5.2.3 Services as Means to Achieve Alignment

A specific way of aligning the functional integration is with services. IT service management (ITSM) approaches strive for an alignment with business needs by defining, maintaining and validating the IT value of services. We present ITIL and ISO 20000 as two representatives of ITSM approaches.

IT Infrastructure Library (ITIL)

The IT Infrastructure Library (ITIL) [Cartlidge et al., 2012] is a set of books about the best practices for delivering IT services. ITIL represents a systematic and holistic approach for aligning IT services with the needs of business [Arraj, 2013]. The adoption of ITIL depends on a combination of certifications and trainings, but it is not restricted to them. Any organization can implement ITIL freely, but employee training is recommended to avoid miscommunication about standardized concepts [CIO web]. The current version published in 2011 has five volumes. Each of the five volumes covers a stage in the service lifecycle and describes the various processes in that stage, together with the processes that connect to other stages [Cartlidge et al., 2012].

1. The *service strategy* is focused on understanding the customers and their needs, so that services offered provide value to the customers, while taking into account the service provider's own strategy and culture.
2. The *service design* activities coordinate the design activities by

ensuring that the changing customer requirements coming from the service strategy stage are met with the new or changed service.

3. The *service transition* activities are about planning and managing changes and releases, in order to ensure that new, modified or retired services are aligned with the requirements defined in the service strategy and the service design stages.
4. The *service operation* deals with the actual delivery of the service, ensuring that users get the agreed service level, by focusing on functions such as the service desk, technical management, application management and IT operations management.
5. The *continual service improvement* describes techniques for quality management, change management, and capability improvement, with the goal of improving the complete service lifecycle and ensuring that the value delivered to the customer is maintained.

ISO 20000

ISO 20000 is an international standard for IT service management, first developed in 2005, then revised in 2011, consisting of two parts. The first part specifies requirements for an IT service management system that helps a service provider in the design, transition, delivery and improvement of services [ISO/IEC 20000-1:2011, en]. The second part mirrors the best practices described within ITIL, giving guidance on how to use a service management system [ISO/IEC 20000-2:2012, en].

ITSM tools

The complete service lifecycle is cumbersome to manage, taking into account all the people, technology and business units involved. Organizations implementing ITSM rely on tools that aid in monitoring and regulating the delivery of IT services. These tools are based on various metrics, sometimes from other systems, such as budgets and service outcomes. Features of ITSM tools include problem and incident management, ticketing, service catalog and license management. The tool we encountered while working on our project is ServiceNow [ServiceNow]. The ServiceNow features that were in operation included the service catalog, the incident management, service knowledge base and reporting.

5.3 Research Method: Design Science in Information Systems Research

The research method we used in the development of the SEAM service-modeling ontology conforms to the design science in the information systems research framework proposed by [Hevner et al., 2004] (see Sec-

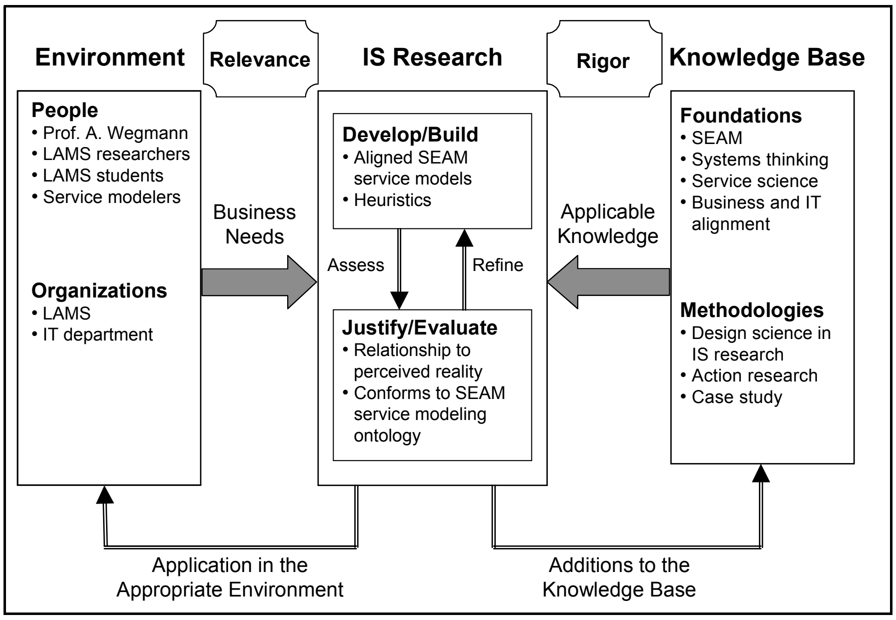


Figure 5.2 – The IS research framework used for building an artifact of aligned service models. The figure is adapted from [Hevner et al., 2004] to show the concrete environment, knowledge base and the research cycle.

tion 3.1). In Figure 5.2, we depict an instance of this framework to present the environment driving our research cycle and the knowledge base we used. All our activities followed the guidelines, presented by [Hevner et al., 2004, p. 82]: they assisted us in understanding the requirements for an effective design-science research. Next, we elaborate how we fulfilled the recommendations in each guideline.

Guideline 1: Design as an Artifact
Our artifact consists of applying the SEAM service ontology, presented in Chapter 4, to build aligned SEAM service models. In the process of developing the alignment, we derived heuristics that facilitate the conceptualization of the reality perceived.

Guideline 2: Problem Relevance
The business needs we address are rooted in the practical problems we observed during our collaboration with the university IT department. In addition, although in an academic environment, the LAMS research group, with its members and students, value an artifact, such as our service-modeling ontology, that demonstrates a specific applicability of SEAM in alignment projects.

Guideline 3: Design Evaluation

One part of the evaluation was conformance to the ontology we propose in Chapter 4. We did not conduct a formal evaluation in terms of a survey or a questionnaire. As our research is based on interpretivist epistemology, we use observations, case studies, informal interviews and feedback.

Guideline 4: Research Contributions
The main contribution of our artifact is the derived heuristics that facilitate the alignment of service models. The application of the heuristics encourages a discussion of concrete services, projects, and perspectives. They embed the SEAM systemic paradigm and are still independent of SEAM models, thus making them compatible with any service approach.

Guideline 5: Research Rigor
Our artifact is based on and conforms to the *SEAM systemic paradigm* and applies constructivist epistemology. In addition, we applied theories and concepts coming from *systems thinking, service science* and *business and IT alignment* fields. Concerning the methodologies, apart from design science for IS research, we conform to interpretative methods for data collection: action research and case study research.

Guideline 6: Design as a Search Process
Our search for an effective artifact had a constraint: be compatible with the previous SEAM ontology while being flexible with the notation.

Guideline 7: Communication of Research
We still have not communicated our aligned SEAM service models and the heuristics to the academic community, but we have presented our approach to IT practitioners in the annual forum meeting of the IT department, as well as to a few people from the same department. We developed a different notation from the one presented in Chapter 4 in order for it to be approachable to wider audiences.

Data Gathering

Due to our collaboration with the IT department, we were involved in real projects. This collaboration emerged into an action research project. Although we do not cover completely the findings of the action research project, we were able to use the data gathered in this alignment research. We took observation notes, conducted in-depth interviews and had access to documentation in Confluence, a knowledge management systems storing materials such as meeting notes, project plans, product requirements, etc.

5.4 Artifact: Aligned SEAM Service Models

In this section, we illustrate how SEAM service models facilitate the discussion of organizing different groups, people and resources around services that can be aligned with a university mission. The example we use is a concrete project that took place from February until October 2015. The project was initiated to solve the maintenance problems of the university's computer classrooms. Through the project's outcomes, we show how a virtualization IT service is coordinated within the teaching mission. We use multiple levels to organize the hierarchy (structure) and the inter-relations of the IT resources, IT groups and people. Such organizations are called service systems or service organizations.

Several definitions

Before continuing the example, we define a few concepts that we will use extensively in our presentation of this alignment example:

- *Service systems* "are value-creation networks composed of people, technology, and organizations" [Maglio et al., 2006]. They also include "other internal and external *service systems*, and shared information" [Spohrer et al., 2007].
- In Chapter 4, we defined a *service (offering)* as the behavior of a system that, observed from the system's environment, brings value to another system in the same environment.
- We call *service implementation* the specific way of organizing the components of service systems that realizes the value brought.

We note that defined around the value-creation, service systems are flexible and group everything that is needed for the implementation of a service, regardless of the organizational structure.

SEAM service models of the example

We illustrate our approach with an example we worked on with the IT department. This example was used in an official presentation of the university's IS strategy at the bi-annual IS forum in May 2016. Everyone managing the IT and IS resources at the university were invited to participate in this forum. As we rework this concrete example, we use SEAM notation style, similar to the one that was used in the official presentation on the annual forum. This style was initially developed by a graphic-design company hired by the university.

Cardinalities of service (systems)

The models of SEAM service systems represent an explicit interpretation of a perspective, e.g., the provider's or user's perspective. In this project, our challenge was to find an appropriate way to model multiple perspectives and instances of the same service (system) in one model. The solution we found was to use signs next to the service (system) name that represent the cardinality, i.e., order of magnitude for the number of instances. We use

- * for 1–9,
- ** for 10–99,
- *** for 100–999, and
- *’*** for 1000–9999.

Description of the Example

Context

At the university, six schools have their own IT groups. These IT groups develop and operate specific IT services for the schools, manage the schools’ collective resources and associated infrastructure, such as computer classrooms, server rooms, storage and backup. With the university’s trend of an increasing number of enrolled students each year, some of these resources are becoming scarce (e.g., space in the server rooms, storage space for research).

Issues

In their classrooms, some professors use computers equipped with the needed software (e.g., simulation software, CAD application) and hardware resources. At the university, there are a total of 25 computer classrooms. Each school’s IT group assigned one computer classroom administrator who was responsible for the management and maintenance of the infrastructure for these resources. After a while, some schools did not have enough computers in their classrooms for all their students. Therefore, the bring-your-own-device (BYOD) [BYOD-Wikipedia] policy was introduced. But then, some of the schools’ IT groups encountered problems with the installation of specific software solutions on the students’ devices. In addition, some of the classrooms had computers that were not maintained. As expected, the professors and students (end-users), were affected by these problems.

Figure 5.3 illustrates the starting point of this project: 25 classrooms, split among six schools (cardinality *), at least one infrastructure environment per school (cardinality *) and six administrators from the schools’ IT groups. We assume that for each course, a different desktop environment is needed, hence the diagram shows the images to load on the virtual machines (cardinality **, for the number of courses per school).

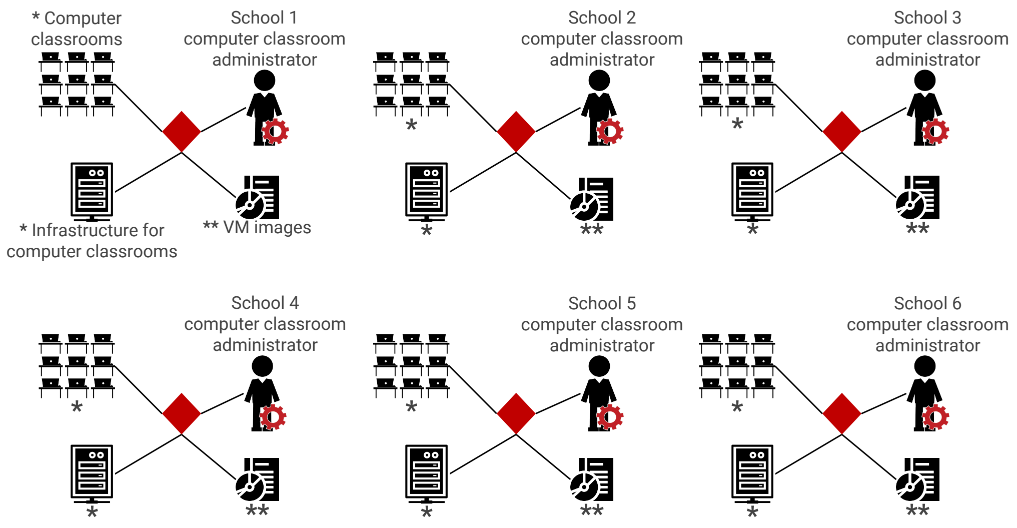

* Computer classrooms
School 1 computer classroom administrator
School 2 computer classroom administrator
School 3 computer classroom administrator
* Infrastructure for computer classrooms
** VM images
School 4 computer classroom administrator
School 5 computer classroom administrator
School 6 computer classroom administrator

Figure 5.3 – Scope of the example: a model of the computer classrooms initial situation. The model depicts six sets of computer classrooms with their own infrastructure and images to load on the virtual machines running on the infrastructure, managed by six administrators from the schools' IT groups.

5.4.1 Virtualization Service

Virtualization as an answer

The question that emerged from this situation was: *"Can the management and maintenance of the infrastructure for the computer classrooms be provided by the the central IT department?"* The central IT department created a study on this subject. The knowledge about the professors' problems from the existing infrastructure, the students' need to run the predefined environment on their personal machines, together with the feedback from the study, resulted in the decision to implement one infrastructure, using virtualization (e.g., VMWare) to support all classrooms. Figure 5.4 sketches how this shared infrastructure could be used by all schools.

Defining the boundary

In Figure 5.4 there are several combinations of where to draw the boundary of the responsibility on the virtualization service system: Are the classrooms themselves part of the infrastructure? Does the IT department manage the images to be loaded on the virtual machines? What is the granularity of services? Boundary identification is often blurry and dependent not only on what the user needs but also dependent on management decisions. It is even more difficult to identify the boundary as service systems often do not relate to the organizational structures, rather to the behavior. In our example, the decision was to provide the virtualization infrastructure only (both hardware and software), without taking ownership of what the virtual machines are running.

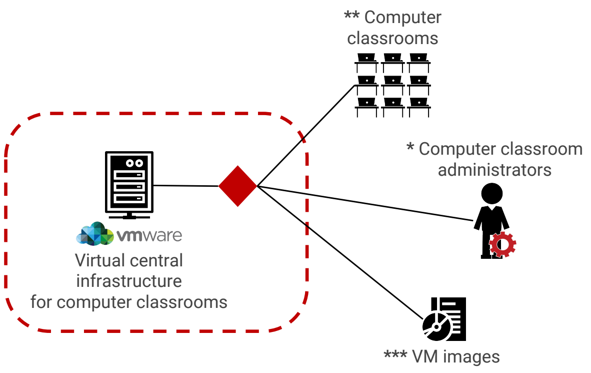


Figure 5.4 – Identification of components for a shared virtualization infrastructure service. This service design was a result from a study carried out by the central IS department. The dashed line defines the boundary of the service system being designed.

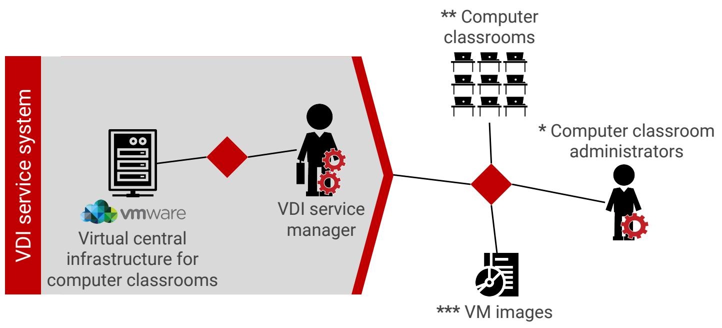


Figure 5.5 – Model of the service system that implements the VDI service. This model represents the viewpoint of the VDI service manager, who knows all details and components of his service.

Design the virtualization service implementation

A project was initiated for designing an implementation for the virtualization service, called Virtual Desktop Interface (VDI): it enabled students to run the software professors used while teaching, in the classrooms, and on their personal devices. The project leader from the central IT department was assigned to be the VDI service manager, once the project was finished, so he could smoothly take over the service operation. All project activities were coordinated with the schools' IT groups, and the central IT department. In addition, it was planned that the university's IT help desk would answer first-level support questions. The details of the implementation, i.e., the viewpoint of the VDI service manager, are illustrated in Figure 5.5.

Separation of concerns

The users of this new VDI service were the administrators of the computer classrooms. For them, the VDI service replaced the problematic infrastructure they managed in each school. Recall that ITIL defines a

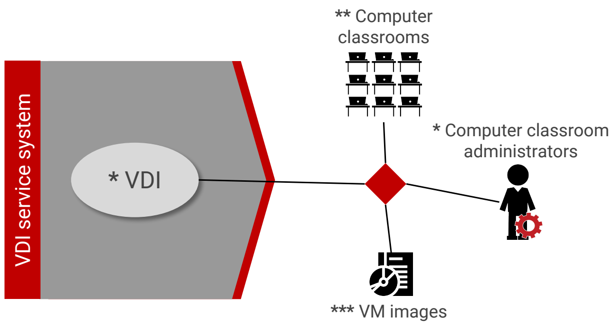


Figure 5.6 – Model of VDI service offering. This represents the viewpoint of the computer classroom administrators who do not know all implementation details of the VDI service, as they see only an abstraction of the VDI service system.

service as "means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks" [Cartlidge et al., 2012]. The outcome of this project was a perfect application of service orientation: separation of concerns into (1) service offering – the value of the VDI service to the computer classrooms administrators and (2) service implementation – managing the infrastructure of the VDI service by the central IT department. The viewpoint of the VDI users, i.e., the IT administrators, is illustrated in Figure 5.6.

Service instances

Despite using the same infrastructure for providing the virtualization service, each school needs a different configuration. Conceptually and practically this yields six instances of the virtualization service. The role of a service manager is to manage and coordinate the service implementation, with respect to all the possible contexts in which his service instances would be used. In this case the VDI service manager has to understand, configure, and to customize the VDI offering for each of the six school different instances. To do so, he had to understand in detail the needs of the classroom administrators and the workflow between them and the professors. The implementation was designed to anticipate such customization, showing that the project/service manager well understood the context (social reality) of the users, the computer classroom administrators. The customized service offering is captured by adding six service instances in the model in Figure 5.6. In this context, customization means that service systems need to be adapted to the particular context and customer problem [Edvardsson et al., 2011].

Inclusion of VDI in the service catalog

When the VDI service was released, as it was provided by the central IT department, it was included in the service catalog. The service catalog did not include all six VDI service instances (one for each school), rather only an abstraction capturing the usage of the same infrastructure.

Aligning perspectives

With this VDI service example, we discussed (1) the service offering perspective about users' needs for custom services and (2) the perspective within the service implementation that is owned by the service manager who coordinates and customizes other services to align with the users' needs. In addition, there is the perspective of the IT management people who care about the overall performance and operation of services that use IT resources. This interplay of different perspectives puts the service manager in a position to coordinate the expectations both of the management (reporting on one service implementation) and of the service users (customizing multiple service instances with the same implementation). Handling these perspectives involves a choice on the level of service granularity to be managed.

5.4.2 Computer Classroom Service

Design of the computer classroom service

The VDI service was not directly provided to the professors and students. As such, the computer classroom administrators had to "repackage" the VDI service by adding the needed software contents and other configurations. As the VDI service manager (project manager initially) had to discover which IT components and technologies to combine in order to provide a tailored VDI service, the administrators also needed to discover what to include in the computer classroom service.

Defining the boundary

This decision defines what is within the service boundary: the VDI service, the configured image to run on the virtual machine and the actual classrooms. Figure 5.7 shows where this boundary is placed, putting the users, i.e., students and professors, outside of the service implementation. Professors require a custom desktop environment for each of their courses; this includes a different set of software products. The computer classroom administrators customize their service offering according to the professors' specific needs. The administrators prepare the images to be loaded on the virtual machines provided via the VDI service. This implies that the number of images and service instances provided by the administrators is proportional to the number of professors using the computer classrooms. There are also some subtle responsibilities assumed by the classroom administrators, such as configuring the computer classrooms for exams, by respecting the exam regulations and providing a backup classroom in case something goes wrong.

Alignment between IT groups

The central IT department and the schools' IT groups agreed that the

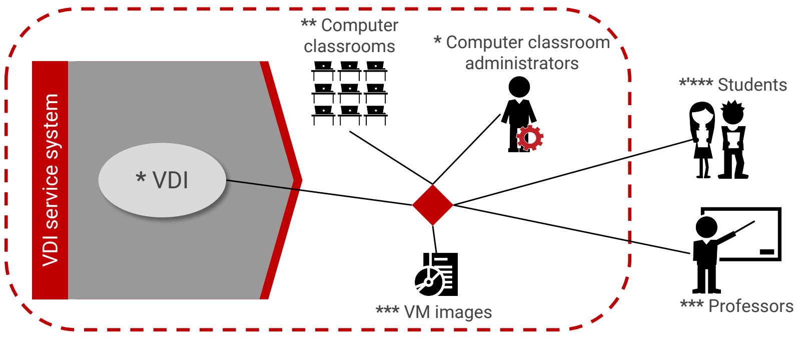


Figure 5.7 – Identification of users and components of the computer classroom service. This service design defines the boundary of the higher-level service system (dashed line).

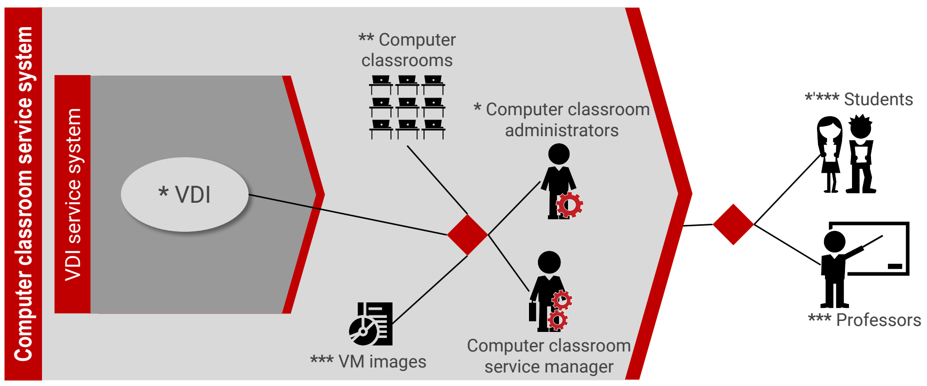

Computer classroom service system
VDI service system
* VDI
** Computer classrooms
* Computer classroom administrators
*** VM images
Computer classroom service manager
*'*** Students
*** Professors

Figure 5.8 – Model of the service system that implements the computer classroom service. This model represents the viewpoint of the computer classroom administrator, in the role of a service manager, who knows all details and components of his service.

administrators of the computer classroom will (1) manage and maintain the software contents on the VDI (images of the virtual machines, access rights), and (2) provide first-level support to professors and students. Such collaboration around the VDI service recognized the fundamental role of a partnership between the central IT department and the schools' IT groups. Although the schools did not use the ITIL approach, such a project was a lightweight adoption of the service-oriented thinking. Figure 5.8 shows the service implementation from the viewpoint of the administrators.

Coordination of services

The value of the model in Figure 5.8 is to show the collaboration necessary between the central IT department that provides the VDI service and the schools that package this service for the end-users. It illustrates the cascading of services (infrastructure – VDI – computer classroom) and how people from different groups reconcile their viewpoints and

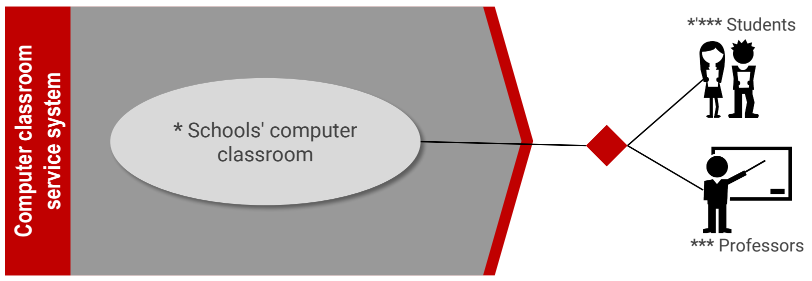

Computer classroom service system
* Schools' computer classroom
*'*** Students
*** Professors

Figure 5.9 – Model of computer classroom service offering. This represents the viewpoint of the users, professors and students, who do not know all implementation details of the computer classroom service, as they see only an abstraction of the computer classroom service system.

organize themselves to guarantee the quality of services. It also shows that schools should coordinate themselves, as a professor could teach the same course in classrooms managed by different schools. This is why we included the role of a computer classroom service manager, someone who similarly to the VDI service manager should coordinate all instances of the computer classrooms. Currently such role does not exist at the university.

Computer classroom service offering

Professors and students access the computer classroom (VDI) service, either directly in the classroom, or on their personal computers, without worrying or knowing how the service is implemented. Their viewpoint is depicted with the model in Figure 5.9. This figure also shows that the computer classroom service follows the same principles as VDI, delivering value to professors and students, without having the ownership of specific costs and risks (as stated in ITIL definition). Even if professors or students might be in direct contact with the VDI service interface, the content they have access to is managed by the computer classroom administrators. Therefore, the overall end-user responsibility is with the administrators and the computer classroom service manager, who align their perspectives of the implementation with the end users' needs for multiple service instances.

5.4.3 Teaching Segment as a Service

Designing the teaching segment

Similar to the way we defined the VDI service and then the high-level computer classroom service, we can continue to define new higher-level services. One reason for defining new services is that the computer classroom service is not used in a vacuum, it would not exist without the coordination of other services used in teaching. We proceed by exploring all services related to the computer classrooms; these services should be tailored and harmonized to the needs of professors and students. Examples of such services, in relation to the initial VDI service include:

– Scheduling computer classroom for courses, exams and student revisions.
– Making specific adjustments to the VDI service for exams in terms of business continuity. More specifically, a backup day is automatically scheduled in the case the infrastructure has a problem during the exam.
– Providing technical support in the form of a help desk (user support), to manage VDI and multi-media issues, such as projector failure.

These services, targeting professors and students, are in the context of teaching, hence we group them in a **segment** called “teaching”. Customer segmentation [Smith, 1956] is a concept from marketing that describes the grouping of customers, based on perceived similarities with respect to customers’ needs, interests, priorities, channel preferences, etc. The perception of the different communities that exist in a university setting helps us to identify a few more segments, such as industry partners, university’s internal users and administration. Figure 5.10 illustrates the segment users and the services we consider within the boundary of the segment.

Segment manager

In our work, a segment is similar to a service system, an aggregation of distinct interconnected service instances from other service systems that, together, form a coherent whole. The segment implementation is depicted in Figure 5.11, and the segment offering in 5.12. As it is the role of a service manager to coordinate and harmonize the implementation in order to provide distinct service instances, we define the role of a segment manager to coordinate the implementation of his segment. More specifically, the segment manager is the specialist in identifying future end-users and he provides user-research data (e.g., survey data) to confirm the needs of the end-users for a service within his segment. This gathered data is the input for defining the Service Level Agreement (SLA) [Hunnebeck, 2011]. Other responsibilities would include, but are not limited to

- promoting the services to the end-users of his segment,
- facilitating the obsolescence of services, and
- managing the relationship with the end-users in his segment (e.g., being in the field, observing, organizing customer satisfaction

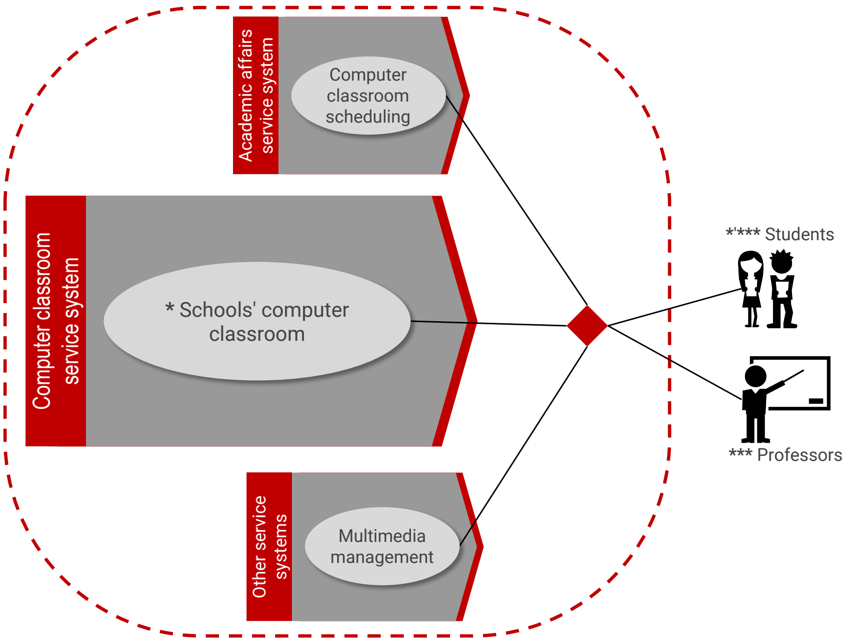

Academic affairs service system
Computer classroom scheduling
Computer classroom service system
* Schools' computer classroom
Other service systems
Multimedia management
*'*** Students
*** Professors

Figure 5.10 – Identification of composing services for the teaching segment. This segment design defines the boundary of the teaching segment service system, identifies users of the services in the segment and defines the segment manager role.

surveys, understanding his segment's culture).

For simplifying the figures, we show only three services in the teaching segment. In practice, the teaching segment could also provide a pedagogical-support service, exam-management service (including all legal aspects relation).

Network or hierarchy of services?

Depending on whether we consider all possible instances of services or not, we have either a hierarchy or a network of services. For example, the exam scheduling service could use an IT system that runs on a virtual machine, accessed via the VDI service. If we consider all possible instances of a service, we have a hierarchy. If we consider the reuse of similar service implementation, then we have a network of services. But even in the hierarchy case, several services can be provided by the same system, not always at the same level, yielding a network of service systems. Again, we are confronted with an epistemological choice that defines how to conceptualize services: all possible instances, having thousands of services, or aggregate services based on choices of what to abstract.

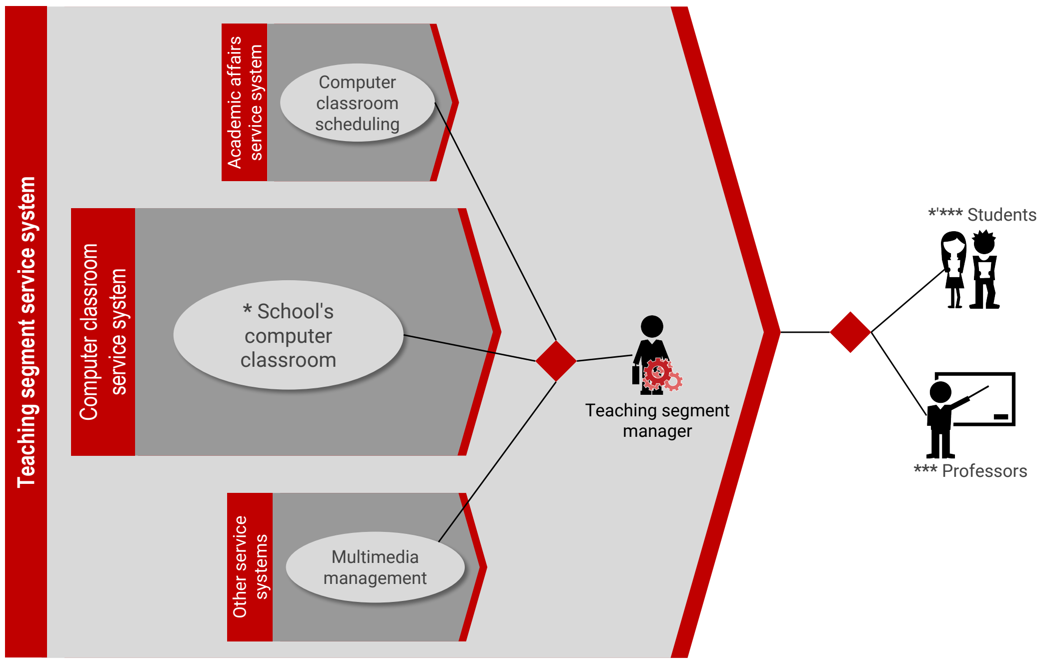

Teaching segment service system
Academic affairs service system
Computer classroom scheduling
Computer classroom service system
* School's computer classroom
Other service systems
Multimedia management
Teaching segment manager
*'*** Students
*** Professors

Figure 5.11 – Model of the service system that implements the teaching segment. This model represents the viewpoint of the teaching segment manager, who similarly to a service manager, knows all details and components of his service.

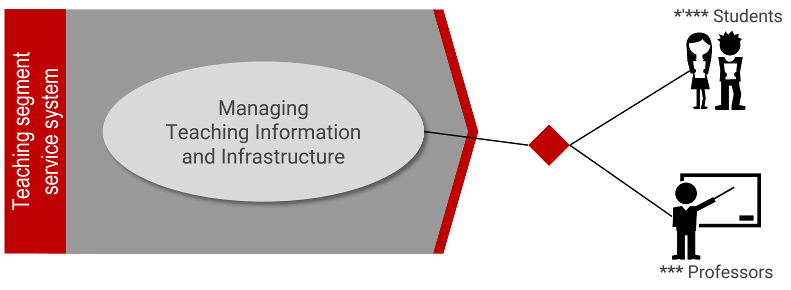

Teaching segment service system
Managing Teaching Information and Infrastructure
*'*** Students
*** Professors

Figure 5.12 – Model of teaching segment services' offering. This represents the viewpoint of the users, professors and students, who do not know all implementation details of the segment services, as they see only an abstraction of the teaching segment service system.

5.4.4 Teaching Mission as a Service

If we consider the prospective students and the alumni as users, other services would appear, such as the promotion of the university and alumni management. Another important service at this level is the provision of Massive Open Online Courses (MOOCs) to the general public. Then, we could consider the university's mission as a service offered to prospective students, the alumni and the general public. Unlike the

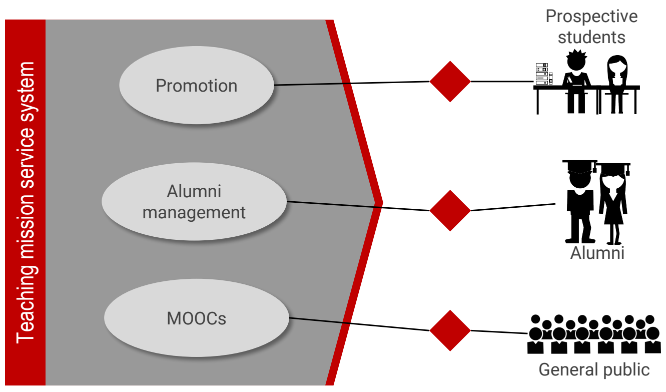


Figure 5.13 – Model of the teaching mission service offering. This represents the viewpoint of the users of the services in the mission who do not know all details of the mission composing services, they see only an abstraction of the services in the teaching mission service system. The users identified are the general public, prospective students and alumni.

different service levels among the VDI, Computer Classroom, and the Teaching segment; the mission is only a different perspective of the services within the university, an external perspective. Although the segment deals with users associated in a formal way with the university, within the 'legal' boundary, the mission targets users outside this 'legal' boundary. With such a perspective, we could also include parents and friends of the prospective students, as well as companies and universities who hire the alumni (or graduated students), the government as a funding organization, etc. Similarly to a service and a segment manager, we define the role of a mission manager, a person responsible for harmonizing all services in his mission. Officially such a role does not exist, but it could be filled by the university's vice president for education. Figure 5.13 depicts our conceptualization of the mission 'service implementation'.

By aligning these three-level services (VDI, MOOCs and teaching mission), we illustrate how IT people can reason about their work and the effect their work has on the university's segment and mission.

5.5 Heuristics for Aligned SEAM Service Models

In this section, we present a formalization of our key learnings in the form of heuristic techniques. The aspects that we cover are not exhaustive, and we do not explore all theoretical groundings. We organize our heuristics around the examples that we encountered during our collaboration with the university's central IT department. To facilitate the discussion, we modified the examples we used from what we observed, but we kept a realistic scenario. We envision using the heuristics for conceptualizing services that lead to an alignment with a mission. The heuristics are rooted in a question related to people's epistemology (what they know), which is in turn influenced by their ontology (what they perceive to exist in reality). These heuristics embrace the existence of different social realities and are the basis of the interpretation of this reality.

5.5.1 Heuristic 2: Validate the service offering by understanding the needs of our users and extending this understanding to the needs of our users' users

Understanding the service use

Let us consider a network connection service; it has the same essential characteristic, connectivity, for all users. But the use of the network is not the same for all; some users might need large amounts of bandwidth and others might need elevated security. Understanding (epistemology aspect) the differences based on the user's location, work habits and practice, would define what (ontology aspect) to include in the network implementation for all users. A good understanding would additionally define the specific configuration elements, such as optical connectivity, settings of the firewall and restrictions. This has an effect on the decisions about an Internet service provider, hardware, software, people's competencies in managing the infrastructure, etc. In our alignment example, the virtualization service was initiated with a study of the computer classroom administrator needs, as well as the needs of professors and students. This study shows the efforts taken to understand one group of users, and the users of this group of users.

Organizing the work of the service user

Service organizations need to organize the work of their users [Normann, 2001, p. 52], so they need to extend their scope by understanding the users' context and work. Even these users have their own users, and consume services to provide a higher-level service. We believe: *it is essential to focus on the benefits of the immediate service users, but service providers need to also organize the workflow of their users, hence to understand the use of the higher-level services (users of users). Looking two levels ahead makes the service alignment explicit.*

5.5.2 Heuristic 3: Contextualize multiple service offerings by configuring one service implementation

Context of service use

ITIL explains that the way service value is defined and differentiated depends on the user [Hunnebeck, 2011]. We find that the context of service use and all possible parameters of a service yield multiple and different service instances. Such differences would enable service managers to tailor the service offering and would influence the choice of components in the service implementation. We tend to forget about different instances: The same service implementation consumed at various locations, or at different times of the day might yield new instances. From our experience, when problems arise, it is always due to a specific set of service properties and parameters, so should we put effort into discovering, understanding and managing **all** possible instances?

Importance of service instances

The university IT department's services are used by thousands of people on a daily basis. Each user wants a service ideally tailored to his needs, but addressing each customer individually would increase the magnitude of services provided. A way to reduce the complexity of dealing with instances is classification [Parsons and Wand, 2008], and the classification we show is based on segmentation [Smith, 1956].

Service segmentation

Segmentation is beneficial for organizing service management. The services designed for one kind of user can be tailored to work best together and they have usually the same kinds of issues to address. For example, all services targeting IT managers need to address business continuity. We define service segmentation as the grouping of IT services and their corresponding service organizations by the kind of users they address. This helps us to adapt services to the skills of their users. For example, a service that is designed for all university's employees needs user support from a help desk. A service that is designed for a very skilled IT professional would need a user support of a higher level. We propose several roles in service management to deal with the two realities: the implementation and the instances of service offerings from one implementation.

Roles in Service Management

Role of a service manager

For one individual service, it is the role of a service manager to take care of the end-to-end service lifecycle. Before putting the service in operation, the service manager negotiates the Service Level Agreement (SLA) with representatives of the service users and organizes the technical implementation. Afterwards, she makes major decisions about the allocation of resources between services. The service manager also creates and maintains supporting documentation for each service. The most important activity is the maintenance of relationships (1) with everyone involved in the service implementation, (2) with the users' representatives, and (3) with all of the segment managers of the segments in which the service belongs. Consequently, service managers know what kind of organization to setup and what the specifics for each kind of service are. This leads to the optimization of the implementation side while providing tailored services for each kind of user, as defined by the segments. In addition, service segmentation is also useful for identifying architectural issues that need to be addressed by the different service managers. For example, all low-level IT services would need an interface with the data centers. This is not the case for higher-level services.

Role of a segment/mission manager

As described in Subsection 5.4, the role of a segment manager is defined to ensure the coordination of the services in the segment. By maintaining relationships with the users of her segment, the segment manager promotes services while gathering requirements for improving existing or creating new services.

5.5.3 Heuristic 4: Define and verify the relationship between the service offering and the service implementation

Steps for defining aligned services

To summarize this section, we propose three steps that apply the previous heuristics in defining one service:

1. Consider the needs of the service's immediate users and those of the users of their users.
2. Define the components and the boundary around them for the service implementation, which enables customization to meet the immediate users' needs.
3. Find (1) an appropriate aggregation of the service offering instances and (2) an appropriate abstraction of the implementation to use in communication with IT management, other IT collaborators and users.

Formal and automatic verification of alignment

In Appendix B, we present our formal approach for automatic alignment, based on properties of services and processes. We first model the properties of the behavior (services and processes) at each service system level and then we formalize the concept of alignment, based on the relationships between these properties in one system level and across levels. This formal and automatic alignment verification approach works with quantitative (non-functional) properties, but we expect it to be easily expanded to any kind of properties.

5.6 Future Work

Beyond alignment: governance, architecture, reporting

The case on alignment presents only a small part of our findings within the action research project. Aspects such as governance, human resources, competencies management, architecture management, and reporting are still to be documented. They all represent one perspective of the service organization. The roles of a service, segment and, mission manager are crucial in the alignment, as they are directly responsible for the use of services. In addition, the dependencies, interaction and collaboration between all managers is still to be defined.

5.7 Chapter 5 Conclusion

Heuristics for reconciling the social realities

Alignment is all about reconciling epistemologies, that include people's vocabulary, background knowledge, values and experience. In this chapter, we illustrated the use of our proposed ontology from Chapter 4 for aligning the perspectives among IT professionals at a university, who belong to different IT groups, with the users of their services. We derive modeling heuristics to reach this alignment. These heuristics are based on asking epistemological questions about the use, management and operation of the service. The answers of such questions confirm our finding that behavior defines the service systems, not the structure; the desired behavior defines what to include in the service system implementation.

Service management

The behavior of each user is unique, hence the service offering needs to be customized based on users' needs. The choice of the components in the service implementation must enable variations for customization. Customizing the service implementation yields an increase in the number of service instances to manage. The challenge is the communication and management of a number of instances, that are supported by the same infrastructure. Therefore, we introduce segmentation, which is the grouping of services, and we introduce service/segment manager roles to harmonize the implementation and the instances.

'Business' and 'IT' alignment does not exist

Following this approach, we notice that we do not make a distinction between business and IT. When taking a service approach, through SEAM conceptualizations we learn that such a separation does not even exist, but there exists separation of perspectives (i.e., social realities) that need to be aligned. Even in aligning the ‘business’ with ‘IT’, the focus is on aligning the expected behavior among service systems, dealing with the implementation of services and the number of instances that arise. Such a task is impossible without a tool, hence in the next Chapter 6 we present a tool that supports the communication of service systems through interactive visualizations.

6 Visualizing Service Systems in a Service Cartography

Mapping services

In this chapter, we describe our use of the previously defined service heuristics in a research project for the development of a service-mapping tool used in services management. Considering the recursive nature of services, together with the challenges of managing numerous service instances, we searched for an effective manner to dynamically build maps that show service connections, aggregations and abstractions. As Weick [1990, p. 3] notes, "in a socially constructed world, the map creates the territory", hence people need to become cartographers and visualize – at the right level of abstraction – their perceptions of services.

Collaboration for building a tool

This project was applied because we were building a service visualization and mapping tool in collaboration with practitioners from the university's IT department. This project took place simultaneously with the department's adoption of service-orientation, where specific roles within service management were defined. Two of these roles included the "service manager" and "segment manager", which we covered in Chapter 5. In addition, the role of an"architect delegate" was introduced to deal with the technical details such as configurations, interoperability and maintenance of information systems. We found that all people having these roles could benefit from mapping services.

Chapter organization

This chapter is organized as follows. First, in Section 6.1, we present the context and our motivations of building a service systems visualization tool. In Section 6.2, we present the related work. Then, in Section 6.3, we describe the research method employed, and we give details on the project iterations. In Section 6.5, we formalize our learning of the project

This chapter is partly based on our work described in a paper published at the 2017 International Conference on Exploring Services Science [Tapandjieva et al., 2017a].

as heuristics. Then, in Section 6.4.4 we discuss the research limitations and challenges. Finally, in Section 6.7 we conclude this chapter.

6.1 Context and Motivation

Services management as a class of problems

Service management denotes the “set of specialized organizational capabilities for providing value to customers in the form of services” [Hunnebeck, 2011]. Properly executed, service management ensures that a service organization’s activities and resources facilitate the “design, transition, delivery and improvement of services to fulfill the service requirements” [ISO/IEC 20000-1:2011, en]. One widely adopted IT service management framework is the IT Infrastructure Library (ITIL) collection of best practices [Cartlidge et al., 2012], but even with these types of frameworks, organizations encounter difficulties in adopting them without problems [Hjalmarsson et al., 2016].

Guidelines on service management in the IT department

Collaborating with the people from the IT department, we became aware that there is lack of vision of the internal service exchange and the alignment of internal services with the IT department’s external stakeholders’ expectations. The existing IT service management frameworks, such as ITIL [Cartlidge et al., 2012], give detailed processes for service management, but they are generic. For example, ITIL defines peoples’ roles with different responsibilities in service management, but it does not provide information on how these roles collaborate, i.e., exchange services. ITIL does not provide solutions on some specific challenges we encountered.

Challenges in service management

One challenge around service management comes from the need to tailor services for specific user segments. Accordingly, one service implementation can be used in several different contexts, so the service manager must configure the implementation for each context. By applying this logic, the number of service instances grows with every context in which a service is used. Services are at the same time recursive, which brings another challenge. For example, a low-level service, such as datacenter hosting, has a web site to display a dashboard for the status of the racks in the data center. This web site runs on a virtual machine that is hosted in the same data center.

Is it an application, process or service?

During this project, as SEAM developers and adopters, we made SEAM sketches to conceptualize some of the immediate problems the IT department was facing. Using SEAM includes a consideration of the service offering, the service implementation and service users. The fact that the SEAM models of a concrete situation in the IT department were incomplete demonstrated that services were not well understood in the department's context. For example, we found that even applications and sequential processes were referred to as services. The service catalog was the best example of this: it contained mainly applications. Near the end of the first iteration, the understanding of services improved, a progress mainly attributed to the SEAM sketches made, as well as an ITIL workshop at the help-desk that happened around the same time.

Service cartography tool

During this project, we discerned the broad spectrum of people who should be involved in service management, holding different roles, belonging to different managerial levels and functional domains. We found that our previous knowledge in the domain of enterprise architecture and SEAM is applicable to the challenges service management brings (multiple levels, viewpoints, use of models for communication), but it was not enough. We anticipated that a service manager, and the people he collaborates with, would need a tool in which different levels of service aggregations are represented, recursive services, as well as specific instances of one service in different contexts. We expected to map hundreds of service models and we knew no EA tool for creating and maintaining such a number of models. Another characteristic of our envisioned tool was the dynamic visualization of services in a map, i.e., cartography. In Section 6.3, we describe the search for an appropriate tool, first by customizing an existing tool, and after experiencing unmanageable difficulties, we proceeded by building a tool ourselves.

Why action design research?

We categorize this project as action research due to its duration (five years) and the active operational role we assumed [Avison et al., 1999; Checkland and Holwell, 1997]. In addition, the research output is a designed artifact, namely the service cartography tool, which makes the project compatible with design-science frameworks [Hevner et al., 2004]. The research method we use is action design research (ADR) [Sein et al., 2011].

6.2 Related Work

We organized the related work in three parts. In Subsection 6.2.1, we describe the domain of enterprise architecture. Enterprise architecture relies on models and the visualization of these models for the communication and creation of a shared understanding between people in an organization. In Subsection 6.2.2, we present existing techniques and challenges found in visualization of models. Finally, in Subsection

6.2.3, we identify the tools that support the creation and management of models.

6.2.1 Enterprise Architecture

Why enterprise architecture?

Enterprise architecture (EA) captures the essentials of the business and IT, and their evolution in a holistic view [Lankhorst, 2009]. Some EA authors use the metaphor of city planning and urbanization [Longépé, 2003], and others use the metaphor of building a house [Zachman, 1987]. The main message of EA is the importance of describing the structures and links that exist in an enterprise, in both business and IT domains. Despite this, we notice a strong bias in most EA approaches towards describing IT resources compared to describing business and people's activities. Examples include the Zachmann framework [Zachman, 1987], the IT4IT Reference Architecture [IT4IT v2.0] and TOGAF [Josey, 2011]. Nevertheless, these approaches conceptualize stakeholders and the stakeholders' viewpoints in different diagram types. We provide an overview of different EA approaches with the diagrams they use and the information they describe. As from the beginning of our project we have used the SEAM approach, we present an overview of SEAM, together with a comparison to the other approaches.

Zachman's enterprise architecture framework

Introduced as a framework for information systems architecture [Zachman, 1987], Zachman's enterprise architecture framework is a systematic taxonomy, specifically a table of concepts. The framework shows five different perspectives (rows) of an enterprise or an information system, and how these perspectives fit together. The inspiration for the perspectives comes from the views (abstractions) an architect has during a design and construction of a building: *planner* (where the scope is the model structure), *owner* (enterprise model), *designer* (system model), *builder* (technology model) and *subcontractor* (components). The framework has six columns that correspond to the different abstractions of the real world: *data, function, network, people, time* and *motivation*. The information put in the columns answers the *what, how, where, who, when* and *why* questions respectively [Sowa and Zachman, 1992]. The complete list of columns correspond to the *"Five Ws and an H"* questions [Singer, 2008] familiar to journalists. A cell of the table can be filled by using different notation and techniques: flowcharts, entity-relationship diagrams, predicate calculus, conceptual graphs, simple English, etc. One major benefit is the possibility to apply recursively the complete framework to any row, thus giving details for the different perspectives of the row [Sowa and Zachman, 1992; Wegmann et al., 2008].

Multi-perspective Enterprise Modeling (MEMO)

The Multi-perspective Enterprise Modeling (MEMO) [Frank, 2002] is an enterprise-modeling method that offers a framework based on set of specialized visual modeling languages. Similarly to the Zachman taxonomy, MEMO's framework is represented as a table with three different perspectives in the rows, with four aspects for each perspective represented as columns. Compared to Zachman's framework, MEMO specifies the modeling languages to use when populating the framework: MEMO Strategy Modeling Language (MEMO SML), MEMO Organization Modeling Language (MEMO OrgML) and MEMO Object Modeling Language (MEMO OML).

TOGAF and ArchiMate

The Open Group Architecture Framework (TOGAF) is a methodological framework providing building blocks and a process model, called ADM, for developing enterprise architecture artifacts in four particular architecture domains:

1. Business Architecture – to describe aspects of the enterprise, such as functionality, motivations, organization and people, business processes, and business services,
2. Data Architecture – to describe the data, master data, how services and processes use data, and how data is transformed, exchanged, created, modified, etc.
3. Application Architecture – to describe individual systems and software to be deployed, the interactions between applications, the relationships between applications and the business processes of the organization,
4. Technology Architecture to describe technical implementations and the operating infrastructure in terms of hardware, software and networks.

The Data and Application Architectures are constructed in one common phase of the ADM, called the Information Systems Architectures. The bias towards IT resources is apparent from the fact that organizational and people's aspects are present only in the artifacts that belong to the Business Architecture, and one artifact (a use-case diagram) belonging to the Application Architecture. The modeling language that complements TOGAF is ArchiMate [Lankhorst, 2009]. The three main layers identified in ArchiMate (business, application and technology) fit perfectly the TOGAF architectural domains considered in phase B (Business Architecture), phase C (Information Systems Architecture) and phase D (Technology Architecture). The most recent specification includes a strategy and a physical layer [Josey et al., 2016] that might be used in multiple phases of the ADM.

URBA

The French companies AXA, FNAC, ORESYS, RATP and SUEZ Lyonnaise des Eaux created an enterprise architecture club, called URBA-EA club[URBA-EA], that relates to the urbanization paradigm. Information systems urbanization (URBA) uses the metaphor of city planning in the context of IT. The reference framework of URBA has four different perspectives organized in levels: business architecture, functional architecture, software architecture and technical architecture. The functional architecture is organized into different types of blocks: zones, districts and plots. The urbanization approach first used the term *cartography of business processes* in the context of enterprise architecture. We found the cartography metaphor inspiring, which directed our research.

6.2.2 Visualization of Enterprise and Service Models

Data visualization

Visualization is often linked with the graphical representation of abstracted data and information from various sources, such as statistics, databases, or other quantitative data. Ideally, data visualizations facilitate the communication and convey information about the source data. Bertin [1983] presents his notable work on guiding information design, including diagrams, charts and cartographic maps. Tufte [1990, 2001] produced his seminal works for representing statistical data, by presenting principles of good and bad examples of information design. Our work is about conveying information on organizing around services, and it was not directly related to the visualization of statistical data. We do however find Bertin's and Tufte's ideas and examples significant for visualization of services in a cartography and, in our last project iteration, we explored their contributions. There is much more to learn on information visualization approaches, so we will keep it for future work.

Existing enterprise model visualizations

We visualize models that represent our conceptualization of reality in terms of services, information systems, and people. The approaches presented in the previous Subsection 6.2.1 visualize different models within a taxonomy (Zachman), or visualize different diagrams with specialized notation (IT4IT and TOGAF) according to the target audience. These diverse representations hinder the connection and navigation between model visualizations. Our initial assumption was that predefined frameworks do not steer people's imagination in using the two-dimensional space for expressing how they conceptualize their services, systems and work in general.

The focus on relationships leads to network visualizations

In visualizing enterprise models we focused on the relationships between entities, rather than the appropriate notation used to represent an entity for a specific audience. Our main quest was to stimulate discussions about the connections between services and people, about who collaborates within one transversal virtual system, and about the impact one service has on another. The notion of nodes, being services or complete service models, connected to other nodes (or models) is inevitably present in our research. Nodes and connections form a graph, or a network. Our search for an artifact in which people could manipulate and visualize a multitude of services and service models was implicitly focused around the concept of service network visualization and navigation. Cardoso et al. [2015] define service network as "a mathematical graph structure composed of service systems, which are composed by nodes connected by one or more specific types of service relationship, the edges".

Network visualization attempts

In 1931, Harry Beck designed London's metro map [Glancey, 2015], which became one of the most famous representations of a network, present at almost any metro system in the world today. Aiming for a similar, simple yet useful representation of a service network for our enterprise models, such as the one of the London metro, we explored various network visualizations (see this chapter's Appendix A.3.1) based on the D3 JavaScript library [Bostock, 2017][1]. One important (for us) feature that was lost in all network visualizations was the context in which services are connected. We explain this in one of the iterations for artifact creation.

Cartography

While working on this project, we found the term *cartography* appealing for our artifact of a network of service models, as it conveys the notion of organized thinking about the spatial reality presented in a cartographic map. People feel comfortable around maps, as the human mind requires spatial relatedness to understand anything [Weick, 1990]. A managerial cartographic map does not necessarily need to be accurate, but it should be perceived as both a useful metaphor and a place to encode and visualize information [Weick, 1990]. By focusing on relations between systems in general, through the services that systems provide, our effort to map the IT department's territory as a multitude of interconnected service models positions our work in a cartography domain. As with any map, we envision the exploration and descrip-

[1]Wang et al. [2015] present an overview of commonly used visualization tools and libraries. They also include D3.js.

tion of enterprise information within service models to be a useful and pleasant visual experience. We needed an appropriate tool to fulfill our vision.

6.2.3 Tools: Repositories and Visualizations

IT service maps and dependencies

There are a few commercial tools [daveirwin1 and Anderson, 2016; ServiceNow-Wiki] that automatically map service dependencies. These tools discern the technical details of services and application deployments, such as the communication endpoint, the address of the server or virtual machine where a service runs. They operate on the level of configuration items and address the IT administrator's needs in finding a root-cause of a technical problem. Consequently, people's explicit responsibilities and collaborations for a specific service are not visualized. Whereas, we visualize all relationships that show collaboration among people, applications and technologies, all the way to the end user. These tools, however, can provide aggregate data for technical services; this data can then be integrated in the service cartography visualizations.

Enterprise Architecture Management

Mega's HOPEX [MEGA] and Link Consulting EAMS [EAMS] are commercial tools similar to our envisioned service cartography. These tools are an Enterprise Architecture Management System (EAMS). Both of them are compatible with TOGAF, and the second one has the feature to show the evolution of the models over time. Enterprise Architect [EA-tool] is different as it has support for BPMN, business, systems and software modeling. It also integrates and connects functionalities such as Model Driven Architecture (MDA) transformations, generation and reverse engineering of source code.

Configuration Management System (CMS)

IT service management practitioners use tools that store and monitor the information about all hardware and software configurations. Such tools are called Configuration Management Systems (CMS). In ITIL, a CMS is defined as "a set of tools and databases that are used to manage an IT Service Provider's Configuration data" [Rance, 2011]. This data includes information about incidents, known errors, changes, processes, formal documentation etc. An example of a CMS tool is ServiceNow [ServiceNow].

SOLU-QIQ

A tool with which we started our cartographic endeavor and that we used for approximately four years was SOLU-QIQ. SOLU-QIQ is a commercial cartographic tool that automatically generates the layout for the data that it stores [AB+Software, n.d.]. Advantageous SOLU-QIQ features are the capability for massive modeling and the fully customizable meta-model. This customization enables SOLU-QIQ to support any modeling framework or approach, thus making it a database to store models. By default, SOLU-QIQ follows the approach of urbanization of IT systems (URBA) [Longépé, 2003]. After (re)defining the meta-model, SOLU-QIQ users must populate the database with data, create a query that filters the data, and define a layout, a visualization in a Scalable Vector Graphics (SVG) format [SVG-Wikipedia], to be applied to the query result[2]. Then, a SOLU-QIQ user links the layouts in a so-called graphical navigation model (MNG). Finally, SOLU-QIQ executes the query, applies the layout to the result, which gives multitude of maps in SVG and, using the MNG, connects these maps together in a static HTML web page. The HTML web page can be then put on a server, making the SOLU-QIQ cartography accessible by anyone on the Internet. In addition, SOLU-QIQ supports various formats of data and model import/export. The tool offers an iterative approach for building a cartography of the information systems. As an output, the tool automatically generates a navigational web site that we use to communicate with people involved in building the systems.

6.3 Research Method: Action Design Research

At the beginning of our research project, we were purely focused on the output, the tool we were building. This tool represented “a purposeful IT artifact created to address an important organizational problem” [Hevner et al., 2004]. It was natural to position our research efforts within the framework of Hevner et al. [2004] for IS design-science research. The research method we finally used emerged after a three-year active collaboration with practitioners from the IT department. We present our work through the lens of action design research (ADR) [Sein et al., 2011], a method that is a blend between action research and design science research.

ADR is carried out in four stages

1. problem formulation,
2. building, intervention and evaluation (BIE),
3. reflection and learning, and
4. formalization of learning

[2]In SOLU-QIQ the layouts are called *modèle de graphe*. They define the maps used in the cartography.

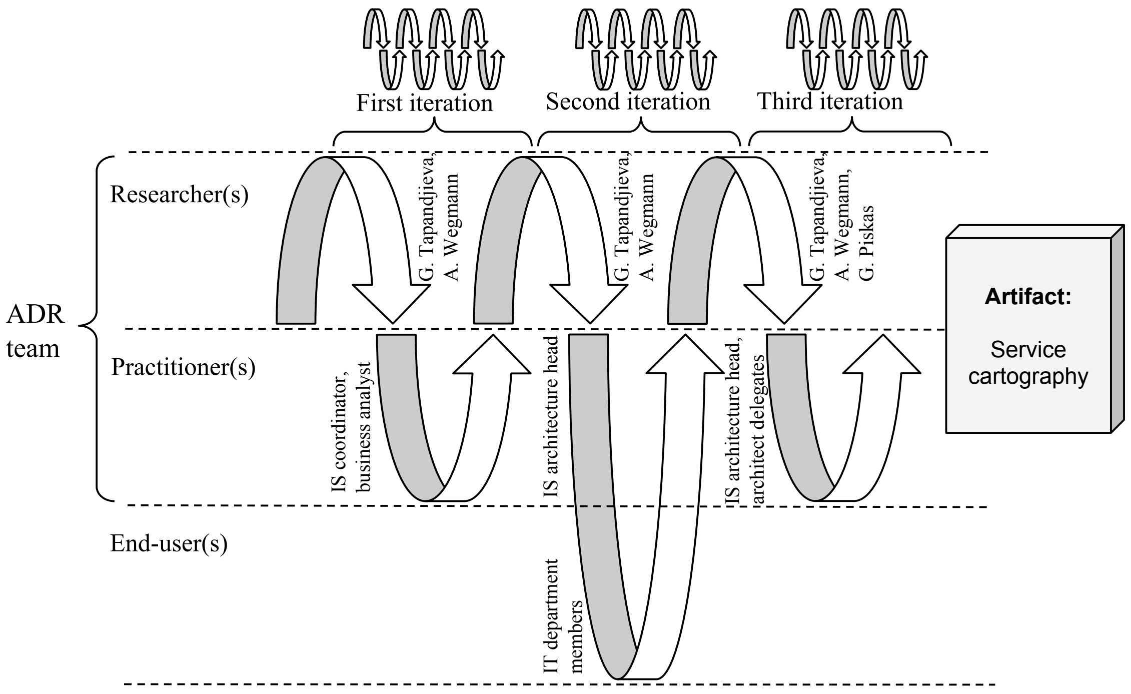

First iteration
Second iteration
Third iteration
ADR team
Researcher(s)
Practitioner(s)
End-user(s)
G. Tapandjieva, A. Wegmann
G. Tapandjieva, A. Wegmann
G. Tapandjieva, A. Wegmann, G. Piskas
IS coordinator, business analyst
IS architecture head
IT department members
IS architecture head, architect delegates
Artifact:
Service cartography

Figure 6.1 – Our three major Building, Intervention and Evaluation (BIE) iterations. These iterations are for the IT-Dominant ADR project we conducted at the IT department.

In Figure 6.1, we depict three major building, intervention and evaluation (BIE) iterations that mark the evolution of the artifact, each having finely-grained, shorter iterations. The timespan we present is a rough estimate, as all iterations were intertwined. As described in [Sein et al., 2011], "during BIE, the problem and the artifact are continually evaluated". In our project, in each major iteration, we gained a better understanding of the problem and we tried to reformulate it. We updated the artifact in parallel to reflect the changing problem addressed. Due to the informal application of ADR, we collaborated, in every major iteration, with different practitioners from the IT department. Accordingly, we did not assume specific roles and responsibilities, neither for us nor for the practitioners. We conducted the BIE iterations in a semi-structured manner where we relied on our observations and discussions with the practitioners.

In the next section, for each BIE iteration we instantiate and describe the first three ADR stages of building the service cartography artifact. Presenting our work in this manner shows the non-linear style of ADR. We illustrate, separately in Section 6.5, the fourth stage, the formalization of our learning.

6.4 Artifact: Service Cartography

6.4.1 First Iteration (November 2012 – May 2014)

Problem Formulation

At the beginning of our collaboration with the IT department, we identified that the IT department's management needed to *build and communicate a common view of the service-oriented enterprise architecture* in the context of the ongoing strategy formulation [Tapandjieva et al., 2013].

Building, Intervention and Evaluation

Map resources and people in different roles

During the first iteration, we focused on modeling the architecture of several business processes, such as the process for hiring new PhD students and for the application for grants. This iteration was used to understand and visualize the internal organization of resources and people in different roles. Our idea was to build a map of IT resources, services, university employees, users, external partners, protocols. As defined in [Woodward and Harley, 1987], maps are "graphic representations that facilitate a spatial understanding of things, concepts, conditions, processes, or events in the human world". Cartography is the practice of making maps. After we adopted the urbanization metaphor, in which people build a cartography of business processes, we named the artifact an IT cartography. We used SEAM to conceptualize the services, hence our idea was to base the design of the IT cartography on SEAM models.

Combine URBA, SEAM and SOLU-QIQ

We chose SOLU-QIQ for the creation of the IT cartography artifact. The SOLU-QIQ tool implements an enterprise architecture approach by defining a meta-model that represents the database schema for the models to be generated. We implemented the first artifact on top of the default SOLU-QIQ meta-model that had predefined layers: technical, application, functional and process. These layers exactly matched the URBA approach [Longépé, 2003]. Since we conceptualized the business processes with SEAM, we sought ways to make the URBA approach compatible with SEAM. Consequently, the visualization result of the models was neither URBA-like (nested rectangles), nor SEAM-like (nested block arrows with ovals and hexagons). Figure 6.2 shows an example of a cartography output during this phase of the first iteration.

Implement SEAM meta-model in SOLU-QIQ

Using the default URBA meta-model enforced a categorization of services into the predefined layers. As SOLU-QIQ is customizable and sup-

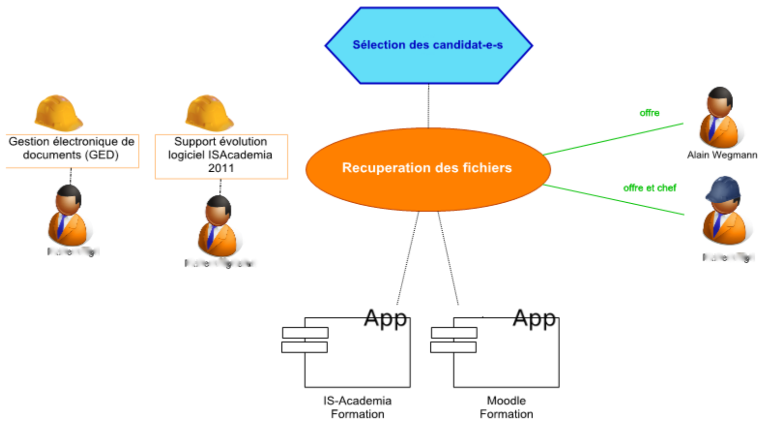

Sélection des candidat-e-s
Gestion électronique de documents (GED)
Support évolution logiciel ISAcademia 2011
Recuperation des fichiers
offre
Alain Wegmann
offre et chef
App
App
IS-Academia Formation
Moodle Formation

Figure 6.2 – IT cartography output in the first iteration. This is taken from [Tapandjieva et al., 2013], showing people and applications involved in one service implementation, with the process that consumes that service.

ports a modification of the meta-model, we modified and implemented the SEAM meta-model (see Section 4.4) to make the cartography compatible with the SEAM methodology [Tapandjieva and Wegmann, 2014]. With the definition of one layout based on the SEAM notation, and linking this layout to itself in a MNG, the SOLU-QIQ cartography generated became capable of showing an infinite number of organizational hierarchy levels, without categorizing them into technical, application, functional or process layers.

How decisions were made

In this iteration, we made all decisions during frequent meetings and semi-structured interviews with two IT department's members: the university's IS coordinator and a business analyst. We also occasionally attended meetings with other university members, to better relate to the problems the IT department was solving.

Reflection and Learning

Our understanding of the ADR project

As university employees, we used IT resources and the services a university's IT department provides on a daily basis, thus enabling ourselves to experience the value these services bring to the community. We also related our experience to the industry and academic service approaches. Despite our regular communication with the IS coordinator and the business analyst, the expectations for the outcome of this project were not defined because, in the first iteration, the main achievement was making the SOLU-QIQ tool and infrastructure work (Microsoft SQL Server, with a virtual machine that runs the SOLU-QIQ desktop application and hosts the HTML web site of the cartography output). But, the vision of creating a shared understanding of what exists within and around the enterprise remained unchanged. Although the problem was formulated around service organization, in the first iteration, we focused on modeling processes and defining peoples' roles in the process. We also studied architectural patterns that are well established among academicians [Ross et al., 2006; Tapandjieva et al., 2013]. These patterns were proposed to the IT department. In addition, the outcome of this research project was targeting all IT employees – a versatile scope of people to whom a one-size-fits-all solution cannot be offered.

Confusion around services

At the end of the first iteration, the IT cartography showed only one process per view and did not show the service offering, or the user. This was obviously a problem because the IT department wanted to be organized around services. In addition, the notation for the cartography visualization was different from the usual SEAM notation. All of this caused confusion among us and the practitioners, as the cartography did not represent the SEAM conceptualization from the discussions.

Before the second iteration

As a result of the reflection,

- we decided to apply the standard SEAM notation, and
- the IT cartography was spontaneously referred to as *service cartography*, with the intention for giving a concrete vision of what service-orientation is, and how to apply it.

Finally, the service cartography built contained only a few example services, hence we believed it was not ready to be shared with all IT people.

6.4.2 Second Iteration (May 2014 – June 2016)

Problem Formulation

We gained a better understanding of the problem: use an IT cartography tool to build maps of the services and share these maps as a means to build and communicate the service-oriented architecture. But first, existing services in the service catalog need to be better understood by both researchers and IT people. This new perspective of the problem is not much different from the first iteration.

Building, Intervention and Evaluation

Improved service understanding

For the few business processes and applications that we modeled with the coordinator and business analyst, we understood well the services offered: the user needs and how to organize the implementation of a service to meet these needs. We modeled one service, the application for grants, at multiple levels, not only at a business process or an application level. We found that partners and regulators are not always only observers or users of the service, they sometimes participate in the service implementation (see [Tapandjieva et al., 2014] for details). In parallel, for this same service, we analyzed the information exchanged, the vocabulary used by the actors, how data is refined from the user level into the implementation level, etc.

Multitudes of services

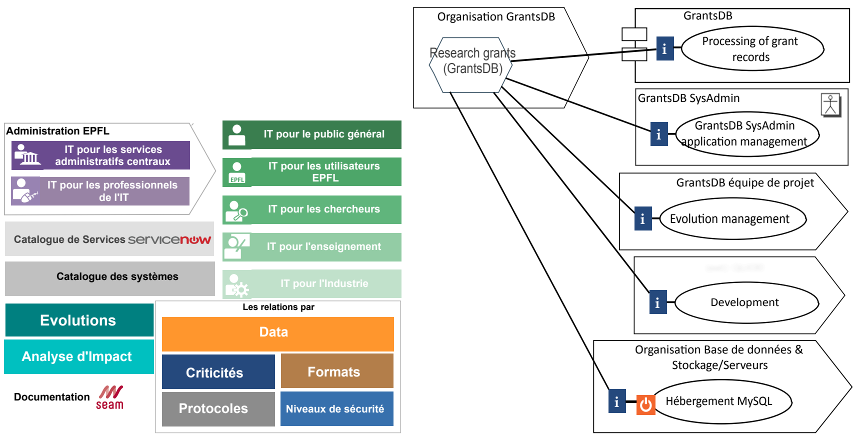
The SEAM sketches showed details of the actions taken by people, of assumptions about what exactly external actors were doing, and we formulated the outcome of an IT application as a service. Actually, every action and function was conceptualized as a service. Consequently, we found that one process or one application was represented with approximately 20 service models that include actions of people and functions of resources, all organized on several levels.

Collaborating with the head of IS architecture

In this iteration, an enterprise architect joined the IT department, as the head of IS architecture. From the moment he joined, on top of his standard work, he began collaborating with us. In the first few months, the collaboration mainly involved knowledge transfer concerning SEAM, SOLU-QIQ and existing services in general. Afterwards, he became the main service cartography user, designer and developer. Together we collaborated on designing the cartography overview page (see Fig. 6.3a) and the navigation between the detailed views. The head of IS architecture was also the main advocate for having the standard SEAM notation in the service cartography. He succeeded in implementing a notation similar to the standard SEAM notation (see Fig. 6.3b). Together, we conducted several interviews with other university members and populated the service cartography with the information gathered. Subsequently, he independently updated the service cartography, and after having the information for many services, he made the tool available to all members of the IT department. In the following months, we observed that no one used the service cartography, despite having the tool at their disposal.

Contextual inquiry with a service manager

The evaluation in this iteration was marked by a contextual inquiry [Beyer and Holtzblatt, 1995] conducted with a service manager. In this one-


Administration EPFL
IT pour les services administratifs centraux
IT pour les professionnels de l'IT
Catalogue de Services servicenow
Catalogue des systèmes
IT pour le public général
IT pour les utilisateurs EPFL
IT pour les chercheurs
IT pour l'enseignement
IT pour l'Industrie
Evolutions
Analyse d'Impact
Documentation
seam
Les relations par
Data
Criticités
Formats
Protocoles
Niveaux de sécurité
Organisation GrantsDB
Research grants (GrantsDB)
GrantsDB
Processing of grant records
GrantsDB SysAdmin
GrantsDB SysAdmin application management
GrantsDB équipe de projet
Evolution management
Development
Organisation Base de données & Stockage/Serveurs
Hébergement MySQL

(a) Starting overview **(b)** One level service implementation view

Figure 6.3 – The service cartography developed in the second iteration

month contextual inquiry, one of us developed a master/apprentice relationship, where the service manager was the master. The idea behind a contextual inquiry is to discover actions as they occur, to allow the service manager to talk about his work as it happens, and not to ask structured questions as in a traditional interview scenario [Beyer and Holtzblatt, 1995]. The goal of this evaluation approach in our project was to learn more about the daily work of a service manager and understand why the service managers did not consult the service cartography. During the contextual inquiry, some observed activities of the service manager were (1) maintaining relationships with end-users and people involved in the service implementation, and (2) producing and updating the service documentation such as service descriptions, service architectures, service change requests, and a knowledge base of services.

Navigation between views and service levels

The output views from the service cartography could be included in various service documentations, but navigating to the specific view of the service of interest was almost impossible due to the implicit constraint of showing one level at a time. With multiple levels, SOLU-QIQ expands all services of one level, thus making the model unreadable.

Reflection and Learning

Reflection on internal services

With the introduction of service orientation, the members of the IT department are expected to separate the "service offering" and the "service implementation". The service offering is the end customer's value-added level, whereas the service implementation is not visible to the end customer and it represents the internal services and resources used to develop the offering. This separation is perceived as the major benefit of service orientation and it focuses solely on the end-customer value. This iteration was marked by our questioning about internal customers and 'internal' services, where internal refers to IT department and other university's central departments.

Expectations of service managers vs senior managers

From the contextual inquiry, we learned that in the service documentation, service managers communicate their internal service-exchange. In this documentation, service managers describe the organization of the collaboration they have with other internal and external people, and the resources they use. From the interview with the IS coordinator, we learned that the senior management would use a service cartography that includes a dashboard of the status and cost of services, and displays maps that help in decision meetings. This use of the service cartography is more than simply for service-documenting purposes. However, we noticed one commonality between the service manager and the IS coordinator: for both of them, the services that exist are defined in the service catalog.

SOLU-QIQ difficulties

The service cartography stores service-to-service, service-to-segment and service-to-person relationships within a defined collaboration context. We became aware that the web pages with this information were difficult to find, were displayed in predefined views (see Fig. 6.3b) and could not be changed. By observing these pitfalls, we realized we did not understand the needs of the users and the constraints imposed by the SOLU-QIQ tool. In addition, the maps generated with SOLU-QIQ were static. Therefore we decided the following:

1. To stop using SOLU-QIQ and design a new employee-centric tool. The new tool should allow and enable the IT department employees to dynamically build their own service map, in order to fit precisely the needs of the employee at a given point in time for a specific purpose.
2. To initiate a frequent collaboration with one specific role, the architect delegate, in order to capture role-specific use cases.

6.4.3 Third Iteration(June 2016 – February 2017)

Problem Formulation

How can an employee of a service-oriented organization visualize and communicate her work? Based on our observations, an employee's

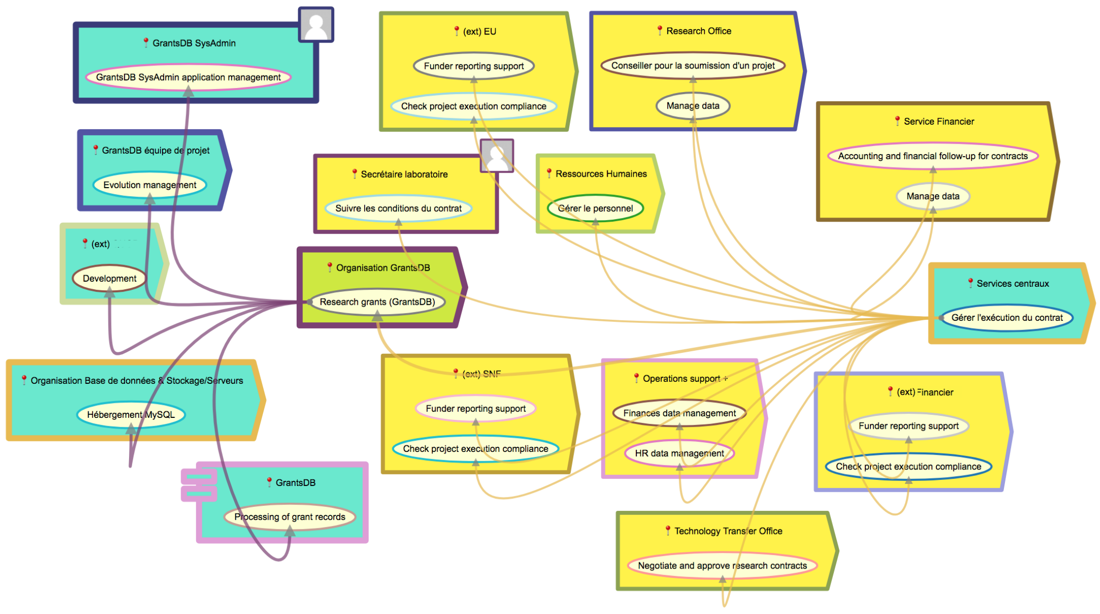

GrantsDB SysAdmin
GrantsDB SysAdmin application management
GrantsDB équipe de projet
Evolution management
(ext)
Development
Organisation Base de données & Stockage/Serveurs
Hébergement MySQL
GrantsDB
Processing of grant records
(ext) EU
Funder reporting support
Check project execution compliance
Secrétaire laboratoire
Suivre les conditions du contrat
Organisation GrantsDB
Research grants (GrantsDB)
Research Office
Conseiller pour la soumission d'un projet
Manage data
Ressources Humaines
Gérer le personnel
Service Financier
Accounting and financial follow-up for contracts
Manage data
Services centraux
Gérer l'exécution du contrat
(ext) SNF
Funder reporting support
Check project execution compliance
Operations support +
Finances data management
HR data management
(ext) Financier
Funder reporting support
Check project execution compliance
Technology Transfer Office
Negotiate and approve research contracts

Figure 6.4 – The service cartography visualization tool in which users build their own map. The figure shows two service levels distinguished by the different colors of relationships. The service level of Fig. 6.3b is depicted with the blue systems on the left side.

work includes exchanging internal services and the value these services provide to her external stakeholders (customers, suppliers).

Building, Intervention and Evaluation

Develop new tool

In the third iteration, we developed a new service cartography that enables members of the IT department to communicate the internal service-exchange. The new service cartography is user centric. Instead of navigating between predefined views, the cartography users search for the services or systems and interactively build service maps they need, starting from an empty canvas. Also, the IT department members are independent from one another while dynamically building their map. The service maps they create can then be saved, exported and shared with other university members. In addition, there is no restriction on the details shown: The map can show multiple service levels, starting from the lowest service level (e.g., network), all the way to the business services, and end-user level. For example, Fig. 6.4 shows two service levels, and it can be expanded. Furthermore, the service cartography has links to the service catalog and the university's directory. Additional features include a few predefined overviews, based on data of the IT members' timesheets:

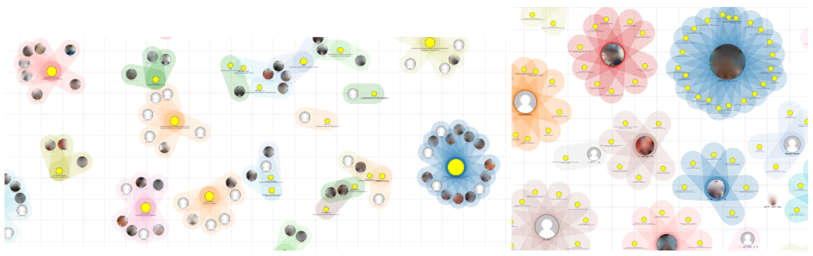

(a) Service-centric overview **(b)** Person-centric overview

Figure 6.5 – Overviews in the service cartography

- Aggregate overview of relationships among all services with their context (not shown).
- The cropped overview in Fig. 6.5a shows collaboration for a specific service, where people are grouped around a service on which they worked at least once.
- The cropped overview in Fig. 6.5b shows the opposite view of Fig. 6.5a: the employee is visualized in the center of all services on which he worked at least once.

Architect delegates need and feedback

In this iteration, the development of the service cartography with all new features was done by a master's student, the third co-author of [Tapandjieva et al., 2017a]. In addition, we tried to initiate a closer collaboration with the IT department members who have an architect delegate role. An architect delegate is a specialist in one architectural domain, such as network, security and databases. The architect delegates and the head of IS architecture form the architecture body that ensures the coherence and efficiency of the university's information systems [EPFL IS Architecture]. We attended two monthly architecture meetings and received feedback, ideas and requests for new features. Due to time constraints for finishing this thesis we did not have the resources to develop a more intensive individual collaboration with each architect delegate, in order to elicit all service cartography use cases and implement them. Nevertheless, from the feedback we obtained, we are able to articulate heuristics for building a service cartography tool in the context of an IT department (see Section 6.5).

Reflection and Learning

In the previous two iterations, we focused on the design rather than on the action in the organization. By including the architect delegates in the early stages of the service cartography redesign, we hoped to avoid some of the mistakes presented in [Markus and Keil, 1994], such as building a system that employees are reluctant to use. But the lack of resources for continuing the project prevented us from further iterations.

Internal service exchange

After all project iterations, we stabilized the problem formulation which emphasizes the visualization and communication of the internal service and value exchange. Even Vargo and Lusch [2004] indicate the importance of the internal service exchange in their second foundational premise. They explain that without direct interaction with the end customer, employees lose the sense of the internal service exchange among themselves, which leads to neglecting “quality and both internal and external customers” [Vargo and Lusch, 2004]. The role of the service cartography is to provide insights of all services, being internal, external, IT or business.

6.4.4 Limitations

Three issues

In terms of building geographic visualizations, Dodge et al. [2008] point to three particular issues that are worth discussing: practical limitations, ethical concerns and political interests. Although in another context, these issues also appeared in our service cartography. We dedicated most of our time dealing with the practical limitations, such as no context in network visualizations and a 24h-long generation of a modified map. The ethical concerns and political interests surfaced in relation with our dual role as both practitioners and researchers. As with geographic maps, a map of the services in an organization reflect politics, such as the power relations between people and departments. A political viewpoint is inevitable in something meant to be shared with many people. As with any action research project, it is important for the practitioners participating in the project to manage politics well, in order to “have a future in the organization when the research is completed” [Coghlan and Brannick, 2014]. In our case, involving the IT department members in this project required us to manage relationships and to negotiate with the employees and their supervisors. Sometimes we encountered delays in securing the involvement of practitioners, thus making the duration of the ADR project unpredictable.

Evaluating claims

The emergent knowledge from an experience is “a knowledge which is contingent on the particular situation” [Susman and Evered, 1978] of a given moment (in this ADR project). All decisions taken “are subject to reexamination and reformulation upon entering every new research situation” [Susman and Evered, 1978]. We cannot provide measurements for the improvements our service cartography brought to the IT department, and this chapter only presents observations and opinions from a few practitioners.

6.5 A Heuristic for Building a Service Cartography Tool

In this section, we present the *formalization of learning* stage in this ADR project in terms of design principles that emerged during the stages of *reflection of learning*. In the context of this thesis, to align the formalization of learning with the previous two chapters, we call these principles heuristics.

6.5.1 Heuristic 5: Facilitate self-recognition while communicating different service perspectives

Epistemological issues

Initially, we were to build the same tool for all IT department employees. But, this posed a problem, as people holding different roles needed and created different kinds of knowledge around a service. Some needed a detailed view, others an aggregate overview. More concretely,

- A service manager is concerned with the description of his service implementation, where his service is used (impact), the ongoing incidents, etc. We elicited these needs with the help of the one-month contextual inquiry we conducted with a service manager during the second iteration.
- A segment manager cares about all services in his segment, knowing the service managers of these services, and what kind of relationships exist among services, recognizing the specific needs of users in his segment and acting accordingly, etc.
- An architect delegate cares about the live service data that is obtained from the infrastructure, the impact of one service and technology on the other services and users, optimizing the usage of technical resources (creating architectural packages) in services, etc. Some of these requirements came from the architect delegate we collaborated with, and others were pointed out via the feedback from the architect delegates.
- A top manager needs the information concerning the cost of a service, a dashboard summarizing all services and issues, people with similar knowledge of a service, etc. We discovered these requirements during a meeting with the IS coordinator.

Despite the fact that everyone should have a shared vision of the service management and see collaboration around a specific service, a one-size-fits-all tool was not feasible. Instead, the service cartography tool needs to provide predefined views of the service maps tailored to the roles of the user.

Build personal maps

In addition, people in all roles need to see *their* map of services, information they need, and of services they manage or depend on. Rodighiero and Cellard [2016] recognize and discuss the impact of self-recognition in social visualizations. One example is when people try to find their current location while looking at a geographic map. To accommodate this need, maps on specific locations contain annotations such as 'YOU ARE HERE'. Although the service cartography tool is another type of visualization, finding oneself and self-recognition in this tool is crucial for facilitating the process of building a shared understanding, collaborating between people, as well as committing for changes and improvements in the organization. During our collaboration with the practitioners, we inferred the need for dynamically changing service maps, allowing users to build a specific representation of the information that fits *their* (self-recognition) needs. And this information often covers several service levels. For example, during the contextual inquiry, we participated in the knowledge base write-up for a service designated for the people providing support. This entry described not only the problems that can be encountered in the implementation, but also problems that can arise from other connected services.

Self-recognition: Enable users to update existing service maps or to interactively build service maps of several service levels in which they identify themselves.

Additional Learning

Navigation

With the transformation towards service orientation, various frameworks and tools were introduced. Some of them included ServiceNow [ServiceNow], Confluence [Confluence] and SharePoint [SharePoint]. We observed that this created confusion among the IT department members, as the relation between them was unclear. In the service cartography, all dimensions around services should be captured. This requires a connection with and navigation to the other existing tools and existing data sources, as they contain a level of detail that does not belong to the cartography, but might be relevant for the cartography user. An advanced feature that results from such connections would be fetching 'live' data from the infrastructure; this data is aggregated and then visualized in a service map. Our cartography tool has to yet implement such features. Another requirement that we noticed, but still have not completely met within the cartography tool, is the intuitive navigation between created service maps and predefined overviews. Navigation within maps follows the second heuristic, self-recognition, in which personal maps of one user relate to personal maps of another user or to predefined overviews.

Context

No service implementation or usage exists in a vacuum, so it is imperative that the cartography tool shows the context for every service. SEAM models always show the context: either the system as a composite for service implementations, or system as a whole for services provided.

Notation

During the BIE iterations, we applied different notations for the visualizations. Some of them were simple boxes in boxes, whereas others represented a simple network. A network visualization was the most appealing, as it enables the impact analysis and other features. Some of our attempts for network visualizations are described and depicted in Appendix A.3.1, but they are static and context-free. From the beginning of our project, we used SEAM to conceptualize our work and in the communication with the IT department members. The people we were collaborating with were often exposed to the SEAM notation and they adopted it. This motivated us to base the service cartography tool on SEAM models, where people and their services can be modeled at any level of the hierarchy and seen in a concrete context. We struggled with the implementation of SEAM notation in our initially chosen tool SOLU-QIQ and, due to other difficulties with SOLU-QIQ (around 24h-long generation of maps), we decided to build a tool from scratch. We had already noticed the pitfalls of SEAM in representing multiple service levels at the same time. For example, there is no ideal way to represent the boundaries for multiple systems as a composite, or to represent multiple levels that use multiple common services. We modified the notation accordingly and removed the notion of a system as a composite; we kept only the context of a service within its system as a whole.

Meta-model

Our second iteration was about modifying/customizing the underlying meta-model of the SOLU-QIQ tool while building different prototypes and applying various approaches. Through this experience, we realized the importance of flexibility in terms of a meta-model. As SOLU-QIQ was not flexible in terms of the generated output (the static web site), the workflow for inserting data and building models, we initiated the third iteration to develop a new tool from scratch. The development of the new tool did not reach the maturity to enable such customization, but we plan to include it in future iterations.

6.6 Future Work

The project ended at the beginning of 2017 due to lack of resources for developing more features and due to the time limitations for completing this thesis. These are a few ideas for future improvement of the cartography tool:

- Login and authentication, as not all perspectives should be visible to everyone.
- Integration with an existing knowledge base and configuration repositories, such as Confluence and ServiceNow.
- Improve the notation to show composite systems. Currently such a distinction is made with the direction of the links between services.
- Understand more organizational roles to provide predefined overviews.
- Clarify how people agree on a common perspective. This could imply using version and revision control for all future modifications after an agreement.

Some of these requirements surfaced at the meetings with the architect delegates. We have documented in more detail the situation, explaining the service cartography tool in a requirements document.

6.7 Chapter 6 Conclusion

In this chapter, we present an ADR project of building a service cartography artifact, the directions taken, and the difficulties encountered. Our envisioned use of the service cartography is in the context of service management. In all iterations of concurrent building, intervention and evaluation, we collaborated with at least one practitioner. Through this collaboration, our involvement in other IT department projects and our observation of the work of other practitioners, we directly experienced the anticipated features for the service cartography tool that helped to refine the problem we were addressing. We were also able to elicit heuristics and give helpful suggestions that can be applied in the development of tools that support the service management in an organization. Some of these heuristics emerged from our involvement in the daily work of practitioners, as we could not predict all requirements for a service cartography. One such heuristic is the user self-recognition (allowing people to build their own maps) with which we transfer the control of describing the organization to all users, not only people with specific roles like architects or top managers. Building specific perspectives of the interconnected services and service instances is another implication of the self-recognition heuristic. The cartography tool helps to navigate and relate people's perspectives, which in turn is a means of achieving alignment.

7 Thesis Conclusion and Future Work

Recursive, interdependent and abstract nature of services

The entities that exchange services are known as service systems, and they are defined as "a value-coproduction configuration of people, technology, other internal and external *service systems*, and shared information (such as language, processes, metrics, prices, policies, and laws)" [Spohrer et al., 2007] (italics added). This definition reveals the recursive and interdependent nature of service systems. At the same time, service systems are also an abstraction of the application of resources for the benefit of another [Maglio et al., 2009]. These characteristics make the conceptualization of services difficult.

Thesis – part of an action research project

In this thesis, we present our research results on overcoming the difficulty of conceptualizing services. The basis for our research was the challenges the IT department of our university faced in its adoption of services across the whole university organization. Our research laboratory had begun a collaboration (an action research project) with the IT department two years before the beginning of this research. We present only part of this collaboration, organized in three interrelated projects. In all the projects, we used the SEAM method [Tapandjieva et al., 2014; Wegmann, 2003] for the development of our results: three artifacts and five heuristics. These results are also our contributions and they facilitate the conceptualization of services spanning across organizational boundaries. We present them in Table 7.1.

SEAM service-modeling ontology

The first artifact is the SEAM service-modeling ontology. It provides a simple way to model and design service systems. The combination of the ontology with our *Heuristic 1* offers a new perspective on systems thinking. Let's consider that we perceive a system that implements a service. If some of its composing systems do not contribute in the implementation, they either should not be part of the main system

Table 7.1 – Summary of the resulting artifacts and heuristics. They are listed as they appeared in the chapter that describe the project.

Chapter 4	SEAM ontology for service modeling **Heuristic 1:** The behavior defines the service system's structure
Chapter 5	Aligned SEAM service models **Heuristic 2:** Immediate users and their users **Heuristic 3:** Different service instances provided by one service implementation **Heuristic 4:** Concrete (and formal) service alignment
Chapter 6	Service cartography **Heuristic 5**: Different user's epistemology/perspectives **Heuristic 6:** Self-recognition

structure, or we did not perceive the system correctly, or we need to change something in the system. Such reflection goes beyond the legal boundaries of organizations, as we care only about what the system does, and not where it officially belongs. To the best of our knowledge, this is the first systemic approach that puts emphasis on the behavior.

Aligned SEAM service models

The second artifact provides a way to define and model aligned service systems, at any level of the organization. These models operationalize the SEAM service-modeling ontology. The three heuristics we propose give guidelines on how to approach the subject of alignment. By adopting the social dimension of alignment [Reich and Benbasat, 2000], our *Heuristics 2* and *3* acknowledge the different contexts of service use. Ideally, each user in a certain context would like a service tailored to his needs. Taking it to the extreme, there should exist as many service instances as contexts. Our heuristics encourage the reflection on these possible instances and contexts. We redefine the role of a service manager to include the reconciliation of the service instances with one implementation. Our final contribution in this project comes from *Heuristic 4* and it is our formal definition of the alignment concept, based on properties of the system behavior. Using this formal definition, we were able to develop a tool for automatic alignment verification of service design models.

Visualization of service systems in a cartography

In the third project, we developed a tool to visualize, manage, and to share various perspectives on services. The heuristics we derive follow the findings from the other two projects. The self-recognition means building maps in which the person finds the services of his interest, and understanding the different perspectives helps to offer predefined views.

Relevance

The relevance of our projects is in their intimate relation with the real challenges faced by IT departments when adopting services. The findings in the form of artifacts and heuristics solve one challenge: they enable the conceptualization of services across the IT department and beyond. Unlike other service approaches that adopt only one perspective, such as business, IT, marketing, management, customer and provider perspectives, the simple modeling ontology that we present is adaptable to any perspective as long as the heuristics we propose are followed. These heuristics help to build models with constant reflection on the reality: Who are the users of the users? What are the service instances? What is the actual behavior of a component? Does that component need to be added in the service implementation? Which context yields a new service instance that needs to be managed (time, location, duration, user, etc.)?; and many more questions. The heuristics might seem rather generic, and we made our best attempt to demonstrate them with examples and the real services that we studied during the collaboration with the IT department.

Applicability of contributions

Despite being situated in the context of an IT department, we show that our approach to service conceptualization is applicable across domains and disciplines. We have provided means to reflect on what constitutes a service system, how this system is aligned with others across the organizations, what service instances are there, how to manage, visualize and share perspectives on business and IT services.

Future work

Many questions remain open. When we consider the artifacts and heuristics as an ensemble, some research opportunities arise:

- What should we include in the extended ontology, so that it best supports the alignment? More reflection and research is needed to find what the important service aspects that might yield a new instance are. Also, what are the instances that impact the alignment?
- Who cares the most about alignment? Are there any means for quantifying it and using this information in a service cartography? What does a flexible structure (defined by the behavior) mean for such measurement?
- The scope of the action research project, which we did not cover, included governance and human resources. There exist many approaches and tools that assist these two functions, such as COBIT and the SAP tool. How can a service approach, which offers formal alignment across uncountable perspectives and a supporting cartographic service tool, help in or relate to governance and HR?

A Appendices to Thesis Chapters

A.1 Complement to Chapter 2

A.1.1 System Definitions: Differences

We illustrate the different positions and views of systems thinking by presenting several system definitions from different scholars.

- Ackoff and Gharajedaghi [1996, p. 13] give the following definition: "A system is a whole defined by one or more functions, which consists of two or more essential parts. (1) Each of these parts can affect the behavior or properties of the whole. (2) None of these parts has an independent effect on the whole; the effect an essential part has on the whole depends on what other parts are doing. (3) Every possible subset of the essential parts can affect the behavior or properties of the whole but none can do so independently of the others. Therefore, a system is a functioning whole that cannot be divided into independent parts."
- Weinberg [2001, p. 62] says that "A system, any system is the point of view of one or several observers."
- Meadows [2008, p. 2] defines a system as "a set of things – people, cells, molecules, or whatever – interconnected in such a way that they produce their own pattern of behavior over time."
- von Bertalanffy [1968, p. 55] simply states: "A system can be defined as a set of elements standing in interrelations."
- Klir [2001, p. 462] points out: "In general, a system is an abstraction distinguished on an object by an observer, which reflects the interaction between the observer and the object."

A.1.2 Two Systems Theories That Use Analogies

Living Systems Theory

In the Living Systems Theory (LST), Miller [1978] integrates biological and social science to provide a set of principles valid for all *living systems*. One of the principles is describing the universe as "a hierarchy of systems, each more advanced or 'higher' level made of systems of the lower levels" [Miller, 1978]. Miller identified eight hierarchical levels: cell, organ, organism, group, organization, community, society, and supranational system. At each level, there are twenty critical subsystems, eighteen that process either matter–energy or information, and two, that process both matter–energy and information. LST uses concepts of space and time, matter and energy, information theory, cybernetics, as well as other concepts applicable to each level. Miller's aim is to describe living systems in terms of flows (of information, energy, matter) which will clarify and unify the facts and nature of life. The general model of a living system is described in an observer's language.

Viable System Model, recursiveness and homeostasis

Another theory is Beer's Viable System Model (VSM) [Beer, 1984], where the analogy is the *human nervous system*. Beer claims that the principles of sensing the environment, regulation, adaptation, coordination, control and development are applicable to all kinds of organizations. In the VSM there are five necessary and sufficient interacting subsystems that undertake different responsibilities

- System 1: Performing primary activities (operations).
- System 2: Performing coordination and regulation for the primary activities.
- System 3: Guiding System 1 and supervising System 2.
- System 4: Interacting and getting feedback from the environment while interacting with System 3.
- System 5: Deciding and steering the whole system by interacting and balancing the activities of System 3 and System 4.

A fundamental phenomenon in the VSM is the recursion principle that "any viable system contains, and is contained in, a viable system" [Beer, 1984, p. 8]. Consequently, to study any of the five subsystems implies that another five subsystems are encountered. This means that hierarchies are also present in the VSM, and these hierarchies look the same. As with other systems theories, in a VSM, the goal of all systems and their interaction is the regulation of homeostasis.

A.2 Complement to Chapter 4

A.2.1 Extended SEAM Meta-Model

This meta-model depicted in Figure A.1 includes the properties of a service and the invariants of a process. Appendix B builds on this meta-model for the automatic alignment verification of SEAM service models.

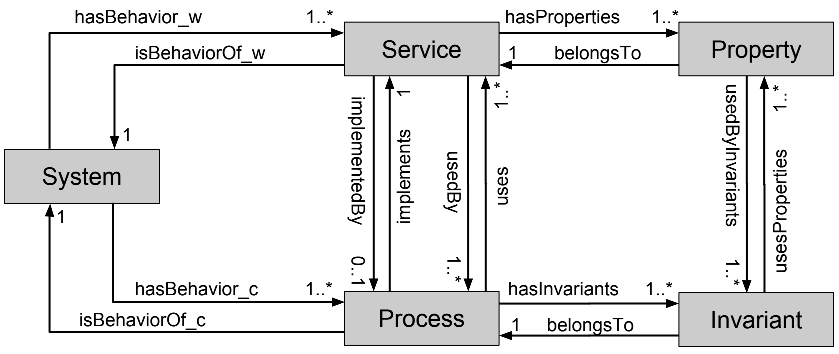


Figure A.1 – Extended meta-model of SEAM service modeling concepts.

A.3 Complement to Chapter 6

A.3.1 Service Visualization Prototypes

The following figures represent our attempt to use information visualization techniques for representing service models during the second BIE iteration. The data source for these visualizations was the IT department's service catalog. In addition, we added the dependencies among services in the way we perceived them. We found our prototypes appealing, but all of them lacked an important feature: representation of the service context, i.e., the service system responsible for the offering. The value we got from making these prototypes was the experience from working with web technologies that provided an excellent usability. This experience helped us to understand and choose the technologies we used in the third iteration.

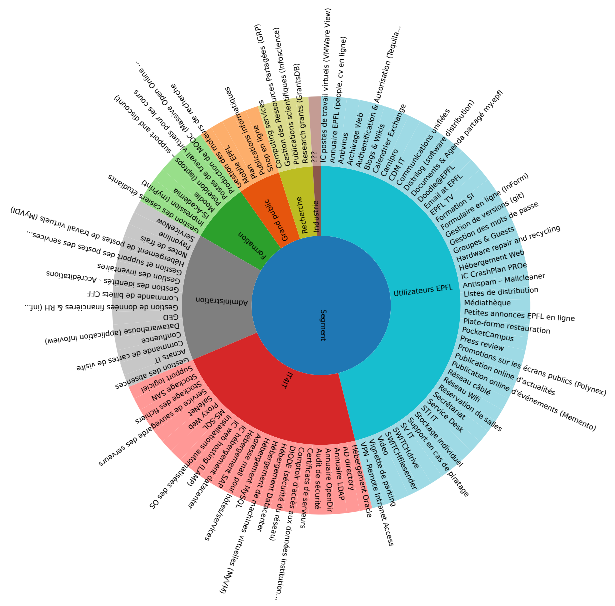


Figure A.2 – Visualization prototype: Zoomable sunburst diagram. Adapted from original visualization by https://bl.ocks.org/mbostock/4348373

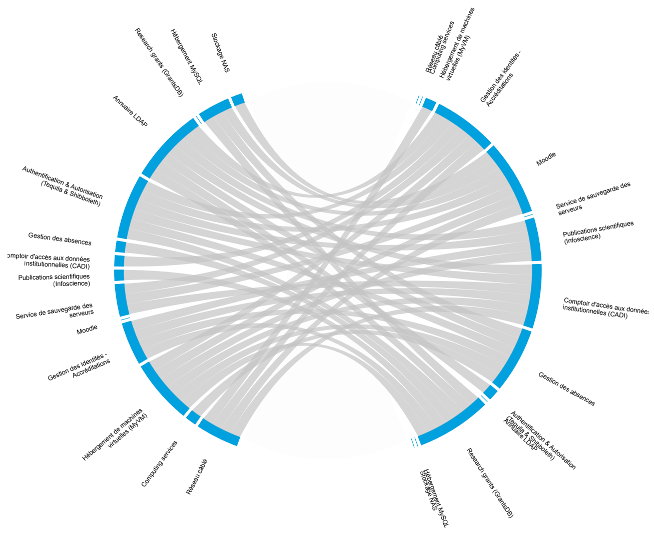
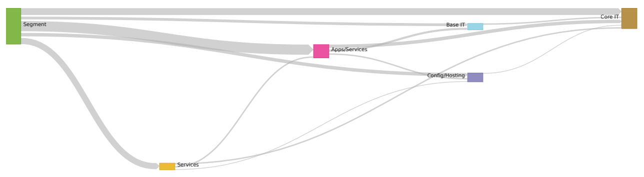
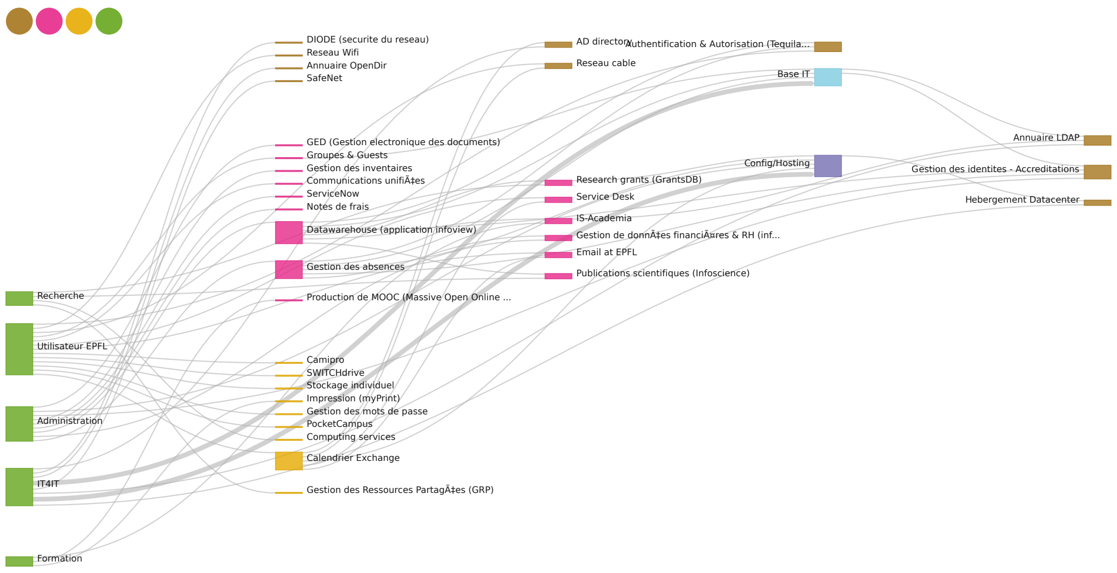


Figure A.3 – Visualization prototype: Chord diagram. Adapted from original visualization by https://www.visualcinnamon.com/2015/08/stretched-chord.html



(a) Bi-sankey aggregated



(b) Bi-sankey expanded

Figure A.4 – Visualization prototype: Bi-sankey diagram. Original visualization from http://bl.ocks.org/Neilos/584b9a5d44d5fe00f779 (accessed December 2017)

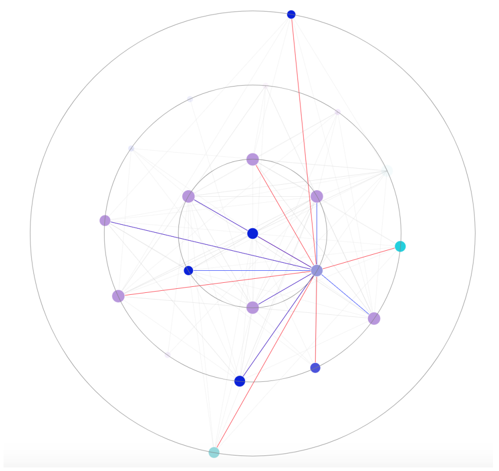


Figure A.5 – Visualization prototype: Services in concentric circles graph. Student project visualization, organizing services in concentric circles based on a computed distance according to the dependencies.

A.3.2 Example of a Service Cartography Output (SOLU-QIQ) in the Second Iteration

The second iteration of this ADR project was marked by the configuration of the SOLU-QIQ around the SEAM method. The first step was the modification of the standard URBA meta-model present in SOLU-QIQ, with the simplified SEAM meta-model reported in Chapter 4 and in the publication by Tapandjieva and Wegmann [2014]. The meta-model has evolved since then, to meet the need for storing additional information around services, but it still conforms and can be mapped to the simplified one.

In this Appendix, we depict the different outputs of the SOLU-QIQ generated service models, one service implementation per view. Due to SOLU-QIQ's difficulties to generate automatically more complicated layouts in nested elements, we had to make the following adjustments:

- Depict systems with rectangles. All the systems have a rectangular shape, with a small icon on the upper right corner that distinguishes the type of system.
  - Gray rectangle is used only for a system as a whole (black box).
  - White rectangle with a black borderline is used only for system as a composite (white box).
- Depict processes with blue rectangles. Like the systems, the octagonal shape icon is used in the upper right corner of the process.
- Depict services with red ovals.
- The nesting of elements in SOLU-QIQ is: system as a composite → process → system as a whole → service, whereas in SEAM it is system as a composite → system as a whole → service, with a system as a composite → process that connects the services.
- In standard SEAM service models all of the levels are shown in one model. In Solu-QIQ only one level for only one process is shown, and there is navigation among the models. If there is information about the implementation of a service, a small **impl** link is put in service oval. This links goes to the next level.

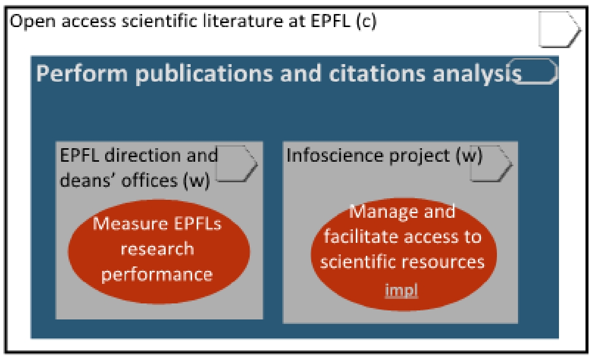

Open access scientific literature at EPFL (c)
Perform publications and citations analysis
EPFL direction and deans' offices (w)
Measure EPFLs research performance
Infoscience project (w)
Manage and facilitate access to scientific resources
impl

Figure A.6 – SOLU-QIQ output of one organizational level. This is the same process as the one depicted in Figure 4.2 with the standard SEAM notation.

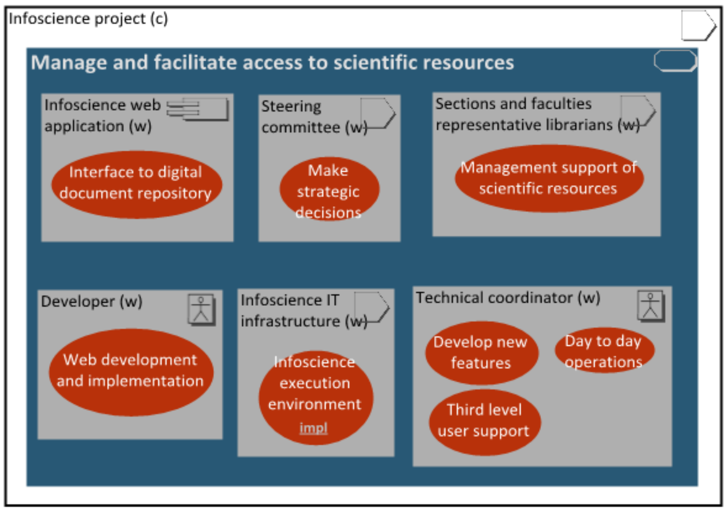

Infoscience project (c)
Manage and facilitate access to scientific resources
Infoscience web application (w)
Interface to digital document repository
Steering committee (w)
Make strategic decisions
Sections and faculties representative librarians (w)
Management support of scientific resources
Developer (w)
Web development and implementation
Infoscience IT infrastructure (w)
Infoscience execution environment
impl
Technical coordinator (w)
Develop new features
Day to day operations
Third level user support

Figure A.7 – SOLU-QIQ output of a lower organizational level. This organizational level depicts the *Infoscience project* service system seen as a composite, the same as in Figure 4.3

B Service Alignment Verification

The authors in [Spohrer et al., 2007] define service systems as "a value-coproduction configuration of people, technology, other internal and external service systems, and shared information". The Information Technology Infrastructure Library (ITIL) [Cartlidge et al., 2012] has a stage that focuses on specifying the service design. In this stage, service designers identify and document requirements [Hunnebeck, 2011] while taking into account the constraints of all components involved in the value co-production. Then, they add the identified requirements and constraints in a service model that is seen as the service design specification. As indicated by [Alloush et al., 2013], an early verification of the service design specification is needed to avoid costly consequences in the service lifecycle.

One aim in the service design stage is to check whether the models are under-specified and allow for scenarios that violate the specification. In the existing literature, such check is done by (1) transforming the modeled specification into a formal language [Ghani, 2014, pp. 80], and then (2) using a model checker or an automated theorem prover to check whether the formal representation satisfies specified properties [Morimoto, 2008].

In industry modeling projects, modelers find verification a difficult task, or even a task that checks the modelers' job performance [Carson, 2002]. The application of formal analysis brings great benefit in eliminating errors at the early stage of development [Sobel and Clarkson, 2002], but not many people have strong formal background, or want to invest

This appendix presents the work described in a paper published at the 2017 10th IEEE International Conference on Service-Oriented Computing and Applications [Tapandjieva et al., 2017b].

much time in verification.

To reduce the effort for formal analysis of service design specifications, we present an automatic verification of requirements and constraints specified in the SEAM modeling language [Wegmann, 2003] developed in our research laboratory. Our approach has the advantage of being independent of formal specification languages and is useful for service designers who do not have substantial knowledge of verification methods. Knowing only the SEAM modeling language and basic Scala programming expressions is sufficient.

SEAM started as an application of the systemic paradigm in the field of enterprise architecture [Wegmann, 2003], with the goal to seamlessly integrate the 'business' with IT. Since its inception, SEAM has expanded to incorporate tools and methods for service modeling, strategic thinking, business-IT alignment, and requirements engineering [LAMS, n.d.a]. Our laboratory and our industrial partners mainly use SEAM in teaching and consulting.

Our model-driven approach has resulted from searching ways to explain alignment to students. We start by specifying the service design requirements and constraints, which are conceptualized as properties. In SEAM, a service is modeled at two different levels of abstraction: service offering and service implementation. The service offering level describes the stakeholders' relationship with the service provider. The service implementation level shows the relationships among actors, components and resources used to deliver the service. In the first level, we specify the properties that describe the requirements. In the second level, we specify the refinement of the first level properties as a combination of the properties (constraints or requirements) set by the actors, components and resources involved in the service delivery. Consequently, we define the verification of the service design specification model as the alignment verification of the model's properties within one level and among levels.

We have developed a tool for automatic verification of these service design specification models. The tool first translates the properties of the SEAM model into functional Scala code. The resulting Scala code is then passed to and checked with a verification system called Leon. Leon either:

- confirms that the specification is correct with respect to the specified properties, or
- provides a counterexample with an erroneous property value.

This automatic translation relies on globally unique identifiers that are used to preserve the structure of the model. If a counterexample is found, the tool uses these identifiers to annotate the model and mark the faulty property specification.

Our approach focuses on properties that can be quantified, such as performance, latency, storage size, budget and maintenance time. This work extends the refinement and verification of behavior properties, expressed in terms of pre- and post-conditions on actions [Rychkova et al., 2008]. A description of this extension is in [Tapandjieva and Wegmann, 2015] and in this paper we present advancements made in:

- automating the translation of the SEAM service design models to Scala verifiable code, and
- showing the result back in the SEAM model.

The remainder of this paper is organized as follows. We first outline the related work. Then we describe the alignment verification on an example and we give details of the implementation of our automatic alignment verification with SEAM, Scala and Leon. This is followed by the limitations we face. Before concluding, we present our envisioned future work.

B.1 Related Work

To the best of our knowledge, our approach differs from the existing literature in verification either in the target domain (service design in our case) or in the support for visually displaying the verification result.

In the context of software engineering, the adoption of Model-Driven Development (MDD) includes the use of tools to check the model correctness. There exist tools for the analysis of Unified Modeling Language (UML) diagrams annotated with Object Constraint Language (OCL) constraints [Cabot et al., 2007; Richters and Gogolla, 2000]. These approaches do not provide visual analysis result.

In the context of business process modeling (BPM), in [Morimoto, 2008] the author presents a survey of different verification approaches. A tool for automatic verification of BPMN choreographies is presented in [Solaiman et al., 2015]. Authors do not show the BPMN diagram in the tool, and unlike our approach, the verification output is displayed in the console. In addition, we find the design and the verification of BPMN choreographies come late in or after the service design stage.

In the context of early requirements,in [Fuxman et al., 2004] the authors present a framework and a tool for formal verification of early requirements specifications. Their framework combines early requirements engineering (i* models) with formal methods (model checking). As i* is static, the user should write formal specifications in the Formal Tropos language to describe the temporal dynamics of the model. Besides learning a formal language, the user does not get a visual feedback from the verification.

There exist many modeling approaches, such as the ones presented in [Engelsman et al., 2011], [Plataniotis et al., 2015] and [Ramesh and Jarke, 2001], that emphasize the importance of the alignment and traceability between the requirements and the design decisions. The impediment of these approaches is the lack of formal verification of the design decisions. Others approaches, such as [Hallerstede et al., 2014] and [Cabot et al., 2007], require users to learn additional formal languages, Event-B or the OCL. On the other hand, our approach offers a new perspective for visually doing alignment verification directly on the model, with only writing basic Scala expressions.

As other approaches, the previous work of Rychkova, [Rychkova et al., 2008] and [Rychkova, 2008], relies on manual mapping of SEAM constructs to Alloy[1] verifiable code, and requires service designers to know Alloy and interpret the result from the Alloy Analyzer tool. In contrast, our work relies on writing basic Scala arithmetic and logic operations in the SEAM model, but does not capture behavioral properties.

B.2 Modeling Example: Manage Gas Leak

We illustrate our approach and we informally explain the semantics of our modeling language on a fictive example inspired by a real project conducted in a utility company. The utility company (UC) manages water, gas and electricity distribution.

In the example, we specify the design for a security service provided by UC for managing reported gas leaks. A regulation body sets the safety standards UC must respect concerning the time for securing a site where a gas leak is reported. The regulation is expressed as a service description: *The UC must neutralize a gas leak reported by a witness, guaranteeing that a specialized UC team or a fire brigade arrives within*

[1]Alloy [Alloy, n.d.; Jackson, 2002] is a language used to describe basic structures, as well as constraints and operations describing how structures change. It comes with an analyzer tool: a solver that graphically displays the structures modeled.

20 minutes and that the incident site is secured within 45 minutes from the time of the registration of the witness' call.

B.2.1 SEAM Service Model Showing the Specification for Managing Gas Leaks

Fig. B.1a depicts the SEAM model of the service offering specification level. We named this level Gas community and we specify three stakeholders with services they provide:

1. the Utility company offering the Manage gas leak service,
2. the Witness with the Report leak action (service), and
3. the Gas safety regulation body providing the Regulate safety of UC services.

This SEAM service model allows designers to conceptualize the utility company as a hierarchy of systems that provide services[2]. In this hierarchy, systems are conceptualized either as *wholes*, denoted with *[w]* (black boxes), or as *composites*, denoted with *[c]* (white boxes).

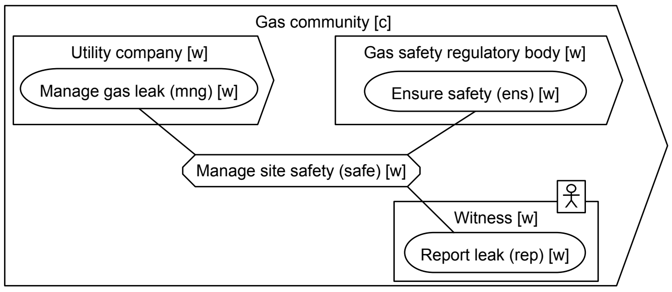
In a system as a whole, the system's components are ignored and the focus is on the *services* offered by the system to its environment. A system as a composite shows the context and contains multiple systems as wholes whose services interact through a *process*. A process in a system as a composite gives the implementation of the corresponding service in the same system as a whole. In SEAM, processes allow for service collaboration and service exchange among systems. This is a direct application of the first foundational premise from service-dominant logic: "Service is the fundamental basis of exchange" [Vargo and Lusch, 2008].

In our example, all actors mentioned interact in the Manage site safety process and they belong to the *Gas community [c]* system (see Fig. B.1a). The *Gas community [c]* composing systems as wholes are visible: the *Utility company [w]*, the *Witness [w]* and the *Gas regulation body [w]*.

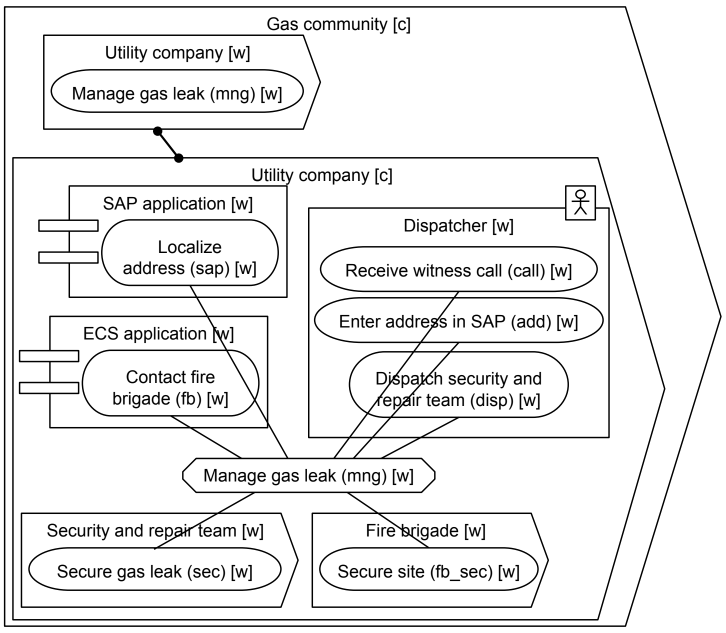
The main service of interest is the *Manage gas leak*, and Fig. B.1a represents the specification of the service offering level. Fig. B.1b illustrates how the UC is organized in providing the *Manage gas leak* service.

The composite view of the *Utility company [c]* depicted in Fig. B.1b reveals the actors involved in the realization of the *Manage gas leak*

[2]We use system to refer to an observed entity: an organization, a customer, an employee, an IT system, or an application [Wegmann et al., 2008]



(a) Service offering level


Gas community [c]
Utility company [w]
Manage gas leak (mng) [w]
Utility company [c]
SAP application [w]
Localize address (sap) [w]
Dispatcher [w]
Receive witness call (call) [w]
Enter address in SAP (add) [w]
Dispatch security and repair team (disp) [w]
ECS application [w]
Contact fire brigade (fb) [w]
Manage gas leak (mng) [w]
Security and repair team [w]
Secure gas leak (sec) [w]
Fire brigade [w]
Secure site (fb_sec) [w]

(b) Service implementation level

Figure B.1 – SEAM service-design specification models.

service:

- the *SAP application* provides location coordinates for a given address,
- the *ECS application* monitors the actions of all actors within the UC and automatically contacts the fire brigade in certain conditions,
- the *Dispatcher* receives the witness call, enters the address of the reported leak in the SAP application, and dispatches the security and repair team to the location of the leak,
- the *Security and repair team* secures the gas leak, and
- the *Fire brigade* secures a site.

The *Manage gas leak* in the context of the *Utility company [c]* is a process that implements the corresponding service of the *Utility company [w]*, therefore we call this a specification of the service implementation level.

B.2.2 Quantitative Properties in a SEAM Service Design Specification Model

The model from Fig. B.1a and Fig. B.1b is extended with properties (specifications of the requirements and constraints of systems' actions). In our example, the Witness and the *Gas safety regulation body* are not concerned with the UC's internal operation and agreement, they only know:

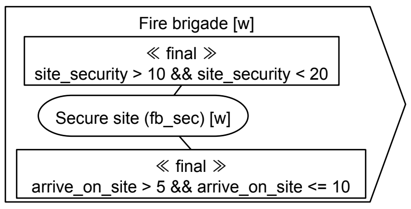
- the time it takes for a specialist to *arrive on site*, and
- the time it takes to *secure* the site or the leak.

We need to refine these two times according to the values of UC actor's properties. Let's consider that the UC has the following agreement among its internal systems:

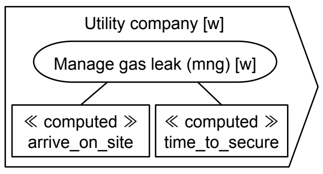
- The *SAP application* takes *one minute* to provide location coordinates for a given address.
- The *Security and repair team*, after receiving the call, needs (1) between *10 and 30 minutes* to arrive on the site location, and (2) between *10 and 15 minutes* to secure the gas leak.
- If the *Security and repair team* does not arrive on the site location after *15 minutes*, the *ECS application* automatically contacts the *Fire brigade*.
- After receiving a call, the *Fire brigade* needs between *5 and 10 minutes* to arrive on the site location, and firemen need between *10 and 20 minutes* to secure a certain perimeter of the site.

It can be either the *Security and repair team* to arrive on the site, or the *Fire brigade*. The *Security and repair team* is dispatched by default after the first minute of receiving the call (the time it takes for *SAP* to retrieve the location coordinates of the leak). If the team does not arrive in the following 14 minutes to the site, the *ECS application* automatically signals the *Fire brigade*.

We already introduced an extension to the modeling language with a specification for four kinds of quantitative properties [Tapandjieva and Wegmann, 2015]. These properties have a stereotype and can only be connected to a service or a process.


Fire brigade [w]
≪ final ≫
site_security > 10 && site_security < 20
Secure site (fb_sec) [w]
≪ final ≫
arrive_on_site > 5 && arrive_on_site <= 10

(a) Final properties for the *Fire brigade [w]* system



(b) Computed properties in the *Utility company [w]* system

Figure B.2 – SEAM service properties types.

Service Property Types

A ≪*final*≫ property specifies service requirements or constraints independent of other entities in the model. Properties defined in an Operational Level Agreement or an Underpinning Contract [Hunnebeck, 2011] are modeled with ≪*final*≫ properties. Fig. B.2a depicts the final properties of the *Fire brigade [w]*. These properties are usually written as a Boolean expression over strictly one variable that gives a range, or even one value.

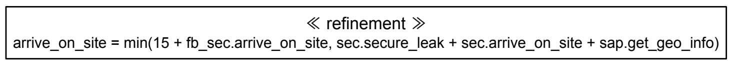
A ≪*computed*≫ property shows there is a corresponding service implementation process that specifies the computation. In the current context, the system as a whole, we do not know details about this computation. Fig. B.2b depicts the computed properties of the *Utility company [w]*.

Process Property Types

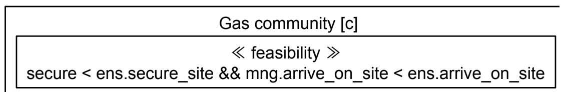
In our modeling language, the process shows a refinement of (1) a service offering to a service implementation, and (2) a system as a whole to a system as a composite. It connects and orchestrates services, so properties connected to a process expose the logic of the model specification.

A ≪*refinement_relation*≫ property contains an expression that uses the values from the final properties of the connected services. The result of this expression is then transferred to the ≪*computed*≫ property of the corresponding service being implemented. Fig. B.3a depicts the *arrive_on_site* refinement relation property in the *UC [c]*, which is transferred to the computed property shown in Fig. B.2b.

A ≪*feasibility*≫ property gives the correctness of the level's specification. The desired service outcome of a property, like a Service Level


≪ refinement ≫
arrive_on_site = min(15 + fb_sec.arrive_on_site, sec.secure_leak + sec.arrive_on_site + sap.get_geo_info)

(a) Refinement relation property in the *Utility company [w]* system


Gas community [c]
≪ feasibility ≫
secure < ens.secure_site && mng.arrive_on_site < ens.arrive_on_site

(b) Feasibility property in the *Gas community [c]* system

Figure B.3 – SEAM process properties types.

Agreement (SLA) used to specify "the metrics the client can use to monitor and verify the contract" (Spohrer, Maglio, Bailey, and Gruhl, 2007) is formalized as a Boolean expression and specified in a ≪*feasibility*≫ property. We use this property to define the alignment of a service as a conjunction of feasibilities: (1) from the level where the service is used in the system as a whole, and (2) from the level where the service is implemented in the system as a composite. Fig. B.3b depicts the feasibility property of the *Gas community [c]*.

Fig. B.4 depicts the complete SEAM model, specifying the service offering and the service implementation levels with all quantitative properties: *arrive_on_site, get_geo_info, secure_leak, secure_site* and *site_security*. The names of these properties and their complete expressions are used in the Scala code, hence the usage of underscores. In addition, we include an identifier in parenthesis next to the names of all services and processes, for example *(ens), (sec), (sap)*, etc., with the aim to uniquely reference properties. This same identifier must be used in the service name and its corresponding process name to capture the references to properties from different services and to transfer values from refinement relationship to the computed properties.

B.3 Verification of SEAM Service Design Specification Models with Scala and Leon

We first describe the concrete and abstract syntax of the SEAM modeling language, and then we give the mapping of the model elements to Scala code, used to:

1. generate a Scala object representing the model,
2. verify the Scala object with the Leon verification system,
3. and generate an output SEAM model to visualize the verification

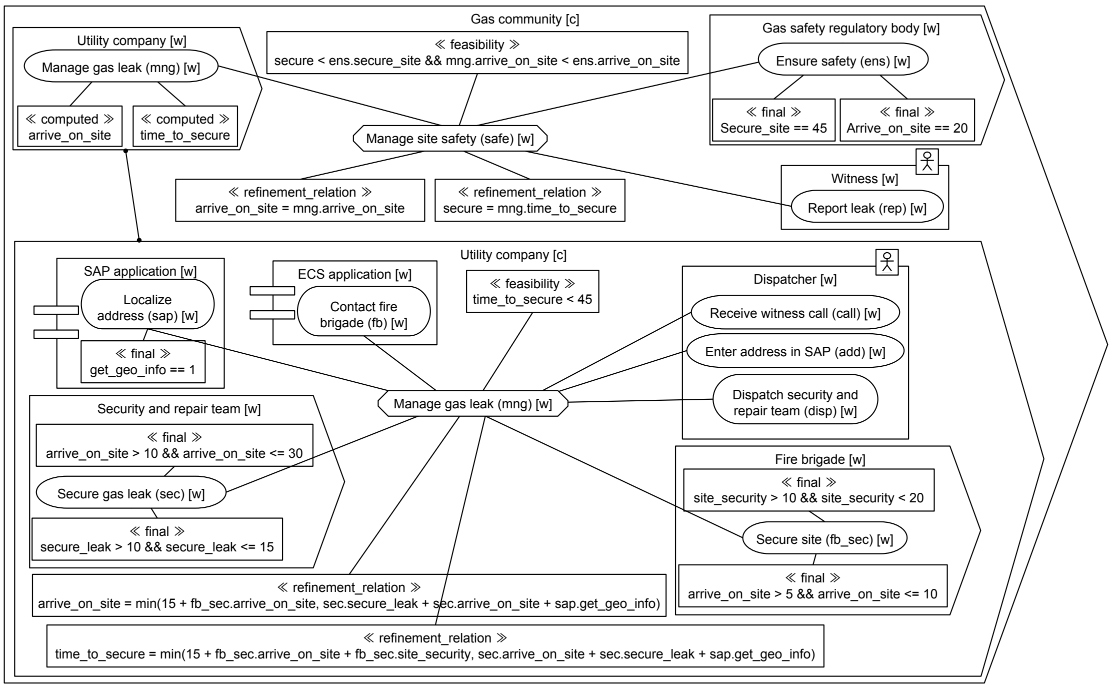

Gas community [c]
Utility company [w]
Manage gas leak (mng) [w]
≪ computed ≫ arrive_on_site
≪ computed ≫ time_to_secure
≪ feasibility ≫ secure < ens.secure_site && mng.arrive_on_site < ens.arrive_on_site
Gas safety regulatory body [w]
Ensure safety (ens) [w]
≪ final ≫ Secure_site == 45
≪ final ≫ Arrive_on_site == 20
Manage site safety (safe) [w]
Witness [w]
Report leak (rep) [w]
≪ refinement_relation ≫ arrive_on_site = mng.arrive_on_site
≪ refinement_relation ≫ secure = mng.time_to_secure
Utility company [c]
SAP application [w]
Localize address (sap) [w]
≪ final ≫ get_geo_info == 1
ECS application [w]
Contact fire brigade (fb) [w]
≪ feasibility ≫ time_to_secure < 45
Dispatcher [w]
Receive witness call (call) [w]
Enter address in SAP (add) [w]
Dispatch security and repair team (disp) [w]
Manage gas leak (mng) [w]
Security and repair team [w]
≪ final ≫ arrive_on_site > 10 && arrive_on_site <= 30
Secure gas leak (sec) [w]
≪ final ≫ secure_leak > 10 && secure_leak <= 15
Fire brigade [w]
≪ final ≫ site_security > 10 && site_security < 20
Secure site (fb_sec) [w]
≪ final ≫ arrive_on_site > 5 && arrive_on_site <= 10
≪ refinement_relation ≫ arrive_on_site = min(15 + fb_sec.arrive_on_site, sec.secure_leak + sec.arrive_on_site + sap.get_geo_info)
≪ refinement_relation ≫ time_to_secure = min(15 + fb_sec.arrive_on_site + fb_sec.site_security, sec.arrive_on_site + sec.secure_leak + sap.get_geo_info)

Figure B.4 – Complete service design model specification. This model shows the service offering and the implementation with quantitative properties.

output.

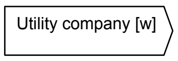
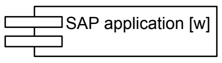
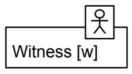
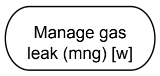
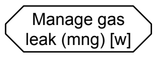
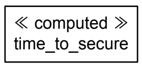
B.3.1 Concrete Syntax

The concrete syntax of a visual modeling language is formed by the visual vocabulary (graphical symbols) and the visual grammar (compositional and well-formedness rules) [Moody, 2009]. The different graphical symbols of SEAM are partly described in [Tapandjieva and Wegmann, 2014] and are listed in Table B.1.

B.3.2 Abstract Syntax

Service designers who create models with our modeling language use the SeamCAD tool that is developed in our research laboratory [LAMS, n.d.b]. SeamCAD uses an abstract syntax as a structure to save the models in an XML file. The abstract syntax is depicted in Fig. B.5. All the systems, actions and properties visual elements inherit from the *Node* class, and the link elements (lines) inherit from the *Edge* class.

Table B.1 – SEAM Modeling language visual vocabulary

Systems	
	A company, a department, an organization
	IT application, program module, IT component, IT platform
	Human actor
Actions	
	An action of a system as a whole
	An action of a system as a composite
Properties	
	A property of a service or a process, depending on the stereotype
Links	
———	A link between service-process, service-property and process-property
•———•	Refinement link, from [w] to [c]

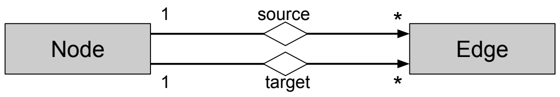


Figure B.5 – The meta-model used in the SeamCAD tool. It is the abstract syntax of our modeling language

B.3.3 Leon and Scala

Scala [Odersky et al., 2008], [Scala] is a functional programming language that implements "means of design-by-contract style specification of pre- and postconditions on functions" [Scala Library]. The `require` clause is used to express a precondition, and `ensuring` a post-condition.

Leon is a system that does software verification, program synthesis and program repair for a subset of the Scala programming language [Blanc et al., 2013], [Leon 3.0 documentation], called Pure Scala. Leon provides to Scala programmers the convenience to use existing Scala clauses to write specification constructs in Pure Scala, without special training in formal logic. For each function written in Pure Scala, Leon generates a verification condition[3] from the `require` precondition and `ensuring` post-condition clauses and tries to prove it [Leon 3.0 documentation]. To solve the generated verification condition, Leon combines an internal algorithm with external automated theorem proving tools [Microsoft Research, MIT licence, n.d.] and CVC4 [Clark Barrett, Cesare Tinelli, n.d.]. For each function, Leon's output can be: (1) *valid*, (2) *invalid* if there is at least one counter-example, and Leon returns one counter-example, and (3) *unknown*. The unknown result is usually due to a timeout or an internal error.

For constraint solving, Leon introduces the `choose` construct. Choose is used to solve a constraint (a Boolean expression) for a given value [Kuncak et al., 2013]. The expression `choose((res: B) => C(res))` evaluates to a value of type B satisfying the constraint C [Kuncak et al., 2013]. With using choose in a function's pre- or post-condition, Leon can generate a counter-example that satisfies the constraint in choose, but violates the function's verification condition.

We have developed a tool that maps the model properties to Scala functions with pre- and post-condition constraints. Afterwards, our tool runs Leon on the generated Scala functions to verify that for any input that satisfies the pre-condition, the post-condition is valid after the function execution.

B.3.4 Model-to-Text (Scala Code) Mapping

The generation of the Scala code from the model starts with parsing the SeamCAD XML file and building data structures for the services and processes. Using these data structures, three main entities of the code

[3]A verification condition is a statement of the form *precondition action postcondition*, which means that "if state *x* satisfies *precondition* and *action* transforms *x* to *y*, then state *y* satisfies *postcondition*" [Van Emden, 1979].

are generated: Scala case classes for the properties, Scala values (val[4] variables) for the services and Scala functions for the processes.

The names for the Scala values and functions are generated from the services' and processes' annotations in the SEAM model, which is the text in parenthesis. These annotations are used as identifiers that allow to map the values from the verification output back to the SEAM model.

Generation of Scala case classes for SEAM properties

Two Scala case classes, P and R, are generated as containers for the SEAM properties. They act as containers for the values that are used to find counter-examples for the ≪*feasibility*≫ properties. The P class contains all the variables that are defined in the ≪*final*≫ properties. The R class contains all the variables that are defined in the ≪*refinement_relation*≫ properties. These two classes are implemented in Scala using case classes and we generate them by extracting information from the data structures obtained after parsing the XML file. The generated P and R classes for our example are:

```
case class P(arrive_on_site: Int, get_geo_info: Int, secure_leak:
    ↪ Int, secure_site: Int, site_security: Int)
case class R(arrive_on_site: Int, secure: Int, time_to_secure: Int
    ↪ )
```

Generation of Scala values for SEAM services

For each service present in the model, a Scala value (val) is generated from the service identifier in parenthesis. If the service has ≪*final*≫ properties, then the service value is defined by an instance of P. Non-defined properties are set to zero in the instance of P. Otherwise, the property value is computed with the `choose` operator explained in Subsection B.3.3. In our example, the *Secure gas leak (sec)* service in the *Security and repair [w]* system is mapped to:

```
val s_sec = P(choose((arrive_on_site: Int) => arrive_on_site > 10
    ↪ && arrive_on_site <= 30), 0, 0, choose((secure_leak: Int) =
    ↪ > secure_leak > 10 && secure_leak <= 15), 0)
```

On the other hand, if the service has a corresponding process, i.e., service properties are ≪*computed*≫, then the service value is defined by a call to the function that maps that process (see next paragraph), return-

[4]"Scala has two kinds of variables, vals and vars. A val is similar to a final variable in Java. Once initialized, a val can never be reassigned." [Odersky et al., 2008].

ing an instance of the R case class. In our example, the *Manage gas leak (mng)* service in the *Utility company [w]* system is mapped to:

```
val s_mng = ps_mng(s_add, s_call, s_disp, s_fb, s_fb_sec, s_sap,
    ↪ s_sec)
```

where the `ps_mng` function is generated based on the mapping of the process in the model.

Generation of Scala functions for the processes in the model

Each process in the model is mapped to a Scala function. The input for this function represents the services connected to the process and the output is an instance of the R case class. Non-defined properties are set to zero in the instance of R. Otherwise the corresponding property is mapped to the expression present in the corresponding ≪*refinement_relation*≫.

The pre-conditions are generated to guarantee that the input properties of the process function match the connected service values.

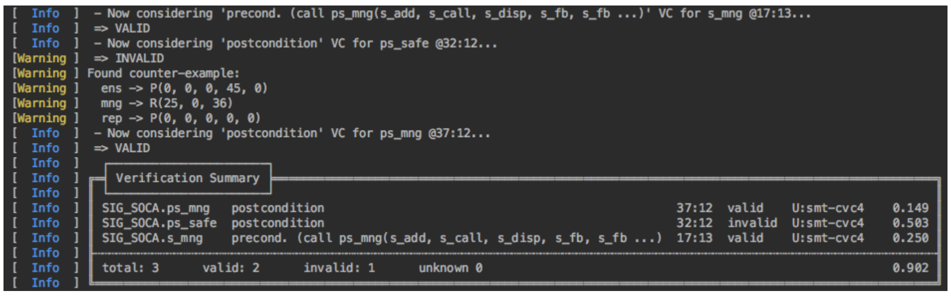
The post-condition applies to the output instance of R and is generated directly from the ≪*feasibility*≫ property of the mapped process.

The generated function for the *Manage site safety (safe)* process from our example is:

```
def ps_mng(add: P, call: P, disp: P, ecs: P, sap: P, sec: P): R =
    ↪ {
require(add == s_add && call == s_call && disp == s_disp && ecs ==
    ↪  s_ecs && sap == s_sap && sec == s_sec)
R(0, sec.secure_leak + sec.arrive_on_site + sap.get_geo_info)
} ensuring(res => res.time_to_secure < 45)
```

B.3.5 Scala Code Verification with Leon

Our tool passes the generated Scala code of the model to Leon. Leon statically verifies it by checking the feasibility properties that are mapped to post-conditions. Counter-examples are generated, if any exist. Fig. B.6 illustrates Leon's output in the terminal of the verification pre- and post-conditions for our example. There exists one invalid post-condition in our SEAM service design model: the process annotated with *(safe)*.



```
[  Info  ]  - Now considering 'precond. (call ps_mng(s_add, s_call, s_disp, s_fb, s_fb ...)' VC for s_mng @17:13...
[  Info  ]  => VALID
[  Info  ]  - Now considering 'postcondition' VC for ps_safe @32:12...
[Warning ]  => INVALID
[Warning ] Found counter-example:
[Warning ]   ens -> P(0, 0, 0, 45, 0)
[Warning ]   mng -> R(25, 0, 36)
[Warning ]   rep -> P(0, 0, 0, 0, 0)
[  Info  ]  - Now considering 'postcondition' VC for ps_mng @37:12...
[  Info  ]  => VALID
[  Info  ]
[  Info  ]    Verification Summary
[  Info  ]
[  Info  ]   SIG_SOCA.ps_mng   postcondition                                              37:12  valid    U:smt-cvc4    0.149
[  Info  ]   SIG_SOCA.ps_safe  postcondition                                              32:12  invalid  U:smt-cvc4    0.503
[  Info  ]   SIG_SOCA.s_mng    precond. (call ps_mng(s_add, s_call, s_disp, s_fb, s_fb ...)  17:13  valid    U:smt-cvc4    0.250
[  Info  ]
[  Info  ]   total: 3      valid: 2      invalid: 1      unknown 0                                                       0.902
[  Info  ]
```

Figure B.6 – Leon output in terminal.

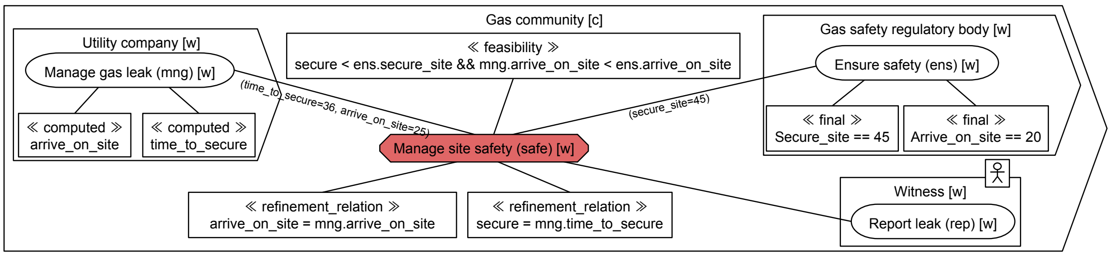

Gas community [c]
Utility company [w]
Manage gas leak (mng) [w]
≪ computed ≫ arrive_on_site
≪ computed ≫ time_to_secure
≪ feasibility ≫ secure < ens.secure_site && mng.arrive_on_site < ens.arrive_on_site
(time_to_secure=36, arrive_on_site=25)
(secure_site=45)
Manage site safety (safe) [w]
Gas safety regulatory body [w]
Ensure safety (ens) [w]
≪ final ≫ Secure_site == 45
≪ final ≫ Arrive_on_site == 20
≪ refinement_relation ≫ arrive_on_site = mng.arrive_on_site
≪ refinement_relation ≫ secure = mng.time_to_secure
Witness [w]
Report leak (rep) [w]

Figure B.7 – Verified service design model. It has annotations for the process specification causing the misalignment.

B.3.6 SEAM Model of the Alignment Verification Output

It is difficult for the user to interpret the output from the terminal and to locate the problem in the model (see Fig. B.6). Therefore, our tool automatically re-transforms the Leon output to a SeamCAD file. We parse the Leon output and with the help of the annotations in parenthesis in the service and process names, we mark the faulty process with red color as in Fig. B.7. The concrete values of counter-examples found are placed on the link between the process and the service. By doing this it is possible to graphically visualize the problem and identify the exact source of the misalignment. Fig. B.7 shows the verified SEAM model of our example. It is clear that the feasibility property of the *Manage site safety* process in the *Gas community [c]* system is violated.

The cause of this violation is the *arrive_on_site* time requirement. The time set by the *Gas safety regulation body [w]* system concerning the arrival on the site is used in the feasibility property in the service offering level, but this constraint is not transferred to the service implementation level, i.e., in the *Utility company [c]*. Service designers can use this visual verification output to further refine their specification in a way that their design choices are aligned among different specification levels.

For example, service designers of the *Utility company* should

1. include the restriction from the regulation body in the feasibility of the service implementation level, and
2. negotiate the operational level agreement with the *Security and repair team*, or the underpinning contract with the *Fire brigade*.

B.4 Limitations

The current implementation of the SeamCAD tool, and our automatic mapping and verification tool do not check for model's well-formedness. In case of a modeling mistake, the model-to-text code could not generate the necessary data structures used in the Scala code generation. In addition, for successful automatic verification of SEAM service specifications, the model properties must be basic Scala Boolean expressions. Our tool directly maps all properties' values to Scala code. In these cases, the service designer is left to find the problem based on the console output.

Service designers must also be attentive on the measurement units of quantities put in the properties. In our example we use minutes, but we do not specify in the model that all quantities are minutes. Consequently, every occurrence of a property variable must have the same unit of measurement and must be expressed in the same order of magnitude. Our tool is not able to detect or perform automatic conversion of units of measurement. It is however possible to define multiple properties in the same model to capture different units.

Finally, Leon's support for only Pure Scala programs limits the expressiveness of service specifications in the service design model as well. Currently it is impossible to use external libraries that deal with dynamic units of measurement conversions and static type checking.

B.5 Future Work

We plan to apply our approach in a real project. Many projects we encounter deal with non-functional requirements, such as security and quality, so we are researching ways of representing and quantifying such properties. In addition, we are working on including the previous work [Rychkova, 2008] on verifying behavior properties in SEAM models.

The example we show is a simple one, as our goal is to present our approach in details and to formally describe what we mean by alignment. With SEAM, service designers are not bound to specify only two levels of a service. Every service in a system as a whole can be refined, and the specification of its implementation can be modeled. In this recursive manner, SEAM is used to show specifications from the service strategy level up until the lowest IT level. We still need to conduct an evaluation (e.g. user study) of the usefulness of the automated verification with SEAM. Our current expectation is that in combination with Leon our approach has the potential to become a powerful verification tool in the field of MDD.

B.6 Conclusions

We present a model-driven approach for automatic alignment verification of quantitative properties modeled in SEAM service design specification. We demonstrate service modeling on a simple example: services modeled at two levels. A feasibility property is specified to capture the desired behavior of services in each level. Feasibility properties of all levels are used check the alignment correctness of each level and among levels. We also show refinement of properties from the service offering (higher) level to the service implementation (lower) level. The tool for this automated verification:

1. takes a SEAM model as an input,
2. translates the model to Scala code,
3. verifies the code with Leon, and
4. displays the result in a new annotated SEAM model.

The novelty of our approach is using software verification tools in the early stage of service design. The practical contribution emerges from using Leon; service designers do not need to learn a formal language, but they only need to know SEAM modeling and basic Scala expressions to capture and verify their design choices. The model-to-text translation and the verification are running in the background, so there is no overhead in learning and manually transforming the service specification to a formal model. The visualization of the verification result in a SEAM model gives feedback about the cause of the misalignment and allows to refine the service specification.

SEAM recursively uses the same notation, so service designers are free to model any number of levels. Our approach is applicable to the complete organizational hierarchy, starting from the business down to the IT level.

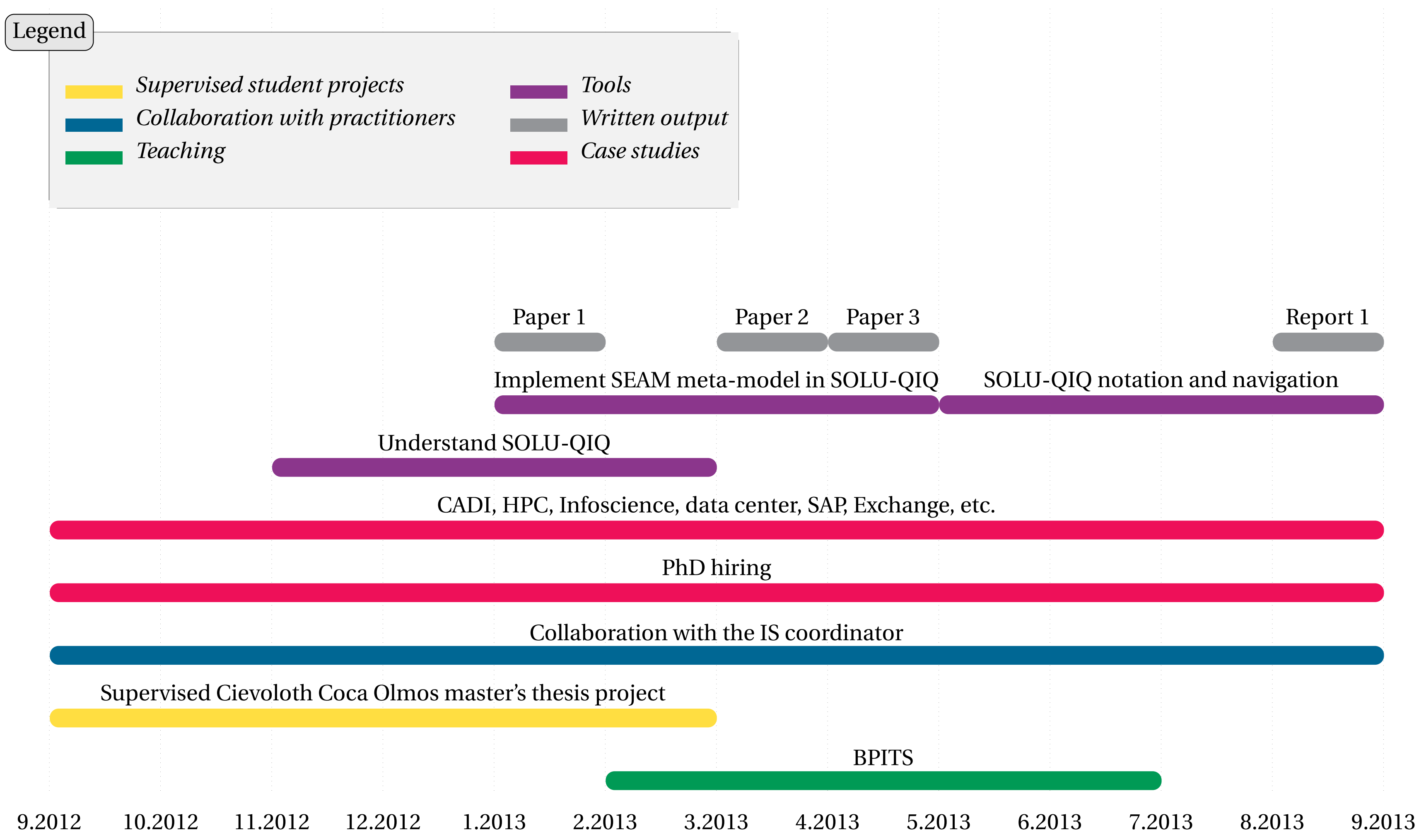
Therefore, because it is a visual approach, a wider audience is able to benefit from designing correctly aligned service specifications among different levels of the organizations and IT systems (such as students, service designers, software architects and developers).

C Thesis Timeline

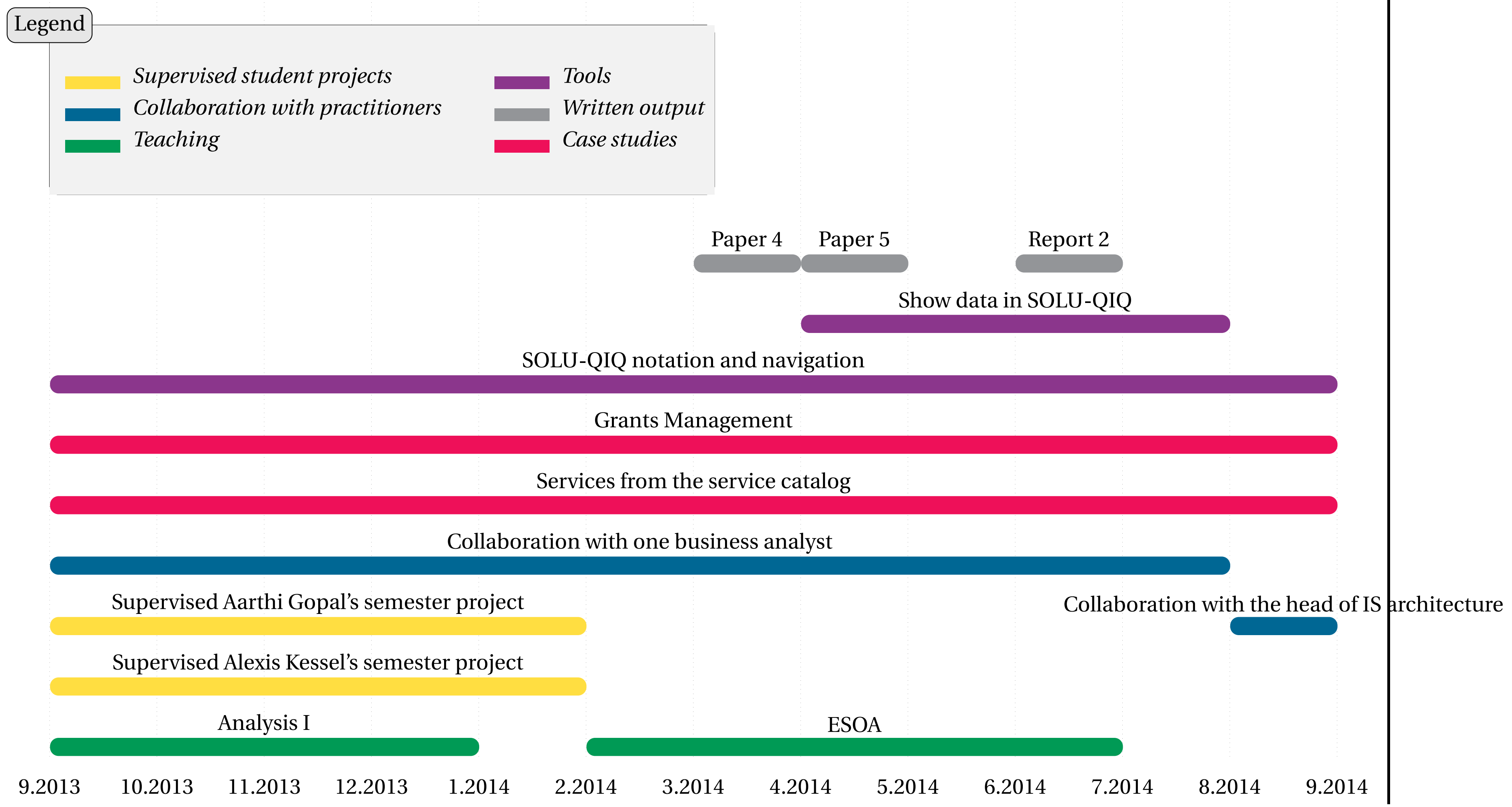
In this Appendix we present a visual account in the form of a timeline[1] of the activities, responsibilities, collaborations and outputs during this PhD thesis. Research activities, such as readings, discussions with other fellow researchers and the thesis advisor, are not represented, as they occurred throughout the entire PhD.

[1]The timeline is modified from the original by Vitaly Repin (vitaly.repin@gmail.com), licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License, available on GitHub

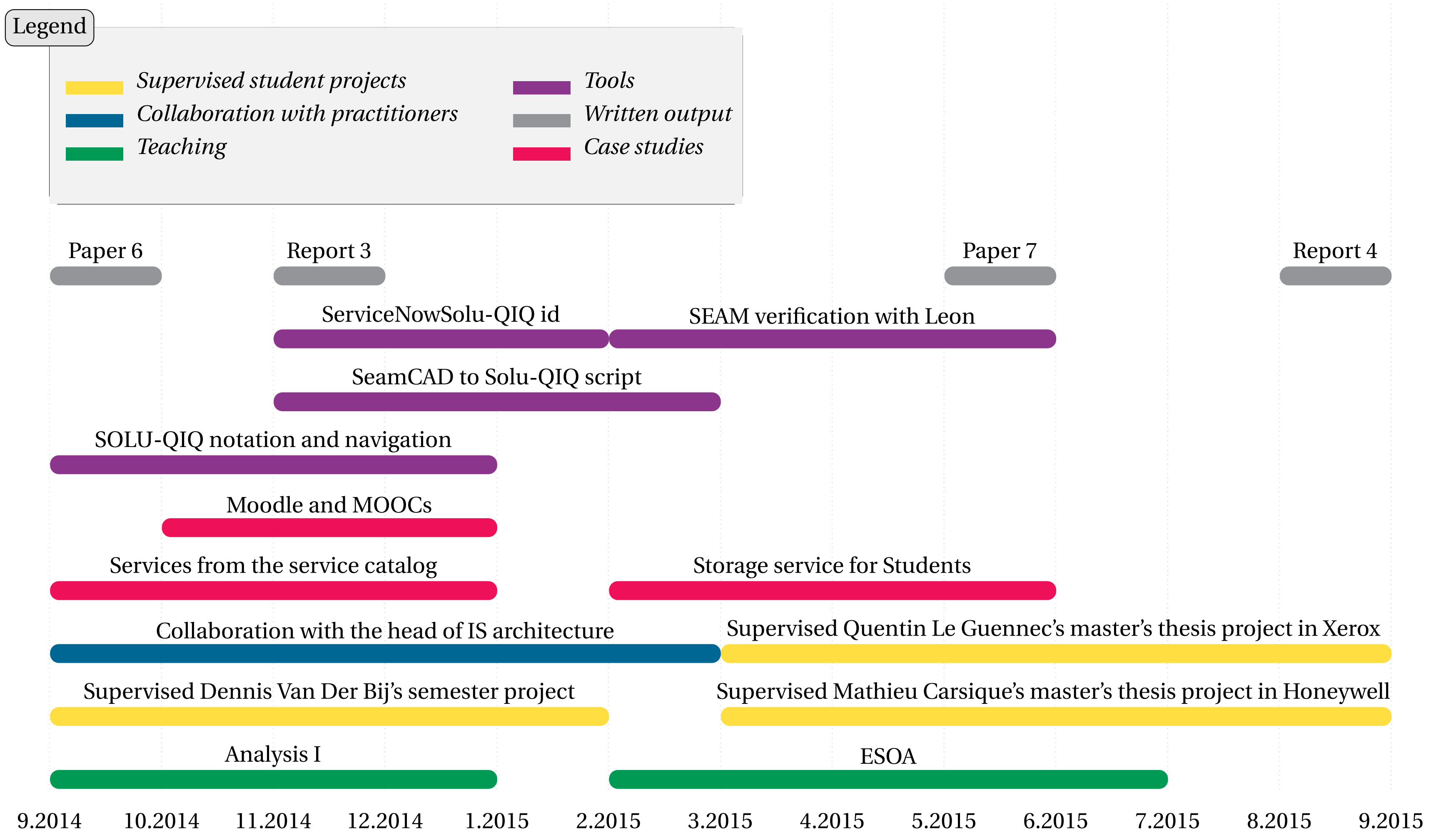
1st year: **9.2012 – 9.2013**



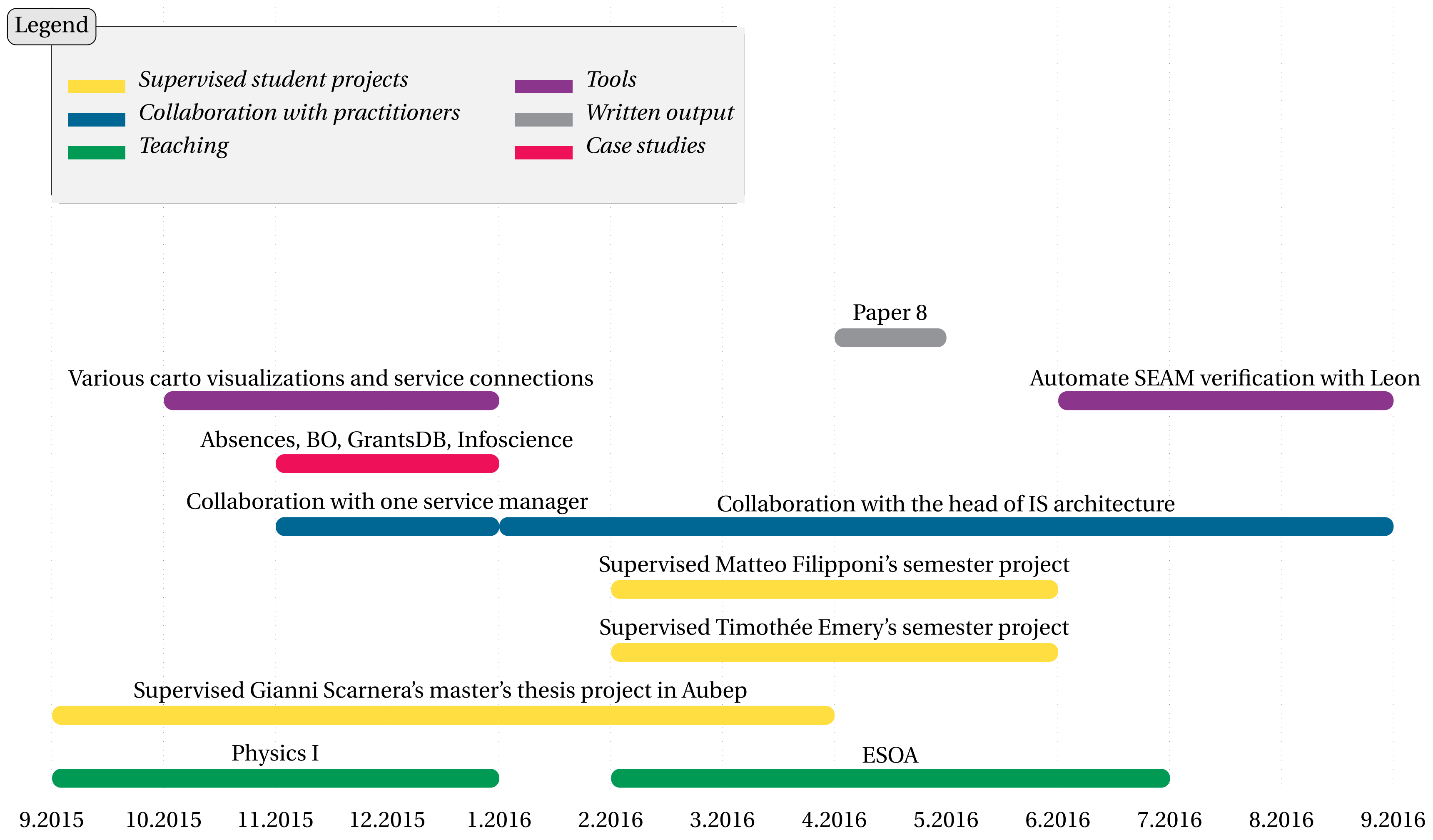
2^{nd} year: **9.2013 – 9.2014**



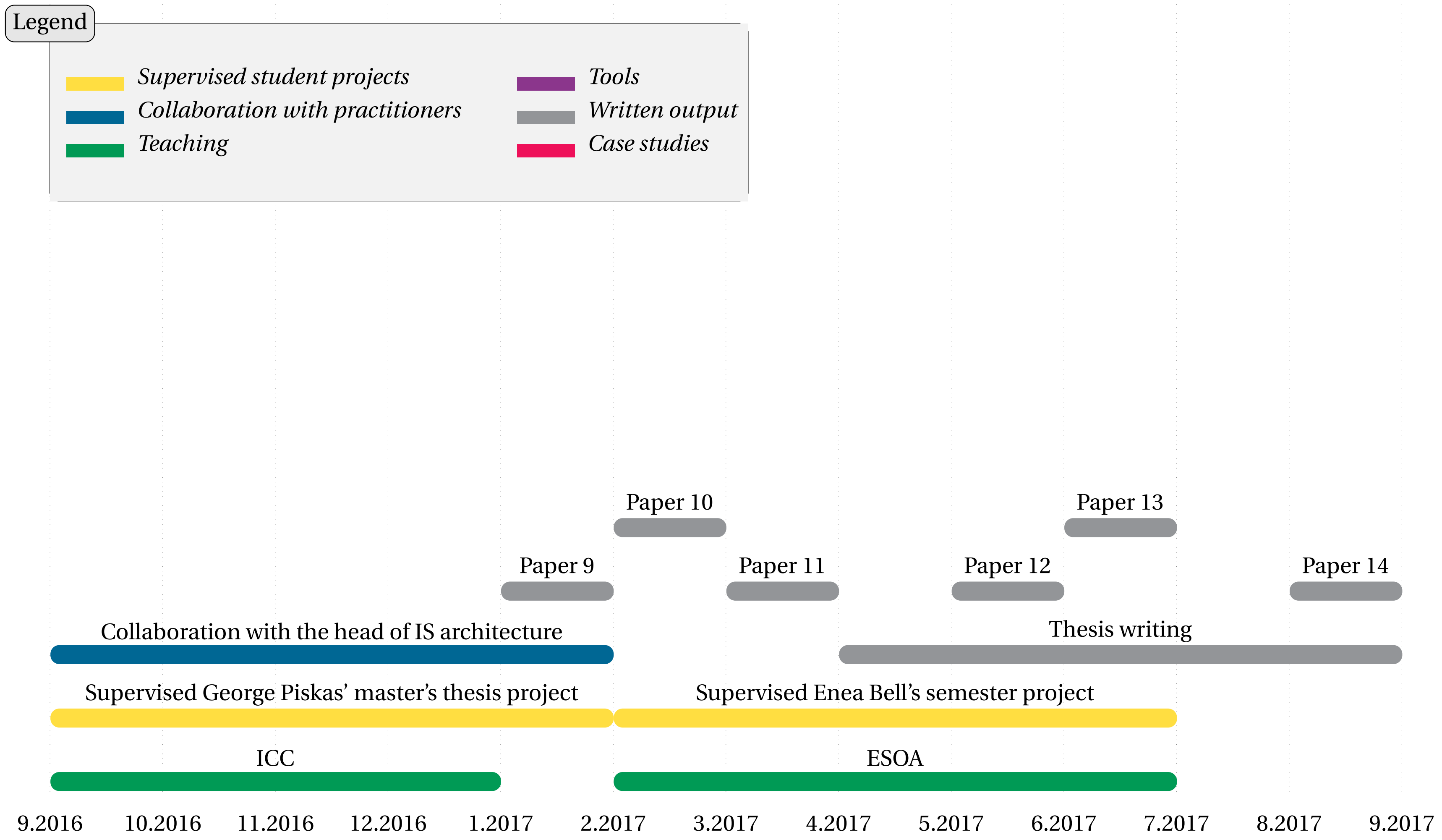
3rd year: **9.2014 – 9.2015**



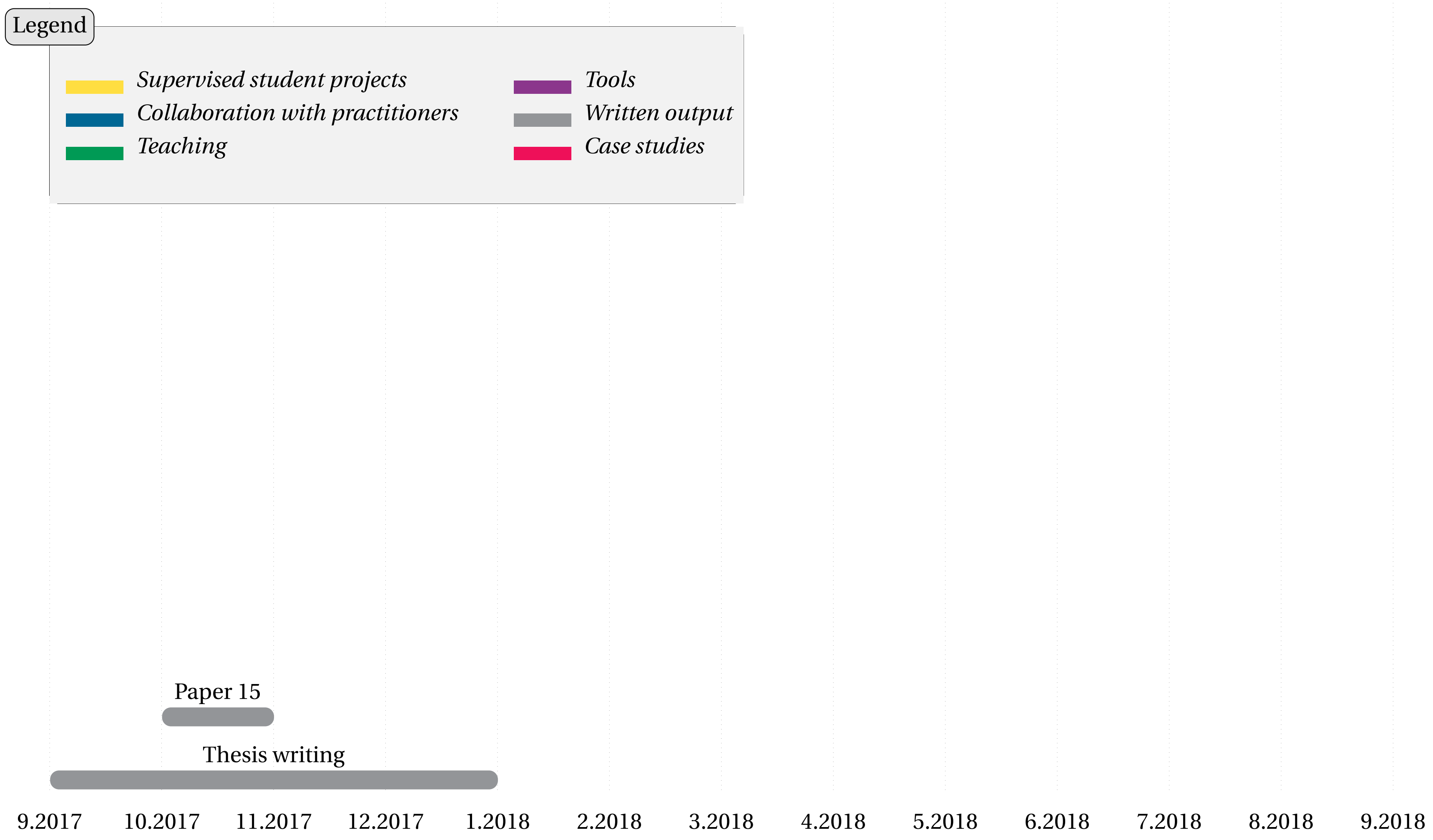
4th year: **9.2015 – 9.2016**



5th year: **9.2016 – 9.2017**



6th year: **9.2017 – 9.2018**



Submissions to conferences (Papers):

1. “Towards the Definition, Implementation and Communication of an IT Strategy: the Case of IT Strategy at EPFL” [Tapandjieva et al., 2013] presented at the BUSITAL 2013 workshop.
2. “A Philosophical Foundation for Business and IT Alignment in Enterprise Architecture with the Example of SEAM” [Regev et al., 2013] presented at the Third International Symposium on BMSD 2013.
3. “Business and IT Design with SEAM: An Illustration with the PhD Hiring Process at École Polytechnique Fédérale de Lausanne” [Popescu et al., 2013] presented at the SMC 2013 conference.
4. “Specification and Implementation of a Meta-model for Information Systems Cartography” [Tapandjieva and Wegmann, 2014] presented as a poster in the CAiSE 2014 Forum.
5. “Patterns for Value-Added Services Illustrated with SEAM” [Tapandjieva et al., 2014] presented at an EDOC 2014 workshop.
6. Submitted an extended version of [Tapandjieva and Wegmann, 2014]. – **Rejected**
7. “IT Service Alignment Verification of Quantitative Properties in Service Design” [Tapandjieva and Wegmann, 2015] presented at an EDOC 2015 workshop.
8. “SLA: to Sign or Not to Sign” [Tapandjieva et al., 2016] presented at the 2016 STPIS workshop.
9. “A Return on Our Experience of Modeling a Service-Oriented Organization in a Service Cartography” [Tapandjieva et al., 2017a] presented at the 2017 IESS conference.
10. Submitted a paper at the 14th IEEE International Conference on Services Computing (SCC). The paper is a continuation of [Tapandjieva and Wegmann, 2015] and describes an improved and automated verification of SEAM models with quantitative properties. – **Rejected**
11. “Coopetition and Ecosystems: case of Amazon.com” [Wegmann et al., 2018], to appear in 2018.
12. Submitted a paper at the AIS International Conference on Information Systems. The paper is a continuation of [Tapandjieva and Wegmann, 2015] and describes an improved and automated verification of SEAM models with quantitative properties. – **Rejected**
13. Submitted a paper at HICSS-51, the Hawaii International Conference on System Sciences. The paper describes services as a unifying metaphor for IT and mission alignment in an IS organization, the initial idea of Chapter 5 in this thesis. – **Rejected**.
14. A continuation of [Tapandjieva and Wegmann, 2015], paper titled: “Alignment Verification in the Early Stage of Service Design” [Tapandjieva et al., 2017b] presented at the 2017 SOCA conference.
15. “Ontology for SEAM Service Models” [Tapandjieva and Wegmann, 2018], paper based on the work presented in Chapter 4, accepted as a short paper, to be presented at the ICEIS 2018 conference.

Reports:

1. Candidacy exam report: Enterprise Architecture Theory and Practice for the Federated Enterprise
2. Segmentation document
3. SEAM documentation and tutorial
4. Internal service management document (description of services and service aggregations)

Bibliography

AB+Software. SOLU-QIQ - Accueil, n.d. URL https://www.abplussoftware.fr/en/solu-qiq/. Accessed September 2017.

Russell L Ackoff and Jamshid Gharajedaghi. Reflections on systems and their models. *Systems Research and Behavioral Science*, 13(1):13–23, 1996.

Pervaiz K Ahmed and Mohammed Rafiq. Internal marketing issues and challenges. *European Journal of marketing*, 37(9):1177–1186, 2003.

Carolina Alaceva and Lazar Rusu. Barriers in achieving business/it alignment in a large swedish company: What we have learned? *Computers in Human Behavior*, 51:715–728, 2015.

Iyas Alloush, Yvon Kermarrec, and Siegfried Rouvrais. A transversal alignment between measurements and enterprise architecture for early verification of telecom service design. In *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*, pages 245–256. Springer, 2013.

Alloy. Alloy: a language & tool for relational models, n.d. URL http://alloy.mit.edu/. Accessed September 2017.

Jim Amsden. Integrating BPMN and SoaML, part 1. motivation and approach. https://www.ibm.com/developerworks/rational/library/integrate-bpmn-soaml/index.html, 2014.

Valerie Arraj. ITIL®: the basics. Technical report, The APM Group and The Stationery Office, 2013.

Ali Arsanjani. Service-oriented modeling and architecture. https://www.ibm.com/developerworks/library/ws-soa-design1/, 2004.

Lerina Aversano, Carmine Grasso, and Maria Tortorella. Managing the alignment between business processes and software systems. *Information and Software Technology*, 72:171–188, 2016.

David E Avison, Francis Lau, Michael D Myers, and Peter Axel Nielsen. Action research. *Communications of the ACM*, 42(1):94–97, 1999.

DE Avison, RM Davison, and J Malaurent. Information systems action research: Debunking myths and overcoming barriers. *Information & Management*, 2017.

Biljana Bajić-Bizumić, Claude Petitpierre, Hieu Chi Huynh, and Alain Wegmann. A model-driven environment for service design, simulation and prototyping. In *Exploring Services Science*, pages 200–214. Springer, 2013.

Bela H Banathy and Patrick M Jenlink. Systems inquiry and its application in education. *Handbook of research for educational communications and technology*, pages 37–58, 2003.

Indranil R. Bardhan, Haluk Demirkan, P. K. Kannan, Robert J. Kauffman, and Ryan Sougstad. An Interdisciplinary Perspective on IT Services Management and Service Science. *Journal of Management Information Systems*, 26(4):13–64, April 2010. ISSN 0742-1222. doi: 10.2753/MIS0742-1222260402.

Richard Baskerville. What design science is not. *European Journal of Information Systems*, 17: 441–443, 2008.

Richard L Baskerville. Investigating information systems with action research. *Communications of the AIS*, 2(3es):4, 1999.

Stafford Beer. The Viable System Model: Its Provenance, Development, Methodology and Pathology. *The Journal of the Operational Research Society*, 35(1):7–25, 1984. ISSN 0160-5682. doi: 10.2307/2581927.

Bruce L Berg. *Methods for the social sciences*. Pearson Education Inc, United States of America, 2004.

Jacques Bertin. *Semiology of graphics*. University of Wisconsin Press, Madison, Wis, 1983.

Hugh R Beyer and Karen Holtzblatt. Apprenticing with the customer. *Communications of the ACM*, 38(5):45–52, 1995.

Régis Blanc, Viktor Kuncak, Etienne Kneuss, and Philippe Suter. An overview of the leon verification system: Verification by translation to recursive functions. In *Proceedings of the 4th Workshop on Scala*, page 1. ACM, 2013.

Mike Bostock. D3.js, 2017. URL https://d3js.org/.

BPMN v2.0.2. OMG Business Process Model And Notation, Version 2.0.2. OMG Document Number formal/13-12-09 (http://www.omg.org/spec/BPMN/2.0.2/), 2014.

Corina Braun, Corina Braun, Karsten Hadwich, and Karsten Hadwich. Determinants of perceived internal service complexity: An empirical analysis of promoting and limiting complexity factors. *European Business Review*, 29(1):123–152, 2017.

Manfred Bruhn. Internal service barometers: Conceptualization and empirical results of a pilot study in Switzerland. *European journal of Marketing*, 37(9):1187–1204, 2003.

Mario Bunge. *Dictionary of Philosophy.* Prometheus Books, 1999.

Gibson Burrell and Gareth Morgan. *Sociological paradigms and organisational analysis: Elements of the sociology of corporate life.* Routledge, 2017.

BYOD-Wikipedia. Bring your own device (BYOD) - wikipedia, n.d. URL https://en.wikipedia.org/wiki/Bring_your_own_device. Accessed December 2017.

Jordi Cabot, Robert Clarisó, and Daniel Riera. UMLtoCSP: A tool for the formal verification of UML/OCL models using constraint programming. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 547–548. ACM, 2007.

Jorge Cardoso, Alistair Barros, Norman May, and Uwe Kylau. Towards a unified service description language for the internet of services: Requirements and first developments. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 602–609. IEEE, 2010.

Jorge Cardoso, Björn Schmitz, and Axel Kieninger. *Service Research*, pages 325–357. Springer International Publishing, Cham, 2015. ISBN 978-3-319-23195-2. doi: 10.1007/978-3-319-23195-2_10.

John S Carson. Model verification and validation. In *Simulation Conference, 2002. Proceedings of the Winter*, volume 1, pages 52–58. IEEE, 2002.

Alison Cartlidge, Colin Rudd, Marco Smith, Paul Wigzel, Stuart Rance, Sue Shaw, and Theresa Wright. An introductory overview of itil® 2011. Technical report, The IT Service Management Forum (itSMF) UK, Norwich, 2012.

Peter Checkland. *Systems Thinking, Systems Practice: Includes a 30-Year Retrospective.* John Wiley & Sons, 1999.

Peter Checkland and Sue Holwell. *Information, Systems and Information Systems: Making Sense of the Field.* Wiley, Chichester; New York, 1997. ISBN 978-0-471-95820-8.

Henry Chesbrough and Jim Spohrer. A research manifesto for services science. *Communications of the ACM*, 49(7):35–40, 2006.

Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, et al. Web services description language (wsdl) 1.1, 2001.

CIO web. What is ITIL? IT Infrastructure Library definitions and solutions, 2017. URL https://www.cio.com/article/2439501/itil/infrastructure-it-infrastructure-library-itil-definition-and-solutions.html.

Clark Barrett, Cesare Tinelli. About CVC4, n.d. URL http://cvc4.cs.nyu.edu/web/. Accessed May 2015.

CMMN v1.1. OMG Case Model Management and Notation, Version 1.1. OMG Document Number formal/16-12-01 (http://www.omg.org/spec/CMMN/1.1/), 2016.

David Coghlan and Teresa Brannick. *Doing action research in your own organization.* Sage, 2014.

Robert Cole, Sandeep Purao, Matti Rossi, and Maung Sein. Being proactive: where action research meets design research. *ICIS 2005 Proceedings*, page 27, 2005.

Confluence. Confluence, n.d. URL https://www.atlassian.com/software/confluence. Accessed December 2017.

daveirwin1 and Bill Anderson. Service Map in Operations Management Suite (OMS), 2016. URL https://docs.microsoft.com/en-us/azure/operations-management-suite/operations-management-suite-service-map. Accessed September 2017.

Haluk Demirkan, Robert J Kauffman, Jamshid A Vayghan, Hans-Georg Fill, Dimitris Karagiannis, and Paul P Maglio. Service-oriented technology and management: Perspectives on research and practice for the coming decade. *Electronic Commerce Research and Applications*, 7(4):356–376, 2009.

Martin Dodge, Mary McDerby, and Martin Turner. *The Power of Geographical Visualizations*, pages 1–10. John Wiley & Sons, Ltd, 2008.

Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Introduction to Business Process Management*, pages 1–31. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-33143-5. doi: 10.1007/978-3-642-33143-5_1. URL https://doi.org/10.1007/978-3-642-33143-5_1.

EA-tool. Enterprise Architect - UML design tools and UML CASE tools for software development, n.d. URL http://www.sparxsystems.com/products/ea/. Accessed December 2017.

EAMS. Enterprise Cartography, n.d. URL http://www.linkconsulting.com/eams/enterprise-cartography/. Accessed September 2017.

Bo Edvardsson, Bård Tronvoll, and Thorsten Gruber. Expanding understanding of service exchange and value co-creation: a social construction approach. *Journal of the Academy of Marketing Science*, 39(2):327–339, 2011.

Wilco Engelsman, Henk Jonkers, and Dick Quartel. Archimate® extension for modeling and managing motivation, principles, and requirements in togaf®. *White paper, The Open Group*, 2011.

EPFL IS Architecture. Architecture | EPFL-SI, n.d. URL http://informationsystem.epfl.ch/architectureEN. Accessed September 2017.

Ulrich Frank. Multi-perspective enterprise modeling (memo) conceptual framework and modeling languages. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 1258–1267. IEEE, 2002.

Ariel Fuxman, Lin Liu, John Mylopoulos, Marco Pistore, Marco Roveri, and Paolo Traverso. Specifying and analyzing early requirements in Tropos. *Requirements Engineering*, 9(2): 132–150, 2004.

Laura Mata Garcia. Understanding design thinking, exploration and exploitation: Implications for design strategy. *IDBM Papers*, 2:150–161, 2012.

Dimitrios Georgakopoulos and Michael P. Papazoglou. Overview of service-oriented computing. In *Service-Oriented Computing*, chapter 1. The MIT Press, 2008. ISBN 0262072963, 9780262072960.

Jennifer E Gerow, Jason Bennett Thatcher, and Varun Grover. Six types of it-business strategic alignment: An investigation of the constructs and their measurement. *European Journal of Information Systems*, 24(5):465–491, 2015.

Imran Ghani. *Handbook of research on emerging advancements and technologies in software engineering*. IGI Global, 2014.

Jamshid Gharajedaghi. *Systems thinking: Managing chaos and complexity: A platform for designing business architecture*. Elsevier, 2011.

Jonathan Glancey. The London underground map: The design that shaped a city, 2015. URL http://www.bbc.com/culture/story/20150720-the-london-underground-map-the-design-that-shaped-a-city. Accessed December 2017.

Arash Golnam. *Problem Structuring with the Systemic Enterprise Architecture Method: Representation of Systems and Value in Business Contexts and Integration with Operations Research Methods*. PhD thesis, EPFL, 2013.

Arash Golnam, Gil Regev, Alain Wegmann, and Sofia Kyriakopoulou. The integration of an RE method and AHP: A pilot study in a large Swiss bank. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 308–313. IEEE, 2013.

John AJ Gowlett. The discovery of fire by humans: a long and convoluted process. *Phil. Trans. R. Soc. B*, 371(1696):20150164, 2016.

Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995. ISSN 1071-5819.

Tom Gruber. Ontology, 2009.

Egon G Guba, Yvonna S Lincoln, et al. Competing paradigms in qualitative research. *Handbook of qualitative research*, 2(163-194):105, 1994.

Evert Gummesson. The marketing of professional services – an organizational dilemma. *European Journal of Marketing*, 13(5):308–318, 1979.

Stefan Hallerstede, Michael Jastram, and Lukas Ladenberger. A method and tool for tracing requirements into specifications. *Science of Computer Programming*, 82:2–21, 2014.

Kristina Heinonen and Tore Strandvik. Customer-dominant logic: foundations and implications. *Journal of Services Marketing*, 29(6/7):472–484, 2015.

Kristina Heinonen, Tore Strandvik, and Päivi Voima. Customer dominant value formation in service. *European Business Review*, 25(2):104–123, 2013.

John C Henderson and Harihara Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *IBM systems journal*, 32(1):472–484, 1993.

A.R. Hevner, S.T. March, J. Park, and S. Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

T. Peter Hill. On goods and services. *Review of income and wealth*, 23(4):315–338, 1977.

Anders Hjalmarsson, Stefan Cronholm, and Hannes Göbel. Hypotheses for examining itsm-framework adoption. In *ITSM Nordic Research Workshop*, 2016.

Thomas Hofweber. Logic and ontology. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2017 edition, 2017. Accessed October 2017.

Lou Hunnebeck. *ITIL Service Design*. The Stationery Office, 2011. ISBN 0113313055.

Juhani Iivari. A paradigmatic analysis of contemporary schools of is development. *European Journal of Information Systems*, 1(4):249, 1991.

Juhani Iivari and John Venable. Action research and design science research-seemingly similar but decisively dissimilar. In *ECIS*, pages 1642–1653, 2009.

ISO/IEC 19506:2012(en). Information technology – Object Management Group Architecture-Driven Modernization (ADM) – Knowledge Discovery Meta-Model (KDM), 2012. URL https://www.iso.org/standard/32625.html.

ISO/IEC 20000-1:2011(en). Information technology - Service management - Part 1: Service management system requirements, 2011. URL https://www.iso.org/standard/51986.html.

ISO/IEC 20000-2:2012(en). Information technology - Service management - Part 2: Guidance on the application of service management systems, 2012. URL https://www.iso.org/standard/51987.html.

Ray Ison. Systems thinking and practice for action research. In Peter W. Reason and Hilary Bradbury, editors, *The Sage Handbook of Action Research Participative Inquiry and Practice (2nd edition)*, pages 139–158. Sage Publications, 2008. doi: 10.4135/9781848607934.

IT4IT v2.0. IT4IT Reference Architecture, Version 2.0, n.d. URL http://pubs.opengroup.org/it4it/refarch20/index.html. Accessed September 2017.

Daniel Jackson. Alloy: A lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(2):256–290, 2002.

Pertti Järvinen. Action research is similar to design science. *Quality & Quantity*, 41(1):37–54, 2007.

Robert Johnston and Graham Clark. *Service operations management: improving service delivery*. Pearson Education, 2008.

Andrew Josey. *TOGAF® Version 9.1-A Pocket Guide*. Van Haren, 2011.

Andrew Josey, Marc Lankhorst, Iver Band, Henk Jonkers, and Dick Quartel. An introduction to the ArchiMate® 3.0 specification. *White Paper from The Open Group*, 2016.

Andrew Josey, Rob Akershoek, Charles Betz, Christopher Davis, Sue Desiderio, Sylvain Marie Arismore, David Morlitz, and Lars Rossen. The it4it™ reference architecture, version 2.1: A pocket guide., 2017.

Leon Kappelman, Ephraim McLean, Vess Johnson, Russell Torres, Quynh Nguyen, Chris Maurer, and Mark Snyder. The 2016 sim it issues and trends study. *MIS Quarterly Executive*, 16(1), 2017.

George J. Klir. *Facets of Systems Science*, volume 15 of *IFSR International Series in Systems Science and Systems Engineering*. Springer US, 2001. ISBN 9780306466236.

David A Kolb. *Experiential learning: Experience as the source of learning and development*. FT press, 2014.

Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional, 2005.

Thomas S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, 1962.

Viktor Kuncak, Etienne Kneuss, and Philippe Suter. Executing specifications using synthesis and constraint solving. In *International Conference on Runtime Verification*, pages 1–20. Springer Berlin Heidelberg, 2013.

LAMS. LAMS, n.d. URL http://lams.epfl.ch/. Accessed September 2017.

LAMS. SEAM, n.d.a. URL http://lams.epfl.ch/seam/. Accessed September 2017.

LAMS. SeamCAD, n.d.b. URL http://lams.epfl.ch/seamcad/.

M. Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, 2009.

Lam-Son Le and Alain Wegmann. Definition of an object-oriented modeling language for enterprise architecture. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 222a–222a. IEEE, 2005.

Lam-Son Lê and Alain Wegmann. Hierarchy-oriented modeling of enterprise architecture using reference-model of open distributed processing. *Computer Standards & Interfaces*, 35 (3):277–293, 2013.

Leon 3.0 documentation. Leon documentation — Leon 3.0 documentation, n.d. URL http://leon.epfl.ch/doc/index.html. Accessed September 2017.

Jenny Leonard and Peter Seddon. A meta-model of alignment. *Communications of the Association for Information Systems*, 31(1):11, 2012.

Kurt Lewin. Action research and minority problems. *Journal of social issues*, 2(4):34–46, 1946.

Kurt Lewin. Frontiers in Group Dynamics: II. Channels of Group Life; Social Planning and Action Research. *Human Relations*, 1(2):143–153, November 1947. ISSN 0018-7267. doi: 10.1177/001872674700100201.

C. Longépé. *The Enterprise Architecture IT Project: the Urbanisation Paradigm.* Butterworth-Heinemann, 2003.

Jerry Luftman. Assessing business-IT alignment maturity. *Strategies for information technology governance*, 4:99, 2004.

Jerry Luftman and Rajkumar Kempaiah. An update on business-it alignment: "a line" has been drawn. *MIS Quarterly Executive*, 6(3), 2007.

Jerry N. Luftman. *Competing in the Information Age: Align in the Sand.* Oxford University Press, 2003. ISBN 9780198036166.

Robert F Lusch, Stephen L Vargo, and Gregor Wessels. Toward a conceptual foundation for service science: Contributions from service-dominant logic. *IBM systems journal*, 47(1): 5–14, 2008.

Paul P Maglio, Savitha Srinivasan, Jeffrey T Kreulen, and Jim Spohrer. Service systems, service scientists, ssme, and innovation. *Communications of the ACM*, 49(7):81–85, 2006.

Paul P Maglio, Stephen L Vargo, Nathan Caswell, and Jim Spohrer. The service system is the basic abstraction of service science. *Information Systems and e-business Management*, 7(4): 395–406, 2009.

Salvatore T March and Gerald F Smith. Design and natural science research on information technology. *Decision support systems*, 15(4):251–266, 1995.

Mike A Marin. Introduction to the case management model and notation (CMMN). *arXiv preprint arXiv:1608.05011*, 2016.

M Lynne Markus and Mark Keil. If we build it, they will come: Designing information systems that people want to use. *Sloan Management Review*, 35(4):11, 1994.

Judy McKay and Peter Marshall. The dual imperatives of action research. *Information Technology & People*, 14(1):46–59, 2001.

Donella H. Meadows. *Thinking in Systems: A Primer*. Chelsea Green Publishing, 2008.

MEGA. HOPEX | MEGA, n.d. URL http://www.mega.com/en/product/hopex. Accessed September 2017.

Microsoft Research, MIT licence. Z3prover, n.d. URL https://github.com/Z3Prover/z3. Accessed May 2015.

James Grier Miller. *Living Systems*. McGraw Hill, New York, 1978. ISBN 978-0-07-042015-1.

John Mingers. A classification of the philosophical assumptions of management science methods. *Journal of the Operational Research Society*, 54(6):559–570, 2003.

John Mingers. *Realising systems thinking: knowledge and action in management science*. Springer Science & Business Media, 2006.

Henry Mintzberg. The fall and rise of strategic planning. *Harvard business review*, 72(1): 107–114, 1994.

Henry Mintzberg, Bruce Ahlstrand, and Joseph Lampel. *Strategy Safari: A Guided Tour Through The Wilds of Strategic Mangament*. Simon and Schuster, 2005.

Daniel Moody. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009.

Shoichi Morimoto. A survey of formal verification for business process modeling. In *International Conference on Computational Science*, pages 514–522. Springer, 2008.

Enid Mumford. The participation of users in systems design: an account of the origin, evolution, and use of the ETHICS method. *Participatory Design. Principles and Practices*, pages 257–270, 1993.

Harold G Nelson and Erik Stolterman. *The Design Way: Intentional Change in an Unpredictable World*. The MIT Press, 2012.

Richard Normann. *Service Management : Strategy and Leadership in Service Business, 3rd Edition*. Wiley, 2001. ISBN 0471494399.

Richard Normann and Rafael Ramirez. From value chain to value constellation: Designing interactive strategy. *Harvard business review*, 71(4):65–77, 1993.

Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala*. Artima Inc, 2008.

Katri Ojasalo and Jukka Ojasalo. Adapting business model thinking to service logic: an empirical study on developing a service design tool. *THE NORDIC SCHOOL*, 309, 2015.

OMG. About OMG. http://www.omg.org/about/, n.a.

Wanda J Orlikowski and Jack J Baroudi. Studying information technology in organizations: Research approaches and assumptions. *Information systems research*, 2(1):1–28, 1991.

Alexander Osterwalder. *The business model ontology: A proposition in a design science approach*. PhD thesis, L'Ecole des HEC de l'Université de Lausanne, 2004.

Alexander Osterwalder and Yves Pigneur. *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons, 2010.

Jeffrey Parsons and Yair Wand. Using cognitive principles to guide classification in information systems modeling. *Management Information Systems Quarterly*, 32(4):839–868, 2008.

Michael Quinn Patton. *Qualitative evaluation and research methods*. SAGE Publications, inc, 1990.

Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

Michael Pidd. *Tools for Thinking: Modelling in Management Science*. Wiley, second edition edition, 2003. ISBN 978-0-470-84795-6.

Georgios Plataniotis, Sybren De Kinderen, Qin Ma, and Henderik Proper. Traceability and modeling of requirements in enterprise architecture from a design rationale perspective. In *Ninth IEEE conference on Research Challenges in Information Systems (RCIS 2015)*, 2015.

George Popescu, Gorica Tapandjieva, and Alain Wegmann. Business and IT design with SEAM: An illustration with the PhD hiring process at École Polytechnique Fédérale de Lausanne. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1938–1943. IEEE, 2013.

Valentina Popescu, Bhalchandra Pandit, and Virginia Smith. Service modeling language, version 1.1, W3C recommendation. https://www.w3.org/TR/sml/, 2009.

Michael E Porter. *Competitive advantage: Creating and sustaining superior performance*. Simon and Schuster, 2008.

Thomas C Powell and Anne Dent-Micallef. Information technology as competitive advantage: The role of human, business, and technology resources. *Strategic management journal*, 18 (5):375–405, 1997.

Otto Preiss. *Foundations of Systems and Properties: Methodological Support for Modeling Properties of Software-Intensive Systems*. PhD thesis, EPFL, 2004.

Balasubramaniam Ramesh and Matthias Jarke. Toward reference models of requirements traceability. *IEEE Trans. Software Eng.*, 27(1):58–93, 2001. doi: 10.1109/32.895989.

Stuart Rance. *ITIL Service Transition (Best Management Practices)*. The Stationery Office, 2011. ISBN 0113313063.

Ivan S Razo-Zapata, Pieter De Leenheer, Jaap Gordijn, and Hans Akkermans. Service network approaches. In *Handbook of service description*, chapter 3, pages 45–74. Springer, 2012.

Iván S Razo-Zapata, Jaap Gordijn, Pieter De Leenheer, and Roel Wieringa. e3service: A critical reflection and future research. *Business & information systems engineering*, 57(1):51–59, 2015.

Gil Regev. *A systemic paradigm for early it system requirements based on regulation principles: the lightswitch approach*. PhD thesis, EPFL, 2003.

Gil Regev and Alain Wegmann. Remaining fit: On the creation and maintenance of fit. In *Proceedings of BPMDS'04*. Citeseer, 2004.

Gil Regev and Alain Wegmann. Where do goals come from: the underlying principles of goal-oriented requirements engineering. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 353–362. IEEE, 2005.

Gil Regev, Julien Favre, Erich Hayek, Paul Wilson, and Alain Wegmann. Business/IT alignment in practice: Lessons learned from a requirements project at P&G. In *CAiSE Workshops*, pages 93–101. Springer, 2011.

Gil Regev, Biljana Bajic-Bizumic, Arash Golnam, George Popescu, Gorica Tapandjieva, Anshuman Bahadur Saxena, and Alain Wegmann. A philosophical foundation for business and IT alignment in enterprise architecture with the example of SEAM. In *Proceedings of the Third International Symposium on Business Modeling and Software Design (BMSD)*, pages 131–139. SCITEPRESS-Science and Technology Publications, 2013.

Gil Regev, Laura Regev, Yasmina Naïm, Julie Lang, and Alain Wegmann. Teaching an ethnographic approach to requirements elicitation in an enterprise architecture course. In *STPIS@ CAiSE*, pages 5–19, 2015.

Blaize Horner Reich and Izak Benbasat. Factors that influence the social dimension of alignment between business and information technology objectives. *MIS Quarterly*, 24(1):81–113, 2000.

Mark Richters and Martin Gogolla. Validating UML models and OCL constraints. In *International Conference on the Unified Modeling Language*, pages 265–277. Springer, 2000.

Dario Rodighiero and Loup Cellard. Self-Recognition in Data Visualization: how people see themselves in social visualizations. *PubPub*, 2016.

Michael Rosen, Boris Lublinsky, Kevin T Smith, and Marc J Balcer. *Applied SOA: service-oriented architecture and design strategies.* John Wiley & Sons, 2008.

Jeanne W Ross, Peter Weill, and David Robertson. *Enterprise architecture as strategy: Creating a foundation for business execution.* Harvard Business Press, 2006.

Irina Rychkova. *Formal Semantics for Refinement Verification of Enterprise Models.* PhD thesis, EPFL, 2008.

Irina Rychkova, Gil Regev, and Alain Wegmann. Declarative specification and alignment verification of services in ITIL. In *Enterprise Distributed Object Computing Conference Workshops, 2008 12th*, pages 127–134. IEEE, 2008.

Scala. Introduction – Scala documentation, n.d. URL http://docs.scala-lang.org/tour/tour-of-scala.html. Accessed August 2017.

Scala Library. Scala Standard Library 2.12.1 - scala.Predef, n.d. URL http://www.scala-lang.org/api/2.12.x/scala/Predef{\protect\T1\textdollar}.html. Accessed September 2017.

Maung K Sein, Ola Henfridsson, Sandeep Purao, Matti Rossi, and Rikard Lindgren. Action design research. *MIS Quarterly*, 35(1):37–56, 2011.

ServiceNow. ServiceNow, n.d. URL https://www.servicenow.com/. Accessed December 2017.

ServiceNow-Wiki. Using a Next Generation BSM Map, n.d. URL http://wiki.servicenow.com/index.php?title=Using_a_Next_Generation_BSM_Map. Accessed September 2017.

SharePoint. Sharepoint, n.d. URL https://products.office.com/en-us/sharepoint/collaboration. Accessed December 2017.

G Lynn Shostack. Designing services that deliver. *Harvard Business Review*, (1):133–139, 1984.

Herbert A Simon. *The Sciences of the Artificial.* MIT Press, 1996.

Jane B. Singer. Five ws and an h: Digital challenges in newspaper newsrooms and boardrooms. *International Journal on Media Management*, 10(3):122–129, 2008. doi: 10.1080/14241270802262468.

Wendell R Smith. Product differentiation and market segmentation as alternative marketing strategies. *The Journal of Marketing*, pages 3–8, 1956.

SoaML v1.0.1. OMG Service Oriented Architecture Modeling Language, Version 1.0.1. OMG Document Number formal/12-05-10 (http://www.omg.org/spec/SoaML/1.0.1/), 2012.

Ann E Kelley Sobel and Michael R Clarkson. Formal methods application: An empirical tale of software development. *IEEE Transactions on Software Engineering*, 28(3):308–320, 2002.

Ellis Solaiman, Wenzhong Sun, and Carlos Molina-Jimenez. A tool for the automatic verification of BPMN choreographies. In *Services Computing (SCC), 2015 IEEE International Conference on*, pages 728–735. IEEE, 2015.

John F. Sowa and John A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM systems journal*, 31(3):590–616, 1992.

Jim Spohrer, Paul P Maglio, John Bailey, and Daniel Gruhl. Steps toward a science of service systems. *Computer*, 40(1):71–77, 2007.

Bernd Stauss. Internal services: classification and quality management. *International Journal of Service Industry Management*, 6(2):62–78, 1995.

Matthias Steup. Epistemology. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2017 edition, 2017. URL https://plato.stanford.edu/archives/fall2017/entries/epistemology/. Accessed September 2017.

Gerald I Susman and Roger D Evered. An assessment of the scientific merits of action research. *Administrative science quarterly*, pages 582–603, 1978.

SVG-Wikipedia. Scalable vector graphics - wikipedia, n.d. URL https://en.wikipedia.org/wiki/Scalable_Vector_Graphics. Accessed September 2017.

Gorica Tapandjieva. Enterprise bus selection for EPFL central services. Master's thesis, EPFL, 09 2012. Supervisor: Prof. Alain Wegmann.

Gorica Tapandjieva and Alain Wegmann. Specification and implementation of a meta-model for information systems cartography. In *Joint Proceedings of the CAiSE 2014 Forum and CAiSE 2014 Doctoral Consortium co-located with the 26th International Conference on Advanced Information Systems Engineering (CAiSE 2014), Thessaloniki, Greece, June 18-20, 2014.*, volume 1164, pages 113–120. CEUR-WS. org, 2014.

Gorica Tapandjieva and Alain Wegmann. IT service alignment verification of quantitative properties in service design. In *19th IEEE International Enterprise Distributed Object Computing Workshop, EDOC Workshops 2015, Adelaide, Australia, September 21-25, 2015*, pages 111–119. IEEE, 2015.

Gorica Tapandjieva and Alain Wegmann. Ontology for SEAM service models. In *ICEIS 2018, Proceedings of the 20th International Conference on Enterprise Information Systems, Funchal, Madeira – Portugal, March 21-24, 2018*, 2018. Accepted for presentation.

Gorica Tapandjieva, Didier Rey Marchetti, Irina Rychkova, and Alain Wegmann. Towards the definition, implementation and communication of an IT strategy: The case of IT strategy at EPFL. In *Advanced Information Systems Engineering Workshops - CAiSE 2013 International Workshops, Valencia, Spain, June 17-21, 2013. Proceedings*, pages 99–110, 2013. doi: 10.1007/978-3-642-38490-5_8.

Gorica Tapandjieva, Aarthi Gopal, Maude Grossan, and Alain Wegmann. Patterns for value-added services illustrated with SEAM. In *18th IEEE International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, EDOC Workshops 2014, Ulm, Germany, September 1-2, 2014*, 2014.

Gorica Tapandjieva, Gil Regev, and Alain Wegmann. SLA: to sign or not to sign. In *Proceedings of the 2nd International Workshop on Socio-Technical Perspective in IS Development co-located with 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016), Ljubljana, Slovenia, June 14, 2016.*, volume 1604, pages 15–24, 2016.

Gorica Tapandjieva, Giorgio Anastopoulos, Georgios Piskas, and Alain Wegmann. A return on our experience of modeling a service-oriented organization in a service cartography. In *Exploring Services Science - 8th International Conference, IESS 2017, Rome, Italy, May 24-26, 2017, Proceedings*, pages 237–250. Springer, 2017a.

Gorica Tapandjieva, Matteo Filipponi, and Alain Wegmann. Alignment verification in the early stage of service design. In *2017 IEEE 10th International Conference on Service-Oriented Computing and Applications, Kanazawa, Japan, November 22-25, 2017*, 2017b.

Alfred Tarski. The semantic conception of truth: and the foundations of semantics. *Philosophy and phenomenological research*, 4(3):341–376, 1944.

TBM-Council. Tbm council, 2017. URL https://www.tbmcouncil.org/.

TBM Taxonomy v2.0. TBM Taxonomy: Version 2.0. https://www.tbmcouncil.org/learn-tbm/tbm-taxonomy, 2016.

Jorge Teixeira, Lia Patrício, Nuno J Nunes, Leonel Nóbrega, Raymond P Fisk, and Larry Constantine. Customer experience modeling: from customer experience to service design. *Journal of Service Management*, 23(3):362–376, 2012.

Edward R Tufte. *Envisioning information*. Graphics press, Cheshire, Connecticut, 1990.

Edward R Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, Conn, 2001. ISBN 1930824130.

Neil Turner. Introducing the service model canvas, 2015a. URL http://www.uxforthemasses.com/service-model-canvas/. Accessed September 2017.

Neil Turner. The updated service model canvas, 2015b. URL http://www.uxforthemasses.com/updated-service-model-canvas/. Accessed September 2017.

TYM. Trade your mind, n.d. URL http://www.tradeyourmind.com/.

URBA-EA. The Urba-EA club, n.d. URL http://www.urba-ea.com/the-club/. Accessed September 2017.

MH Van Emden. Programming with verification conditions. *IEEE Transactions on Software Engineering*, 5(2):148–159, 1979.

Sandra Vandermerwe and Juan Rada. Servitization of business: adding value by adding services. *European Management Journal*, 6(4):314–324, 1988.

Stephen L Vargo and Robert F Lusch. Evolving to a new dominant logic for marketing. *Journal of marketing*, 68(1):1–17, 2004.

Stephen L Vargo and Robert F Lusch. Service-dominant logic: continuing the evolution. *Journal of the Academy of marketing Science*, 36(1):1–10, 2008.

Ludwig von Bertalanffy. An Outline of General System Theory. *The British Journal for the Philosophy of Science*, 1(2):134–165, 1950. ISSN 0007-0882.

Ludwig von Bertalanffy. *General system theory: foundations, development, applications*. International library of systems theory and philosophy. G. Braziller, 1968.

Rui Wang, Yasset Perez-Riverol, Henning Hermjakob, and Juan Antonio Vizcaíno. Open source libraries and frameworks for biological data visualisation: A guide for developers. *Proteomics*, 15(8):1356–1374, 2015. ISSN 1615-9861. doi: 10.1002/pmic.201400377. URL http://dx.doi.org/10.1002/pmic.201400377.

Alain Wegmann. On the systemic enterprise architecture methodology (SEAM). In *ICEIS 2003, Proceedings of the 5th International Conference on Enterprise Information Systems, Angers, France, April 22-26, 2003*, pages 483–490, 2003.

Alain Wegmann, Anders Kotsalainen, Lionel Matthey, Gil Regev, and Alain Giannattasio. Augmenting the Zachman enterprise architecture framework with a systemic conceptualization. In *12th International IEEE Enterprise Distributed Object Computing Conference, ECOC 2008, 15-19 September 2008, Munich, Germany*, pages 3–13. IEEE Computer Society, 2008. doi: 10.1109/EDOC.2008.49.

Alain Wegmann, Paavo Ritala, Gorica Tapandjieva, and Arash Golnam. Coopetition and ecosystems: case of amazon.com. *The Routledge Companion on Coopetiton Strategies*, 2018. To appear.

Karl E Weick. Cartographic myths in organizations. *Mapping strategic thought*, pages 1–10, 1990.

Gerald M. Weinberg. *An Introduction to General Systems Thinking*. Dorset House Publishing Company, Incorporated, New York, 25 anv edition edition, 2001. ISBN 978-0-932633-49-1.

Stephen White. Using bpmn to model a bpel process. *BPTrends*, 3(3):1–18, 2005.

Alan Wilson, Valarie A Zeithaml, Mary Jo Bitner, and Dwayne D Gremler. *Services marketing: Integrating customer focus across the firm*. McGraw Hill, 2012.

David Woodward and James Brian Harley. The history of cartography. volume 1: Cartography in prehistoric, ancient, and medieval europe and the mediterranean, 1987.

J.A. Zachman. A framework for information systems architecture. *IBM systems journal*, 26(3): 276–292, 1987.

Valarie A Zeithaml, Ananthanarayanan Parasuraman, and Leonard L Berry. Problems and strategies in services marketing. *The Journal of Marketing*, pages 33–46, 1985.

Andreas Zolnowski, Christian Weiß, and Tilo Bohmann. Representing service business models with the service business model canvas–the case of a mobile payment service in the retail industry. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 718–727. IEEE, 2014.

Gorica Tapandjieva

EDUCATION

Sept. 2012 – Apr. 2018	**Ph.D. in Computer and Communication Sciences** *École Polytechnique Fédérale de Lausanne (EPFL)* *Lausanne, Switzerland* *Thesis:* Designing, Aligning, and Visualizing Service Systems *Supervisor:* Prof. Alain Wegmann
Feb. 2011 – Sept. 2012	**M.Sc. in Computer and Communication Sciences** *École Polytechnique Fédérale de Lausanne (EPFL)* *Lausanne, Switzerland* *Thesis:* Enterprise Bus Selection for EPFL Central Services *Supervisor:* Prof. Alain Wegmann
Sept. 2005 – Oct. 2009	**B.Sc. in Informatics and Computer Engineering** *Faculty of Electrical Engineering and Information Technologies (FEEIT)* *Ss.Cyril and Methodius University* *Skopje, Macedonia*

EXPERIENCE

Aug. 2012 – Apr. 2018	**Doctoral assistant** *École Polytechnique Fédérale de Lausanne (EPFL)* *Lausanne, Switzerland* *Teaching assistant:* Enterprise and Service Oriented Architecture, Business Design for IT Services (master level); Analysis I, Physics I, Information–Calculus–Communication (bachelor level). Supervision of semester and master thesis projects, at EPFL and in industry.

Feb. 2012 – Aug. 2012	**Master thesis internship** *Nexell GmbH (formerly e-novate)* *Lausanne, Switzerland*
Nov. 2009 – Jan. 2011	**Software developer** *NeoData (formerly DATA Kompjuterski Inzinering)* *Skopje, Macedonia*
Feb. 2009 – Mar. 2009	**Internship** *Sparkasse bank* *Skopje, Macedonia*

LANGUAGES

Macedonian	**Native language**
English	**Fluent**
French	**Intermediate**
German	**Basic**

PUBLICATIONS

CONFERENCE PAPERS

Gorica Tapandjieva and Alain Wegmann. “Ontology for SEAM service models”. In: *ICEIS 2018, Proceedings of the 20th International Conference on Enterprise Information Systems, Funchal, Madeira – Portugal, March 21-24, 2018*. 2018.

Gorica Tapandjieva, Matteo Filipponi, and Alain Wegmann. “Alignment verification in the early stage of service design”. In: *2017 IEEE 10th International Conference on Service-Oriented Computing and Applications, Kanazawa, Japan, November 22-25, 2017*. 2017.

Gorica Tapandjieva, Giorgio Anastopoulos, Georgios Piskas, and Alain Wegmann. “A return on our experience of modeling a service-oriented organization in a service cartography”. In: *Exploring Services Science - 8th International Conference, IESS 2017, Rome, Italy, May 24-26, 2017, Proceedings*. Springer. 2017, pp. 237–250. DOI: 10.1007/978-3-319-56925-3_19.

Gorica Tapandjieva and Alain Wegmann. “Specification and implementation of a meta-model for information systems cartography”. In: *Joint Proceedings of the CAiSE 2014 Forum and CAiSE 2014 Doctoral Consortium co-located with the 26th International Conference on Advanced Information Systems Engineering (CAiSE 2014), Thessaloniki, Greece, June 18-20, 2014*. Vol. 1164. CEUR-WS. org. 2014, pp. 113–120. URL: http://ceur-ws.org/Vol-1164/PaperVision15.pdf.

George Popescu, **Gorica Tapandjieva**, and Alain Wegmann. “Business and IT design with SEAM: an illustration with the phd hiring process at Ecole polytechnique Fédérale de Lausanne”. In: *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE. 2013, pp. 1938–1943. DOI: 10.1109/SMC.2013.333.

Gil Regev, Biljana Bajic-Bizumic, Arash Golnam, George Popescu, **Gorica Tapandjieva**, Anshuman Bahadur Saxena, and Alain Wegmann. “A philosophical foundation for business and IT alignment in enterprise architecture with the example of SEAM”. In: *Proceedings of the Third International Symposium on Business Modeling and Software Design (BMSD)*. SCITEPRESS-Science and Technology Publications. 2013, pp. 131–139.

Workshop papers

Gorica Tapandjieva, Gil Regev, and Alain Wegmann. “SLA: To sign or not to sign”. In: *Proceedings of the 2nd International Workshop on Socio-Technical Perspective in IS Development co-located with 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016), Ljubljana, Slovenia, June 14, 2016*. Vol. 1604. 2016, pp. 15–24. URL: http://ceur-ws.org/Vol-1604/Paper2.pdf.

Gorica Tapandjieva and Alain Wegmann. “IT service alignment verification of quantitative properties in service design”. In: *19th IEEE International Enterprise Distributed Object Computing Workshop, EDOC Workshops 2015, Adelaide, Australia, September 21-25, 2015*. IEEE. 2015, pp. 111–119. DOI: 10.1109/EDOCW.2015.36.

Gorica Tapandjieva, Aarthi Gopal, Maude Grossan, and Alain Wegmann. “Patterns for value-added services illustrated with SEAM”. In: *18th IEEE International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, EDOC Workshops 2014, Ulm, Germany, September 1-2, 2014*. 2014. DOI: 10.1109/EDOCW.2014.56.

Gorica Tapandjieva, Didier Rey Marchetti, Irina Rychkova, and Alain Wegmann. “Towards the definition, implementation and communication of an IT strategy: The case of IT strategy at EPFL”. In: *Advanced Information Systems Engineering Workshops: CAiSE 2013 International Workshops, Valencia, Spain, June 17-21, 2013. Proceedings*. Ed. by Xavier Franch and Pnina Soffer. Berlin, Heidelberg: Springer, 2013, pp. 99–110. ISBN: 978-3-642-38490-5. DOI: 10.1007/978-3-642-38490-5_8.

Book chapters

Alain Wegmann, Paavo Ritala, **Gorica Tapandjieva**, and Arash Golnam. “Coopetition and ecosystems: Case of amazon.com”. In: *The Routledge Companion on Coopetiton Strategies* (2018). To appear.