FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Masters' Degree in Informatics Engineering
Dissertation

# LaNDLESS

## Integrating Linked Data with Linked Services

July 2, 2013

Ricardo Lopes
rplopes@student.dei.uc.pt

Advisor
Jorge Cardoso

## Abstract

Services have grown to become the strongest and most influential economic sector of our society. Paradoxically, they are also the least studied and least understood. It is thus of utmost importance to develop scientific research towards better understanding of services.

Some efforts have been done in the past to conceive models for studying services. However, those models usually lack the completeness we need for a full study. Moreover, because those models are typically from an economic or managerial background, they are not designed with data computation in mind and thus fail to provide the necessary foundations for a deeper study and automated reasoning.

The Unified Service Description Language (USDL) tries to fill that gap by providing a computer language for describing services. However, due to its complexity and lack of extensibility, its adoption is still not impressive. Linked USDL tries to overcome those flaws by providing a simpler model and by using the Semantic Web technologies and Linked Data. It is, however, a model for describing services in a customer-oriented view. If we seek to obtain a better understanding of services, we must focus on the full length of a service, specially in internal views such as managerial and operational.

In order to meet those requirements and thus advance science's understanding of services we propose the model Linked Service Systems for USDL (LSS-USDL). This model builds upon the previous service model approaches and on the USDL research efforts to enable a complete description of services in a machine-readable notation. We also propose a tool set to demonstrate the applicability and usefulness of this model.



**Keywords:** Service, Service system, Service model, Business model, Service description, Semantic Web, Linked Data, Linked Services, RDF, USDL, Linked USDL.

# Acknowledgements

One thing I learned while working in this thesis is that scientific research is hard. It requires method, perseverance and a sharp focus to stay on the right path. Therefore, this has been a challenging but very enriching experience for me. Moreover, I am grateful for the help of many people who have contributed to some extent to the outcome of this research work.

Firstly, I want to thank my advisor Jorge Cardoso for all the guidance throughout the year, for the feedback to my work artifacts and also for providing all means to conduct a strong research effort, such as showing drafts of unpublished work of the USDL research and arranging meetings with other researchers.

Furthermore, I would also like to thank the rest of the Genssiz research group for all the insightful discussions and the useful feedback provided.

In addition, I want to thank my parents Isabel Pinto and Sérgio Lopes for providing me everything I needed throughout the years so that I could do this thesis, and my girlfriend Tânia Vargas for her patience and understanding and all her support.

Lastly, I would also like to thank my Master's degree colleagues for the companionship and shared knowledge, my friends and all other people that had any direct or indirect impact in the outcome of this thesis.

Ricardo Lopes

# Contents

# List of Tables

# List of Figures

# List of Acronyms

**BPMN** Business Process Modeling Language

**IEEE** Institute of Electrical and Electronics Engineers

**ITIL** Information Technology Infrastructure Library

**IoS** Internet of Services

**LDC** Linked Data Cloud

**LOS** Linked Open Services

**MVC** Model-View-Controller

**OWL** Ontology Web Language

**OWL-S** Semantic Markup for Web Language

**RDF** Resource Description Framework

**SaaS** Software as a Service

**SKOS** Simple Knowledge Organization System

**SWSF** Semantic Web Services Framework

**UML** Unified Modeling Language

**USDL** Unified Service Description Language

**W3C** World Wide Web Consortium

**WSDL** Web Services Description Language

**WSMO** Web Service Modeling Ontology

# 1 Introduction

This chapter explains the background and motivation of this thesis and describes its goals, challenges and approaches.

The first section explains the background of this thesis, explaining the relevance of the field and describing its core concepts. The second section discusses the motivation to develop the proposed solution and describes that solution. The third section presents a list of goals and challenges that must be addressed. The fourth section proposes a set of approaches to solve the problem. Finally, the fifth section briefly explains the structure of the document.

## 1.1 Background

We live in a society of services. Although the service sector was initially considered a residual economic category next to the agriculture and manufacturing sectors [38], in the last three decades it has grown to become the largest part of most industrialized economies [85], now contributing to around 70% of the total GDP[1] of Western Europe's economies [56]. As of 2006, Germany has experienced a service sector growth of 44% in 25 years, Russia experienced 38%, Japan experienced 40% and China reached a growth of 191% [57]. It is, then, by far the strongest economic growth driver in the world [8].

However, despite the soaring importance of this economic sector to society, scientific understanding of modern services is still rudimentary [21]. Moreover, there is still no widely accepted definition of service [85], and its meanings might differ and generate inconsistencies not only across disciplines, but also within them [32].

In order to better understand the scope of this thesis, it is then necessary to disambiguate the meaning of service and formalize a definition that we can use from now on.

[1] Gross Domestic Product

The term "service" was first used in the 1930s by the U.S. Department of Commerce's Standard Industrial Classification to denote the residual economic activities that did not fit into agriculture and manufacturing [21]. Services were called "unproductive labor" because their labor did not result in physical goods, contrasting with agriculture and manufacturing, which resulted in the production of physical goods and were therefore called "productive labor" [57]. However, due to the service sector's growth, we shifted from that Goods-Dominant (GD) logic to a Service-Dominant (SD) logic, focusing on the creation of value rather than the exchange of goods [92].

According to the ITIL[2] library, "a 'service' is a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks" [71]. The W3C[3] defines service as "... an abstract resource that represents capability of performing tasks that form a coherent functionality from the point of view of provider entities and requester entities. To be used, a service must be realized by a concrete provider agent" [89]. Hill states that "a service may be defined as a change in the condition of a person, or a good belonging to some economic unit, with the prior agreement of the former person or economic unit" [43]. Based on the previous definitions and on more opinions from different authors [5][31][32][58][85], we can define service as follows:

**Definition 1.** *A service is a previously agreed exchange of competences and knowledge between a provider and a customer in order to provide value to both parties.*

When studying services, we are faced with other keywords that need clarification to develop a better understanding of this area of study. In order to do so, we need to formulate unambiguous definitions about the most relevant concepts, namely "service system", "service model", "service architecture", "business model" and "service description".

Service system is described in the literature as "... a system comprised of a facilitator and appraiser systems for generating value through the provision and consumption of services" [65], "... complex adaptive systems made up of people, (...) dynamic and open, rather than simple and optimized" [85], among other definitions [5][26][37][58][66]. Based on those descriptions, we can reach the following definition of service system:

**Definition 2.** *A service system is a collection of resources, stakeholders, processes and other service assets that, combined, enable value co-creation between producer and consumer. It can thus be seen as a schema for services.*

[2]Information Technology Infrastructure Library: `http://www.itil-officialsite.com`
[3]World Wide Web Consortium: `http://www.w3.org`

Models "... help us by letting us work at a higher level of abstraction (...) by hiding or masking details, bringing out the big picture, or by focusing on different aspects of the prototype" [72]. Their essence is abstraction: "... the removal of fickle and distracting detail of implementation technologies as well as the use of concepts that allow more direct expression of phenomena in the problem domain. (...) the only effective means that we have of dealing with complexity that overwhelms our cognitive capacities" [80]. Crossing these statements with others in the literature [45][82] and with the previously derived service definition, we get the following description of service model:

**Definition 3.** *A service model is an abstraction of a service system that highlights and categorizes the service's core structure and interactions, hiding its complex nature from who does not need to know it.*

The term architecture is defined by IEEE[4] 1471[5] as "... the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution" [60]. Zachman, while explaining his framework for enterprise architecture, defines architecture as "... that set of design artifacts, or descriptive representations, that are relevant for describing an object such that it can be produced to requirements (quality) as well as maintained over the period of its useful life (change)" [90]. Cardoso et al. states that service architectures "... look into the organization of software-based services, how they are connected, and what service information is exchanged between consumers and providers" [18]. Building on top of the our service definition and these architecture definitions (including Kruchten's contribution [53]), we can define service architecture as:

**Definition 4.** *Service architecture is the set of rules and guidelines for the components, relationships and interfaces of the structural elements of a software-based service that guides the organization of that service.*

Business model is defined by Timmers as "... an architecture for the product, service and information flows, including a description of the various business actors and their roles; and a description of the potential benefits for the various business actors; and a description of the sources of revenues" [86]. In their book about the Business Model Canvas, Osterwalder and Pigneur state that "a business model describes the rationale of how an organization creates, delivers, and captures value" [74]. With the help of these descriptions and more found in the literature [3][28][59][75], we reach the following definition:

**Definition 5.** *A business model is a conceptual representation of the business of a certain organization, intended to describe its stakeholders, interactions and*

[4]Institute of Electrical and Electronics Engineers

[5]`http://www.iso-architecture.org/ieee-1471/index.html`

*value propositions and explain how the organization meets customer goals and how it makes profit.*

A WSDL[6] service description "... indicates how potential clients are intended to interact with the described service. It represents an assertion that the described service fully implements and conforms to what the WSDL 2.0 document describes" [22]. Oberle et al. argue that "Information systems such as a service marketplace will manage descriptions of a service and not the service itself. The service itself is an event (...) that can be executed arbitrary times used by different consumers" [70]. Cardoso et al. state that service descriptions "... bring various ways to describe services' interfaces using schema, models and semantics" [18]. Hence, we can define service description as follows:

**Definition 6.** *A service description is a static descriptive representation of a service system, used to educate the different stakeholders about its properties and interactions, that represents the service as a whole, and not specific instances or occurrences.*

Figure 1.1 builds on the concepts defined above to explain the scope of a service model like what we intend to achieve.

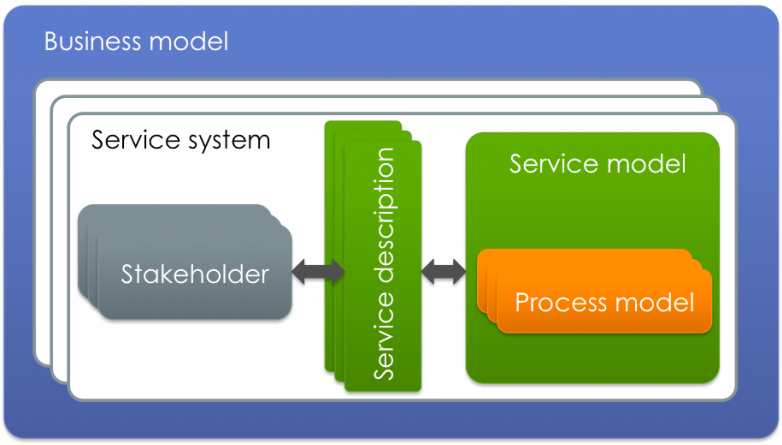


Figure 1.1: Diagram explaining the context of business models, service systems, service models, process models and service descriptions

If we look back at the definitions of business model and service model, we see that they are different models that serve different goals. While business models have a greater focus on a company strategy and long-term goals, service models put a bigger focus on the company's operations and their goals and outcomes. This means that, hierarchically, business models provide a higher

[6]Web Services Description Language

level of abstraction. A business model may contain one or more service models. Following this reasoning, we may also identify process models (such as BPMN[7], Petri Nets, UML[8] State Diagrams and so forth) as models focusing on enabling a company's process operations, thus acting at a lower level of abstraction than service models. A service model may contain one or more process models. This hierarchy is resumed in Figure 1.2.

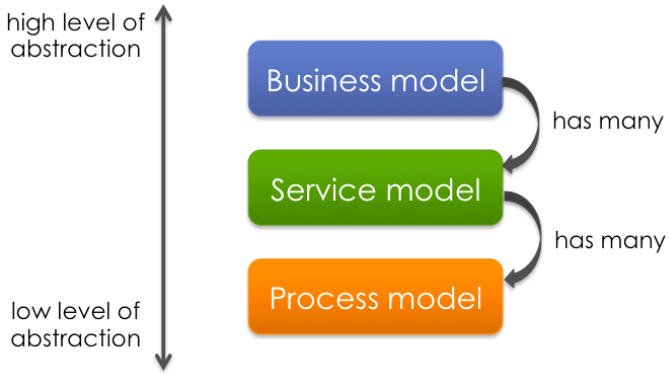


Figure 1.2: Business models, service models and process models compared by level of abstraction

Both business models and process models have been intensively studied in the past. Service models, however, only recently have began to draw attention of researchers. As seen in Figure 1.2, a deeper study of service models is thus very relevant in order to fill the gap of conceptual representation between the long-term strategic vision of business models and the short-term operational vision of process models.

The definitions we discussed help us understand this thesis's context and raise some interesting questions and conclusions that are further explored in the next sections.

## 1.2 Motivation

As we have seen in the previous section, services have become the most influential economic sector of our society. Yet, few studies are found in the literature that present service models or frameworks for better organization management. We may find services everywhere but, nevertheless, most of them are not properly documented or modeled.

[7]Business Process Modeling Notation
[8]Unified Modeling Language

One of the original goals of USDL[9] was to address this issue and provide a service description language for managers to formalize their organization's services in a standardized format. However, USDL is limited to the description of the service's customer interactions, so a complete service system description is still not possible. In other words, USDL treats a service as a blackbox where internal details are not known.

A model that could formalize and describe the whole service system operations would thus facilitate managers' transition from the currently chaotic service management to a formalized one. This would result in many benefits, such as:

- **Documentation:** Service systems would be modeled in a machine-readable language, which would make it possible to generate service descriptions for people to analyze and better understand it. This documentation generation facilitates staff formation, customers knowledge about the service offerings and so forth.

- **Transparency:** A whitebox service system description would let stakeholders better understand what is happening. This would prove organizations' credibility and sustainability and would let foundations and governments clarify where is people's money being spent and justify potential decisions.

- **Bottlenecks and fail points identification:** Only after modeling a complete service system it is possible to get a big picture view while also being able to look at all the details. If a service is well described, it should be possible to identify potential drawbacks such as bottlenecks and fail points and study solutions to overcome it.

- **Automation:** If a service system is fully modeled in a machine-readable language, then all requirements are met for an automation tool to read that model and dispatch worker processes for certain inputs.

- **Simulation:** Since we can model the full length of a service system in a machine-readable language, it is possible to conduct service system behavior simulations to aid managerial and operational decisions. Such simulations could be executed using the principles of System Dynamics [6].

- **Integration:** USDL may already have some of the aforementioned benefits, but lacks good integration of services with data and external services. Linked USDL[10] tries to solve that problem by using Semantic Web technologies and integration with Linked Data[11], but it cannot fully address that problem because it still describes the service system as a blackbox.

[9] Unified Service Description Language (USDL

[10] `http://www.linked-usdl.org`

[11] `http://linkeddata.org`

A whitebox service model, however, would be able to fully integrate with other data and services because it can provide the description of all its components.

- **Discovery:** One of the goals of the Internet of Services (IoS)[12] and USDL is to be able to describe services in such a standardized way that it might be possible to browse them in a generic online services marketplace. However, USDL can only provide a single service description, since it only models customer interactions. In other words, one service system generates only one service description, and thus it is not possible to generate a view or description for a manager with specific skills and another for an engineer with a different set of skills and interests. A complete service system model would be able to generate dynamic service descriptions depending on what is relevant to display. Those service descriptions could then be aggregated in service marketplaces for easier discovery and comparisons [17].

Finally, a whitebox service system modeling tool could be able to generate different service descriptions based on who is accessing it and what goals is the description trying to address at that moment. As such, customers could interact with a service description of customer interactions (such as USDL), staff could interact with a service description for internal operations and so forth. Those would be different views of the same service system. Such an achievement might prove to be a very important advancement in the field, potentially putting previous service models that generate a single service description to obsolesce.

## 1.3 Objectives and challenges

Since existing models that describe service systems typically follow a blackbox approach, in this thesis we take the challenge of defining a model to describe a service system using a whitebox approach (i.e. also providing an internal view). In order to do so, we discuss the specification, creation and validation of such a model and its associated toolset. Such toolset should be designed using Semantic Web technologies, since they facilitate interoperability, which is one of the service information challenges pointed out by Oberle et al. [70].

In addition to the Semantic Web technologies, it should also include Linked Data principles and practices. This lets us reuse existing LDC[13] vocabularies (thus maximizing compatibility and reusability and minimizing engineering efforts), making use of the recent trend towards organizations and governments

[12] `http://internet-of-services.com`
[13] Linked Data Cloud

publishing data on the web [83], and facilitates the automatic computer processing of data [18]. The use of these technologies enables the open exposure of service features, resulting in LOS[14] [68].

The objectives of this project are thus presented below, along with considerations about their main challenges.

1. **Define a service system model:** The main objective with this work is to define a machine-readable model that describes general service systems. The biggest challenge is the research area of that goal, services science [21], that requires knowledge on disciplines such as economics, law, marketing, management sciences, industrial and systems engineering, computer science, information systems and so on [84]. Another challenge is the use of Semantic Web technologies and Linked Data in order to facilitate computer reasoning and data interoperability.

2. **Develop a graphical modeling tool:** In order for this model to be accepted by managers and other non-technical service system modelers it must present an interface they can understand. Hence, one of the goals is to develop a tool that hides technical details and is easy for that user group to use and understand. This creates the challenge of trying to hide as much complexity as possible while still making full use of all its capabilities. It also requires a basic understanding about what is cognitively difficult for that user group and what metaphors may be used to make the tool easier to use. Such a goal also requires the need to convert the data modeled with Semantic Web technologies to the data model of the tool and vice-versa.

3. **Align hte model with Linked USDL:** One of the main advantages of the proposed model over similar approaches such as USDL is the possibility of describing full service systems and not only what is visible to customers. As such, it should be possible to generate different views depending on the desired scope. Since Linked USDL may be considered as a view for customers, then allowing the automatic exportation to that language proves the concept of views generation and makes this model a valuable addition to Linked USDL's user base. Another valuable feature we get from this alignment is the possibility of extracting information from Linked USDL service systems to automatically build a service model. The biggest challenges this goal creates are the non-trivial process of aligning the elements of two different logical models, exporting and importing data from one to another and also the lack of examples of service descriptions in Linked USDL, which complicates the study of that tool.

[14] Linked Open Services

## 1.4 Approach

The first step that needs to be one in order to implement this model is to conduct an extensive research to be able to identify the most common service model attributes found in the literature. During such a research it is necessary to identify previous approaches for service system modeling and the attributes they use to describe a service system. With that information we can compare and contrast them in order to identify the most common and useful attributes used in the studied models. By identifying those attributes we have the necessary foundations to start developing a new service model.

After designing the intended model, it is necessary to describe it using Semantic Web technologies and to follow the principles of the Linked Data. This means building the model with RDF[15], because that is the standard for Linked Data [13] and reusing existing vocabularies found in the Linked Data Cloud.

The completion of this step ensures that we have a working ontology to describe internal and external elements of services, which we may call Linked Services, as they are described as Linked Data [76]. Because of that and because our goal, inserted in the USDL research efforts, is modeling service systems, the model will be called Linked Service Systems for USDL (LSS-USDL).

An initial evaluation of the model will be done by using it to describe a set of service systems. The model should be considered acceptable and relevant if the aforementioned services can be successfully described and no important information is missing. Some minor model improvements might arise during this empirical study.

After the successful development and evaluation of our service model there are new tasks to complete in order to accomplish our goals.

The first task is to implement an abstraction layer on top of the service system model in order to allow better interaction with it. This will be done using standard software development tools, focusing on rapid prototyping frameworks to reduce development time. The resulting application should be able to generate new RDF for service descriptions, load existing service descriptions in RDF files and edit the service systems' information in a graphical notation. The evaluation of this task may be performed by viewing and modifying the previously described services and by modeling new services and checking their RDF code correctness.

Another task is to implement a tool that generates new specific service descriptions based on the complete service system model and a new service system model based on an existing service description. The main challenge of this task is doing such conversions to and from Linked USDL, as discussed in the previous section. In order to overcome this challenge, a study of Linked USDL

[15]Resource Description Framework

must be performed and enough reasoning must be applied for elements that might not have a direct match. The evaluation of this task may be performed by exporting existing service descriptions to Linked USDL and checking the correctness of the results, and also by converting Linked USDL descriptions into our model's notation.

Finally, these two tools should be merged in a single prototype application. That application should be open sourced to allow future improvements, have good documentation and be deployed for demonstration purposes.

## 1.5 Document structure

This document is structured as follows: the first chapter introduces the problem and the proposed solution. The second chapter studies the state of the art and proposes a framework to evaluate service descriptions based on the identified common attributes. The third chapter presents the service system model building blocks, the design approach and the resulting model. The fourth chapter presents two software prototypes that intend to demonstrate and improve the model's usage. The fifth chapter reveals the evaluation process of the model with three different use cases. Lastly, the sixth chapter draws our conclusions and discusses possible future work.

This document has two appendices: the first one is the schedule of the work for the second semester and how its tasks were performed, and the second one is the documentation found in the two code repositories created in the scope of this thesis.

# 2
# State-of-the-art

This chapter presents and discusses several service and business models found in the literature, their strengths and weaknesses and how can we compare and contrast them to obtain a set of common attributes that describe a service. It also presents some related work in the field of service description tools.

The first section briefly introduces the studied service models and proposes a framework of the most common attributes between them. The second section explains each of the identified frequent attributes and relates them to their corresponding service models. The third section describes each of the aforementioned service models in greater detail. Finally, the fourth section presents current service description tools.

## 2.1 Service models overview

To identify the most frequent attributes of service models, an extensive literature review was conducted on existing approaches. From this study, 8 models were chosen for a deeper analysis. Some of those models are classified by their authors as business models but, as a business model represents a set of services, what we get in fact is a different view of the same problem, more focused on business aspects. Because of the importance of mixing different perspectives to get richer results, we consider those approaches to our state of the art study.

These 8 models were then categorized based on their nature, in order to better understand how closely related they are to our goals and how can they contribute to them. In this analysis, we tested if the models referred to an internal or an external service description, how formal its notation it is and their classification as a service description tool.

According to Ferrario et al., service description efforts may be categorized into six different groups [32]:

1. **Service-Oriented Architectures (SOA):** Focuses on IT aspects of ser-

vices and is used mainly to exchange service information over the web.

2. **Semantic Web Services:** Based on SOA but focused on the automation of discovery, composition and invocation of services using ontology reasoners and planning algorithms.

3. **Software-as-a-Service (SaaS):** Similar to SOA, but focused on value delivery to the customer, rather than generic service information exchange between two or more parties.

4. **Purely economic:** Usually not specified in a computer readable way, is used to describe the economic aspects of a service regardless of its nature.

5. **Service networks:** Also focused on economic aspects, but emphasizing the ecosystem and value chain relationships between services of economic value.

6. **Big picture of service systems:** Focuses on the bigger picture of service systems or service science, including assets such as co-creation and knowledge.

Using these six service description groups for categorization, it is now possible to complete the proposed analysis. Table 2.1 presents the classification of the studied service models according to the aforementioned parameters. It also includes the desired classification of the service model that we need for the work proposed in this thesis, so as to compare our goals with the state of the art.

| | Scope | Formality | Category |
|---|---|---|---|
| Alt and Zimmermann (2001) | I | I | 4 |
| Petrovic et al. (2001) | I | F | 4 |
| Kaner and Karni (2007) | I | I | 6 |
| Kinderen and Gordijn (2008) | E | C | 5 |
| Spohrer and Maglio (2009) | I | F | 6 |
| Osterwalder and Pigneur (2010) | I | I | 4 |
| Fielt (2010) | I | I | 6 |
| Zolnowski et al. (2011) | I | I | 6 |
| Desired model | I | C | 6 |

Table 2.1: Service models classification. Scope: I = Internal, E = External. Formality: I = Informal, F = Formal, C = Computer readable. Category: [1-6] = same category as in the explanatory numbered list.

As we can see, no current approach matches this simple set of requirements of the desired model. This means that there is currently no model that we

can adopt and implement to satisfy our goals. Thus, there is a great need to develop a new service description model that fully satisfies our needs.

In order to develop such a model, a new study on the selected approaches must be conducted, such that we can understand common patterns and use them as a framework to find the attributes needed for a complete service description.

Comparing these models, it was possible to identify common patterns and thus derive a framework of the most frequent attributes that are used to describe a service. Those attributes are `Goals`, `Stakeholders`, `Processes`, `Inputs`, `Outputs`, `Resources`, `Measures`, `Legal` and `Financial`.

Table 2.2 shows the contributions of each service model to the framework of identified common attributes.

| | *Goals* | *Stakeholders* | *Processes* | *Inputs* | *Outputs* | *Resources* | *Measures* | *Legal* | *Financial* |
|---|---|---|---|---|---|---|---|---|---|
| Alt and Zimmermann (2001) | ■ | ■ | ■ | | | □ | | ■ | □ |
| Petrovic et al. (2001) | ■ | | ■ | | | ■ | | | ■ |
| Kaner and Karni (2007) | ■ | □ | ■ | ■ | ■ | ■ | □ | □ | □ |
| Kinderen and Gordijn (2008) | ■ | ■ | □ | | ■ | | | | □ |
| Spohrer and Maglio (2009) | ■ | ■ | ■ | ■ | ■ | ■ | ■ | □ | |
| Osterwalder and Pigneur (2010) | □ | ■ | ■ | | | ■ | | | ■ |
| Fielt (2010) | □ | ■ | ■ | ■ | | ■ | | | ■ |
| Zolnowski et al. (2011) | □ | ■ | ■ | | | ■ | | | ■ |

Table 2.2: Service models attributes categorization. □= moderate contribution, ■= important contribution.

In the next section we discuss each of these attributes in detail. The models are described in a later section, where individual tables show which elements of each model contributed to each identified framework attribute.

## 2.2 Framework attributes

This section describes the 9 attributes of the framework we derived from the literature review for modeling service systems.

### 2.2.1 Goals

Service or business goals are among one of the most used attributes in the studied models. Goals are also called `Mission`, `Value proposition`, among

other naming options.

There is no doubt that this is a crucial element for a service description model, not only because of its wide acceptance among the studied approaches, but also because it states the objectives of the service and its value proposition to the consumers.

### 2.2.2 Stakeholders

Stakeholders are one of the most important attributes of a service, as the whole service operation is conditioned by the people and organizations involved. This attribute is used by almost all the studied approaches, due to its importance.

When studying how the service models describe the stakeholders, it is possible to notice some frequent attributes in use. In most service models, there is an attribute for the service customers. In the Business Model Canvas and the two studied improved approaches there is also an attribute to describe the service partners [34][74][92]. Spohrer and Maglio propose the description of other parties, namely authorities and competitors [84].

Stakeholders are, therefore, an important yet complex element to describe. Thus, when developing a service description model, it is necessary to use caution and select the most adequate level of complexity.

### 2.2.3 Processes

Processes are, along with goals, a service attribute that all service description approaches studied included in their models. Processes are also called `Key activities`, `Elementary service`, among other naming options.

This attribute is of utmost importance when describing services for internal organization, because corporations must have a strong knowledge of the processes needed for any of their services, in order to identify bottlenecks and other flaws that can be corrected. A good process description tool is, then, an organizational advantage, and thus should be taken into consideration when creating a new service description model.

We also consider sub-processes such as customer relations, marketing and so forth to be included in this category.

### 2.2.4 Inputs

Service inputs are described in a small set of service models. Spohrer and Maglio refer to them using the concept of `Ecology` [84]. Fielt, when describing a model that improves on Osterwalder's Business Model Canvas, introduces `Partner activities` and `Customer activities`, which are a set of activities that act as an input for the service [34].

However, the best description of service inputs lies within Karni and Kaner's CAIOPHYKE model, where we can describe main classes of the major class `Inputs`, such as `Physical`, `Information` or `Constraints` [49].

### 2.2.5 Outputs

Service outputs, like service inputs, are described in a small set of service models. Spohrer and Maglio refer to them using the concept of `Outcomes`, where they reference game theory to propose a desired outcome of win-win among the stakeholders [84]. In the e$^3$service we can see service outputs description in the classes `Consequence`, `Benefit` and `Value derivation` [51].

As with service inputs, the best description of service outputs lies within Karni and Kaner's approach, where we can describe main classes of the major class `Outputs`, such as `Knowledge`, `Waste` or `Money` [49].

### 2.2.6 Resources

Resources are described in all internal service description models, and thus is also an important attribute to consider. Alt and Zimmermann's approach is the only internal service description model that does just a partial description of resources, as it only considers technology [4].

Kaner and Karni classify resources as `Human enablers`, `Physical enablers` and `Information enablers` [47]. Osterwalder and Pigneur classify their model's `Key resources` attribute as `Physical`, `Financial`, `Intellectual` and `Human` [74]. Fielt further expands this resources description to include `Partner resources` and `Customer resources` [34].

### 2.2.7 Measures

This attribute refers to how the company can measure the service's performance, in order to get feedback of their operations. Surprisingly, only a very small number of models was found in the literature that addressed this attribute.

Spohrer and Maglio classify measures as quality measures (determined by the customer), productivity measures (determined by the provider), compliance measures (determined by the authority) and sustainable innovation measures (determined by competitors) [84]. Kaner and Karni use the main class `Performance measures` in the major class `Information enablers` [47], which shows that this attribute is not regarded as a core pillar of their model.

### 2.2.8 Legal

The legal aspects of a service or business have a surprisingly low presence on the literature. Alt and Zimmermann propose `Legal issues` as one of their six generic elements of a business model [4]. Karni and Kaner use the main class `Legal factors` in the major class `Environment`, dividing it with the minor classes `Constraints on customer`, `Constraints on employees`, `Constraints on location`, `Constraints on service fees` and `Standards` [49]. Finally, Spohrer and Maglio identify `Governance mechanism based interactions` (which they define as an interaction often invoked by authorities, which usually takes place when one stakeholder interacts in non-normative ways) and `Access rights` (the access rights of any given resource) [84].

### 2.2.9 Financial

The financial factors of a service or business are also a very important attribute to define. These factors are considered in almost all of the studied approaches, although not all of them present a comprehensive description.

Osterwalder and Pigneur's model and the two models that were built on top of it use the elements `Cost structure` and `Revenue streams` to describe this attribute [34][74][92]. However, Fielt further explores this attribute and adds two more elements: `Partner cost structure` and `Customer cost structure` [34]. Alt and Zimmermann only refer to revenues [4], while Petrovic et al. present not only a `Revenue model` but also a `Capital model` [77]. Kaner and Karni have no major class for this attribute, but explore it in many main classes: `Financial factors` (from both the major classes `Inputs` and `Outputs`), `Economic factors` (from the major class `Environment`) and `Service payment` (from the major class `Processes`) [47]. Finally, Kinderen and Gordijn refer to financial and economic factors in their element `Sacrifices`, which describes what must be given in order to benefit from a provider's service [50]. This approach, however, does not cover the full scope of this attribute, as it is just about payments, and may even miss its scope when that payment is not monetary.

## 2.3 Service model approaches

### 2.3.1 Six generic elements of business models

Alt and Zimmermann distinguish six generic elements of a business model: `Mission`, `Structure`, `Processes`, `Revenues`, `Legal issues` and `Technology` [4]. The first four elements are the core elements, while the last

two represent requirements and constraints. The authors propose those six generic elements as a comprehensive framework to develop sustainable business models [4]. Figure 2.1 shows the proposed model.

| Generic attributes | Model attributes |
|---|---|
| Goals | Mission |
| Stakeholders | Structure |
| Processes | Processes |
| Inputs | - |
| Outputs | - |
| Resources | Technology |
| Measures | - |
| Legal | Legal issues |
| Financial | Revenues |

Table 2.3: Attribute analysis of the six generic elements of business models [4]

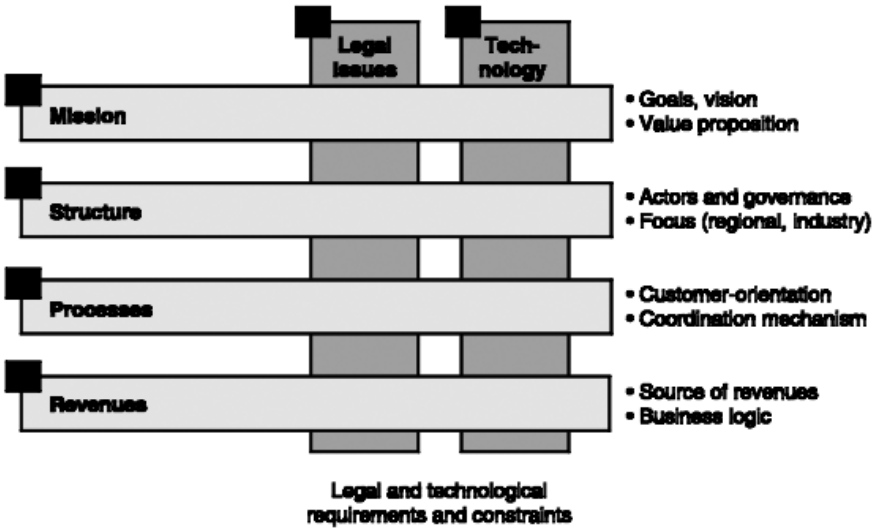


Figure 2.1: Six generic elements of business models [4]

This is the earliest proposed model of this state of the art study. However, as we can see by looking at Table 2.2, it already mentions most of the generic attributes that newer models used the most, which indicates that it had impact in the field. Furthermore, Petrovic et al. make a reference to this model to check as a validation of their proposed model [77].

Despite the presence of the majority of the most used attributes, when looking at the generic attribute `Resources`, it is clear that this model does

not fully describe it, as it only proposes the attribute `Technology`, which is but a segment of a resources description.

### 2.3.2 Business model seven sub-models

Petrovic, Kittl and Teksten divide a business model into seven sub-models in order to describe the logic behind business processes. Those are the `Value model`, the `Resource model`, the `Production model`, the `Customer relations model` (which is further divided into `Distribution model`, `Marketing model` and `Service model`), the `Revenue model`, the `Capital model` and the `Market model` [77].

| Generic attributes | Model attributes |
|---|---|
| Goals | Value model |
| Stakeholders | - |
| Processes | Production model, Customer relations model |
| Inputs | - |
| Outputs | - |
| Resources | Resource model |
| Measures | - |
| Legal | - |
| Financial | Revenue model, Capital model |

Table 2.4: Attribute analysis of the Business Model seven sub-models [77]

The naming of this model's elements hints at a lower level description for each of them. However, the authors do not identify any further specifications. When looking at Table 2.2, it is interesting to notice how this model's contributions to the list of generic attributes is similar to the contribution of Osterwalder's work, despite the years of difference.

The biggest limitation of this approach is the lack of descriptions of each of the seven proposed sub-models, which makes it impossible to fully grasp the concepts proposed by the authors. Another limitation is the lack of a description of stakeholders.

### 2.3.3 CAIOPHYKE

Kaner and Karni propose a service model in [49], extending it and naming it CAIOPHYKE in [47]. This model describes a service according to 9 major classes: `Customers`, `Goals`, `Inputs`, `Outputs`, `Processes`, `Human`

`enablers`, `Physical enablers`, `Information enablers` and `Environment`. Each of these major classes can be further described by their respective main classes, such as `Service provision` or `Call center` (main classes for the major class `Processes`). Each of these main classes can then be further described by their respective minor classes, such as `Call center service` or `Call routing` (minor classes for the main class `Call center`) [47].

| Generic attributes | Model attributes |
|---:|---|
| Goals | Goals |
| Stakeholders | Customers |
| Processes | Processes |
| Inputs | Inputs |
| Outputs | Outputs |
| Resources | Human enablers, Physical enablers, Information enablers |
| Measures | Performance measures (from Information enablers) |
| Legal | Legal factors (from Environment) |
| Financial | Financial factors (from Inputs and Outputs), Economic factors (from Environment), Service Payment (from Processes) |

Table 2.5: Attribute analysis of CAIOPHYKE [49]

This model was developed based on a study with 150 student projects that covered around 100 service domains [48]. Figure 2.2 shows a graphical representation. For a more complete version, refer to Tables 2 and 3 of [47].

The authors' proposed model is one of the most comprehensive models found in the literature, and also one of the best suited to the identified generic attributes, as seen in Tables 2.2 and 2.5. The empirical study that is shown to be the root of this proposal is also regarded as an advantage.

There are, however, some disadvantages due to its high level of complexity. Firstly, as we can see in Table 2.1 or by studying [47][48][49], this model is based on an informal description, which means that a big level of complexity might induce some level of ambiguity or doubt, which in turn might rise the number of mistakes during the description of a service. Moreover, while a complex, formal model might hide some of its complexity for some stakeholders by applying a set of rules that return simpler, contextual views, an informal model is not capable of such operations. Finally, the names of this model's elements have slight variations between the articles, and some main and minor classes appear and disappear across them, which makes it difficult to summarize

| | | | | | | |
|---|---|---|---|---|---|---|
| Customers | Customer features | Customer attitudes | Customer preferences | | | |
| Goals | Business goals | Service goals | Customer goals | Culture goals | | |
| Inputs | Physical | Human beings | Information | Knowledge | Currency | Constraints |
| Outputs | Physical | Human beings | Information | Knowledge | Currency | Waste |
| Processes | Service provision | Service operations | Service support | Customer relationships | Planning and control | Call center management |
| Human enablers | Service providers | Support providers | Management | Owner organization | | |
| Physical enablers | Buildings | Equipment | Furnishings | Location | Owner organization | |
| Informatic enablers | Information | Knowledge | Procedures and processes | Decision support | Skill acquisition | |
| Environment | Physical factors | Economic factors | Technological factors | Social factors | Ecological factors | Legal factors |

Figure 2.2: CAIOPHYKE table (major classes and their main classes) [49]

all the information scattered across the sources.

### 2.3.4 e$^3$service

e$^3$service[1] is an ontology for configuring IT services based on consumer needs [50] proposed in [51] that belongs to the e$^3$family[2]. This ontology puts a greater focus on satisfying consumer needs and displaying the various value offerings from different services for an easier comparison. As such, its elements are very different from the ones in other service model approaches. Its needs ontology is displayed in Figure 2.3.

In this ontology, there is an `Elementary service` performed by a `Supplier`, that has a certain `Benefit` (which, in turn, has a `Consequence`). A `Consumer` gives a `Sacrifice` in order to have a `Functional need`, which is `Concretized by` a `Want`, which in turn is `Concretized by` a `Demand` that has a certain `Benefit`. `Wants` and `Consequences` are also connected by an `Adds value` element [50].

The biggest conceptual difference between e$^3$service and USDL is that the former focuses on the relations of multiple services, while the later focuses on the description of a single service.

This model is a valuable contribution to the state of the art, as it is represented by a machine-readable ontology, which is the level of formality we want for our model. However, as stated before, the scope of this model is

[1] `http://e3value.few.vu.nl/e3family/e3service/`
[2] `http://e3value.few.vu.nl/`

| Generic attributes | Model attributes |
|---|---|
| Goals | Need, Want, Demand |
| Stakeholders | Consumer, Supplier |
| Processes | Elementary service |
| Inputs | - |
| Outputs | Consequence, Benefit, Value derivation |
| Resources | - |
| Measures | - |
| Legal | - |
| Financial | Sacrifices |

Table 2.6: Attribute analysis of the e$^3$service needs ontology [50]

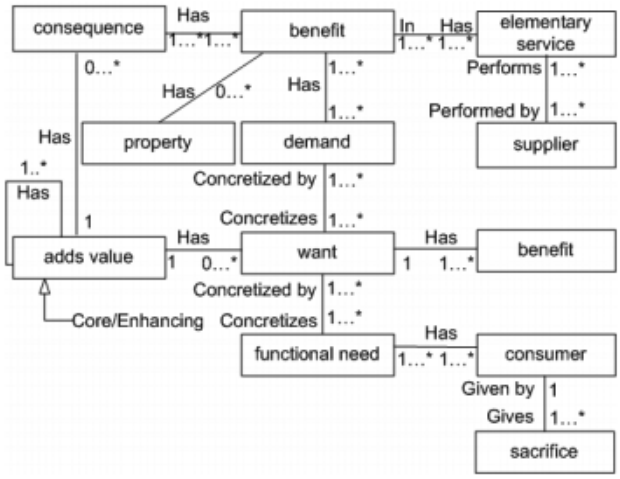


Figure 2.3: e$^3$service needs ontology [50]

for customers, not managers, and so it describes an external representation of a service, and not an internal representation, as we want for our model. In addition, as we can see in Table 2.1, e$^3$service falls in the Service networks group of Ferrario et al. [32], which is also different from what we need.

### 2.3.5 Ten foundational service science concepts

Spohrer and Maglio define service as value-cocreaton [84]. As such, they list ten foundational concepts to describe it. Those concepts are `Ecology`, `Entities`, `Interactions (networks)`, `Outcomes`, `Value proposition based interactions (individuals)`, `Governance mechanism based interactions (collective)`, `Stakeholders`, `Measures`, `Resources`, `Access rights` and `Questions` [84]. These ten foundational concepts are then connected, as seen in Figure 2.4.

| Generic attributes | Model attributes |
|---|---|
| Goals | Questions |
| Stakeholders | Stakeholders |
| Processes | Value proposition based interactions, Interactions (Networks) |
| Inputs | Ecology |
| Outputs | Outcomes |
| Resources | Entities, Resources |
| Measures | Measures |
| Legal | Governance mechanism based interactions, Access rights |
| Financial | - |

Table 2.7: Attribute analysis of the ten foundational service science concepts [84]

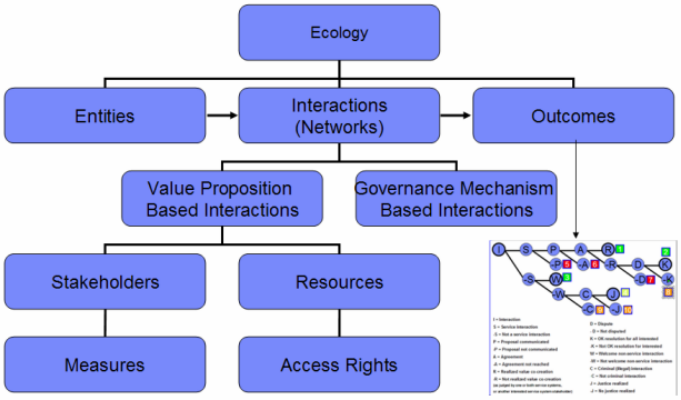


Figure 2.4: Ten foundational concepts of service science diagram [84]

As we can see in Tables 2.2 and 2.7, this model is one of the most complete according to the identified generic attributes that can be found in the literature. It includes elements such as `Measures`, which is surprisingly rare in the other approaches, and the attribute `Stakeholders` is very comprehensive, defining customers, providers, authorities and competitors [84].

There are, however, some flaws in this model, such as the lack of elements for financial attributes description, or the strange naming for some of the foundational concepts (such as `Ecology` for `Inputs` or `Questions` for `Goals`).

### 2.3.6 Business Model Canvas

The Business Model Ontology was created by Alexander Osterwalder in his PhD thesis by identifying the most common business model building blocks in the literature [73], and later explained in [75]. The resulting nine building blocks,

categorized according to four different pillars, were: `Value proposition` under the pillar `Product`, `Target customer`, `Distribution channel` and `Relationship` under the pillar `Customer interface`, `Value configuration`, `Capability` and `Partnership` under the pillar `Infrastructure management` and `Cost structure` and `Revenue model` under the pillar `Financial Aspects`. For a graphical representation of the Business Model Ontology refer to Figure 21 of [73].

| Generic attributes | Model attributes |
|---:|---|
| Goals | Value proposition |
| Stakeholders | Customer segments, Key partnerships |
| Processes | Key activities, Channels, Customer relationship |
| Inputs | - |
| Outputs | - |
| Resources | Key resources (physical, financial, intellectual, human) |
| Measures | - |
| Legal | - |
| Financial | Cost structure, Revenue streams |

Table 2.8: Attribute analysis of the Business Model Canvas [74]

Osterwalder and Pigneur later presented an improved version of the ontology, which they named Business Model Canvas[3]. The improved model uses a high-level graphical representation for informal service description, instead of the low-level ontology elements for a more formal description. This new model drops the usage of pillars, and the new building blocks are `Value proposition`, `Customer segments`, `Channels`, `Customer relationships`, `Key activities`, `Key resources`, `Key partners`, `Cost structure` and `Revenue Streams` [74]. Its graphical representation is shown in Figure 2.5.

The Business Model Canvas has the advantage of offering a very simple and easy to understand tool for managers to describe their services. Another notable advantage of this model is its appraisal by other authors that built their models on top of this, such as Fielt [34] in the next described approach and Zolnowski et al. [92] in the last approach. Moreover, its pragmatism and popularity ensures that this is an influential work that is worth studying.

However, some disadvantages arise when studying this model with our goals in mind. Firstly, its simplicity, although very desirable in some scenarios, is actually very limiting for a complete service description as intended. In addition,

[3]Available online at `http://www.businessmodelgeneration.com/`

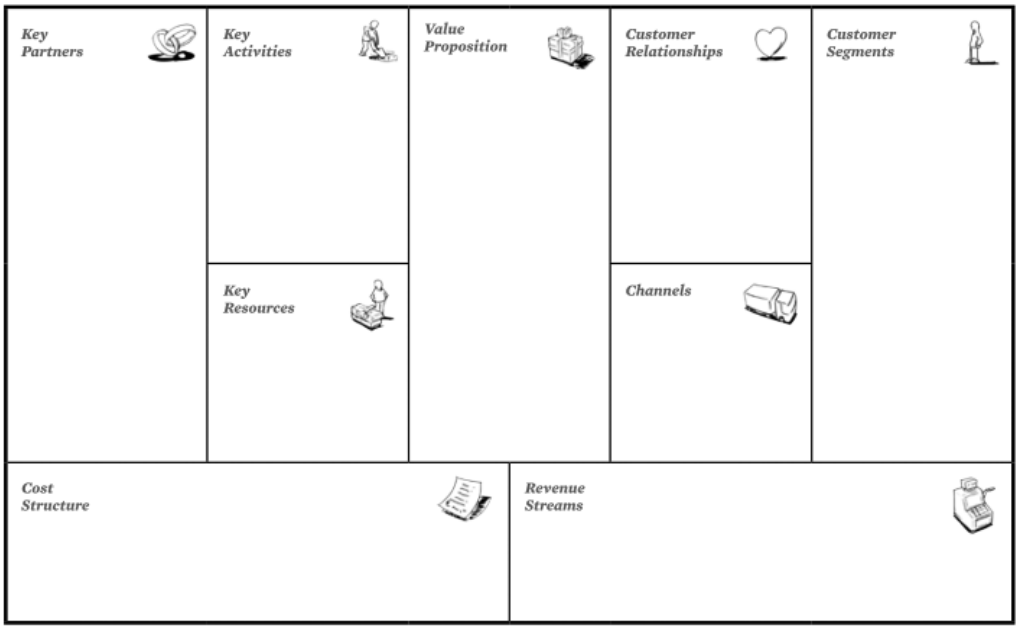


Figure 2.5: Business Model Canvas [74]

the model imposes a very informal description in each of its attributes, which is the opposite of the desired level of formality.

### 2.3.7 Extended Business Model Canvas for Co-Creation and Partnering

This extended version of Osterwalder's Business Model Canvas [74] was developed by Erwin Fielt after two previous attempts [33][35] at improving the original model. This model extends the original one in the following aspects [34]:

- `Customer segments` is called `Target customer` and gets the building blocks `Customer activities` and `Customer resources`
- `Key partners` gets the building blocks `Partner activities` and `Partner resources`
- `Cost structure` gets the building block `Partner cost structure`
- `Revenue streams` gets the building block `Customer cost structure`

Figure 2.6 shows the graphical representation of the improved model in an easy way to compare to the original one.

This model improves the original one by addressing what Fielt identifies as Osterwalder's model's biggest flaws: partnering (in [35]) and co-creation (in [33]). This third iteration builds on top of those two by combining their new elements.

| Generic attributes | Model attributes |
|---|---|
| Goals | Value proposition |
| Stakeholders | Target customer, Key partners |
| Processes | Key activities, Channels, Customer relationships |
| Inputs | Partner activities, Customer activities |
| Outputs | - |
| Resources | Partner resources, Key resources, Customer resources |
| Measures | - |
| Legal | - |
| Financial | Cost structure, Partner cost structure, Customer cost structure, Revenue streams |

Table 2.9: Attribute analysis of the Extended Business Model Canvas for Co-Creation and Partnering [34]

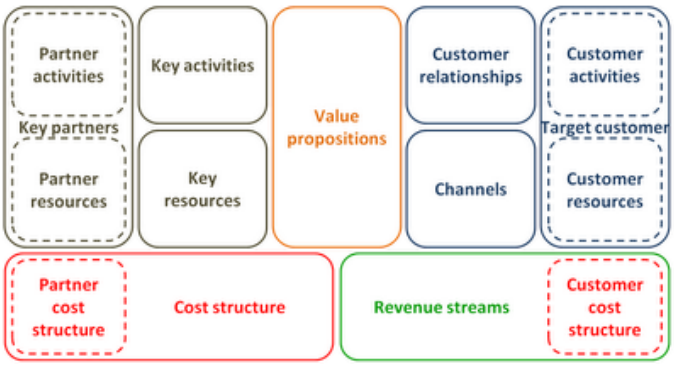


Figure 2.6: Extended Business Model Canvas for Co-Creation and Partnership [34]

Regarding our goals, this model addresses the high simplification of the original model, which hints at better, more complete service descriptions.

Fielt's model does not, however, share the same level of acceptance of Osterwalder's work due to the lack of scientific publications and marketing efforts. Furthermore, although six elements were added, Table 2.2 shows that this model only contributes to one more element of the common attributes, so there is a risk that this increase in complexity might not be very beneficial.

### 2.3.8 Adapted Business Model Canvas

This new adaptation of Osterwalder's Business Model Canvas [74] tries once again to tackle the issue of the lack of elements to describe service co-creation.

However, this new approach focuses on a redistribution of the elements and their connections, rather than changing them as seen in Fielt's approach [34]. As such, this model shares the same building blocks as the original Business Model Canvas (although some of them might have a slight renaming, still maintaining its semantic value). The big difference then lies within the organization of those building blocks and the connections between them, as seen in the respective diagram in Figure 2.7.

| Generic attributes | Model attributes |
|---:|---|
| Goals | Value proposition |
| Stakeholders | Customers, Key partners |
| Processes | Key activities, Channels, Customer relationship |
| Inputs | - |
| Outputs | - |
| Resources | Key resources |
| Measures | - |
| Legal | - |
| Financial | Cost structure, Revenue streams |

Table 2.10: Attribute analysis of the Adapted Business Model Canvas [92]

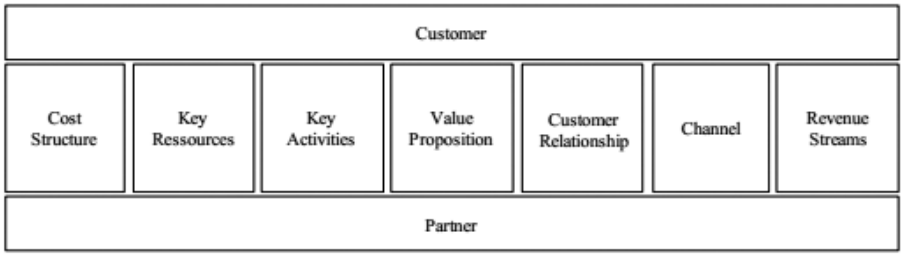


Figure 2.7: Adapted Business Model Canvas [92]

One of the biggest advantages of this model is that its author states that it focuses directly in services and their specific aspects, as opposed to other models, which typically put a greater focus on business in general [92]. Also, as in Fielt's approach, this model emerges in an attempt to correct identified flaws of the Business Model Canvas which, according to Zolnowski et al., are the lack of attention for the specific aspects of services and the lack of elements to describe service co-creation [92].

This model suffers, however, from one of the problems of Fielt's approach:

due to the high popularity and usage of Osterwalder's original model, the improved version does not share the same amount of acceptance. In addition, there is little contribution to be found in this model after analyzing the original Business Model Canvas for our study of the most common attributes of service models, as it shares the same elements, only with different connections and placements.

A further comparison of the original Business Model Canvas and the improved approaches by Fielt and by Zolnowski et al., along with more detailed explanations of the improvements of each of the aforementioned new models, can be read in [93] and are thus left out of this literature review work.

## 2.4 Technical service descriptions

In the last section we discussed several high-level models and frameworks that different authors followed in order to describe service systems and businesses in general. This discussion provided useful insights about important elements to consider for service systems description in a business perspective. In this section we focus on existing technical tools and ontologies for service description, in order to acquire a better understanding of the technical perspective of our problem.

### 2.4.1 *-USDL

The Unified Service Description Language (USDL) is a tool for describing various types of services ranging from professional to electronic services [8]. The USDL family (*-USDL) is composed of $\alpha$-USDL, which is the original USDL version that was released in 2009, USDL, which is currently at version 3.0, and Linked USDL, which is a simpler description model that builds on top of the Semantic Web technologies [18].

This family of service description tools are classified by Ferrario et al. as purely economic, since they model generic service systems regardless of their technological implementation, merging the business and technological scope of services [32].

USDL is an effort to merge the business, operational and technical perspectives of service systems: the business perspective describes the fundamental properties for characterizing a service, the operational perspective describes the operations that are executed by a service and the technical perspective allows the specification of technical information of services exposed by an organization [19]. USDL is based on 9 modules: foundation, service level, participants, pricing, legal, service, interaction, functional and technical [69]. Their relation is show in Figure 2.8.

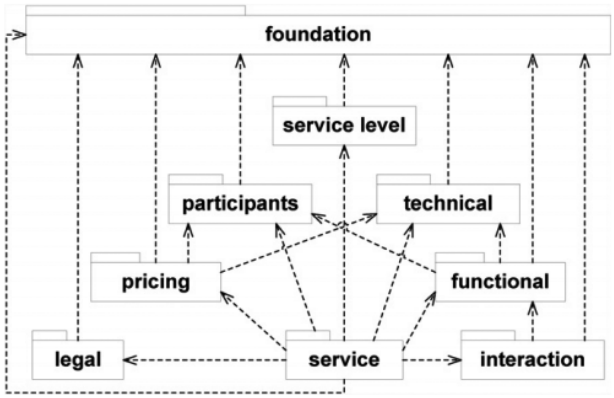


Figure 2.8: USDL dependencies between modules [69]

Linked USDL is an attempt to overcome some of the original USDL disadvantages, like the model complexity and limitations in extensibility. Thus, it is built using RDF and uses Linked Data, which increases interoperability by allowing sophisticated ontology representation techniques [25]. It also greatly simplifies USDL's original structure, having only 4 modules: USDL-Core, USDL-Pricing, USDL-SLA and USDL-SEC [2].

USDL and Linked USDL are service description tools that can model generic service systems, which makes them very similar to our intended model. However, they are limited to a blackbox approach, which makes them unsuitable for a complete service system description. Thus, although they share some traits with our intended model, they do not comply with all our goals and therefore do not eliminate our need for creating a new model.

### 2.4.2 WSDL

The Web Services Description Language (WSDL) is a model and XML format for describing web services [22]. Its formalization was published in 2001 as version 1.1 [23]. Its current version is 2.0, which is a W3C recommendation [16].

This service description tool focuses on web services and not in business services. Hence, according to Ferrario et al.'s classification, WSDL belongs to the group of Service-Oriented Architectures (SOA) [32].

WSDL enables the separation of the description of the service's abstract functionality from the service's concrete details. At the abstract level the web service is described in terms of messages it sends and receives. An operation associates message exchange patterns with messages and an interface is a collection of operations. At the concrete level a binding specifies interfaces details, an endpoint associates a network address with a binding and a service groups the endpoints that implement a common interface [22].

This tool can be improved by using Semantic Annotations for WSDL and

XML Schema (SAWSDL), which is a set of extension attributes that add semantic annotations to WSDL components [54].

WSDL is a widely used service description tool and its current version is a W3C recommendation. However, since it focuses on web services and technological details, it is not capable of providing relevant descriptions of arbitrary services, specially non-technological ones.

### 2.4.3 WSMO

The Web Service Modeling Ontology (WSMO) is a conceptual framework and formal language for semantically describing web services in order to aid in the automation of discovering, combining and invoking services over the web [24]. This ontology is based on the Web Service Modeling Framework [29].

Due to its focus on web services and semantic technologies, WSMO is classified according to Ferrario et al. in the category of Semantic Web Services.

The overall structure of WSMO is divided in four main elements: ontologies, which provide the terminology used by other WSMO elements, web services, which provide access to services that provide value in some domain, goals, which represent user desires and mediators, which deal with interoperability problems between different WSMO elements [79].

This ontology is also the inspiration for WSMO-Lite, a lightweight set of semantic service descriptions that can be used for annotations of WSDL elements using the SAWSDL annotation mechanism. This set of semantic service descriptions uses a small subset of WSMO to define a limited extension of SAWSDL [30].

WSMO provides a solution for semantic web services description but, as other service description tools, is not suitable to describe general service systems.

### 2.4.4 OWL-S

OWL-S[4] is an ontology built on top of OWL[5] [63] that facilitates the automatic discovery, invocation, composition and monitoring of web resources offering particular services or properties. It is made of three different parts: the service profile, the process model and the grounding that provides details on how to inter-operate with services through messages [61].

Since it is build on top of a semantic language and aimed at the Semantic Web, OWL-S is classified according to Ferrario et al. in the category of Semantic Web Services.

The top level structure of this ontology provides three essential types of knowledge about a service, as seen in Figure 2.9. These types of knowledge try to an-

[4]Semantic Markup for Web Language
[5]Ontology Web Language

swer the questions: "What does the service provide for prospective clients?" (answered in `ServiceProfile`), "How is it used?" (answered in `ServiceModel`) and "How does one interact with it?" (answered in `ServiceGrounding`).

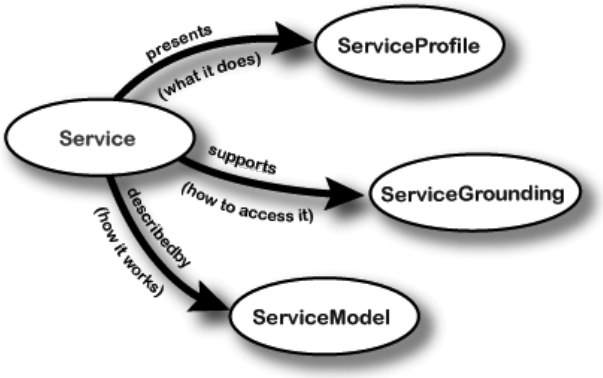


Figure 2.9: OWL-S top level [61]

As seen in other service description tools, OWL-S provides the necessary structure to automate services discovery and automation and to integrate them with the Semantic Web, but also lacks a broader scope, as it only focuses on web services, and not on generic service systems.

### 2.4.5 SWSF

The Semantic Web Services Framework (SWSF) [9] is a tool for service description based on two major components: the Semantic Web Services Language (SWSL) [10] and the Semantic Web Services Ontology (SWSO) [11].

This tool is classified according to Ferrario et al. in the category of Semantic Web Services, due to both the importance it gives to semantic web services and its usage of Semantic Web tools.

SWSL is used to specify the SWSO and individual web services [10].

SWSO is expressed in two forms: FLOWS, which is a first-order logic ontology for web services, and ROWS, which is a rules ontology for web services [11].

Comparing to OWL-S, SWSF presents a more complete descriptive language, because while SWSF uses FLOWS, which is expressed in first-order logic [11], OWL-S is expressed in OWL-DL [61], which sacrifices expressiveness in favor of computational completeness and decidability [63].

Similarly to WSMO and OWL-S, SWSF is seen as a positive attempt to use Semantic Web technologies to facilitate services description and interaction, while lacking the possibility to be used in generic service systems.

# 3

# Service system modeling

This chapter presents our proposed model, along with its building blocks and iterative approach.

The first section presents the concepts and building blocks to be used for the design of the service system model. The second section describes how they are used in an iterative approach to develop the desired model. Finally, the third section described the resulting service system model in greater depth.

## 3.1 Concepts and building blocks

In Chapter 2 we studied some approaches for business and service modeling. By analyzing the most common attributes across them, we were able to build a framework to evaluate service modeling approaches. This framework combines the knowledge gathered by different authors in order to provide a set of attributes commonly used for the description of a service. Those attributes are `Goals`, `Stakeholders`, `Processes`, `Inputs`, `Outputs`, `Resources`, `Measures`, `Legal` and `Financial`. Therefore, this framework is also a valuable resource for the specification of the service system model.

For the specification of the service system model, we may also classify its components according to the interrogative pronouns commonly used in journalism: "What", "How", "Where", "Who", "When" and "Why". This strategy in not uncommon in the literature, as we can see it in Zachman's framework for enterprise architecture [90][91] and other approaches by different authors [15][27][82]. This classification enhances readability and understandability, gives an intuitive meaning to abstract concepts and helps organizations to ask questions about their processes and process models [82]. It also helps identifying some characteristics of a service offer (such as provider, channel or payment) and can be used as a common framework for querying different services [27].

Another useful source for service design is the service blueprint. Service

blueprinting is a method created by Shostack [81] that is used for analyzing the service delivery process, by using a flow chart-like presentation to distinguish several types of customer interaction [56]. When designing a service blueprint, the horizontal axis indicates the chronological sequence, while the vertical axis separates the different areas of actions [36]. Furthermore, the vertical axis contains a set of lines that separate the different areas of context. The first, called line of interaction, separates the customer action area from the service provider action area, the second, called line of visibility, separates visible from invisible actions in the customer perspective, and so forth [36]. The line of visibility is also regarded as the separation between the service "font stage" and "back stage" [38]. Since this is a well known method for service modeling that is easy to understand and use by business managers, it is thus a very important inspiration for our model. Hence, although the desired model is meant to work as an ontology for service systems, its graphical representation should resemble an improved version of the service blueprint tool.

Finally, the concept of co-creation is also used. This concept shifts our study of economic activity from a Goods-Dominant logic (GD) where value exchange is perceived through goods transactions (such as selling/buying a product) to a Service-Dominant logic (SD) where value exchange is co-created by all parties of service interactions [58]. As such, we no longer see value exchange as a provider delivering value to a customer by selling a product, but rather as both provider and customer co-creating value to each other during the interactions of the service.

Using these four building blocks it is now possible to develop the desired service model. The approach for joining these building blocks and reaching a solution is described in the next section.

## 3.2 Approach

Based on the notion that co-creation during service interactions is a core feature of service systems [58] and that the interactions flow is also a core feature in service blueprints [81] we can conclude that a service system should represented by its flow of interactions and their contextual information, such as the value being co-created. Therefore, we should focus on how to describe service interactions, their context and their flow.

Matching the framework for service modeling derived in the study of the state of the art with the interrogative pronouns, we get the attribute `Stakeholders` for the pronoun "who", the attribute `Goals` for the pronoun "why", the attribute `Resource` for the pronoun "what" and the attribute `Process` for the pronoun "how". The interrogative pronouns "when" and "where" are easily matched with the spatial and temporal context, respectively, of a service interaction. As

such, we can describe service interactions with the six interrogative pronouns by using the following attributes:

- Who: `Role` (human or computer actor belonging to a stakeholder)
- Why: `Goal` (a service goal for the specified actor)
- What: `Resource` (may be physical, knowledge or financial)
- How: `Process` (the business process a service interaction belongs to)
- When: `Time` (may contain different levels of granularity)
- Where: `Location` (may contain different levels of granularity)

We now have a 6-point interaction star model for describing service interactions, as seen in Figure 3.1.

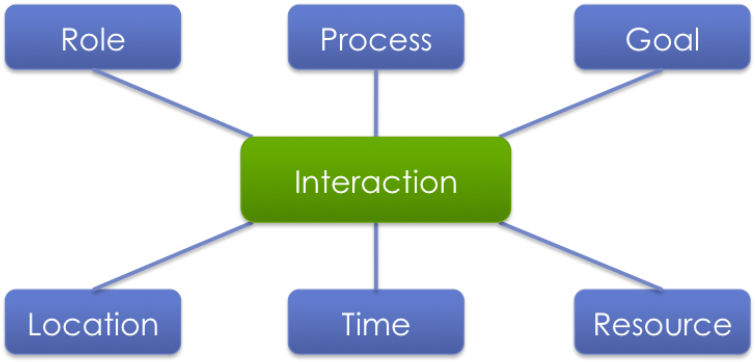


Figure 3.1: 6-point interaction star model

Moreover, inspired by the work of service blueprint, we may also classify interactions based on their area of action. Thus, an interaction can be a customer interaction, an onstage interaction, a backstage interaction or a support interaction [36].

In the foundational ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) resources may be classified as endurants if they are physical objects or perdurants if they are not physical, such as services or events [62]. Poels classifies resources as operand if they are passive resources like objects or operant if they are knowledge and skills that embody competences [78]. We can also find this pattern in some of the models we studied in Chapter 2 [49][74]. Therefore, resources should be classified as physical or knowledge. We also consider a third classification, financial resources, because it is important in a business-oriented model to give high visibility of that particular kind of resource.

These extensions to the interaction and resource entities are depicted in Figure 3.2.

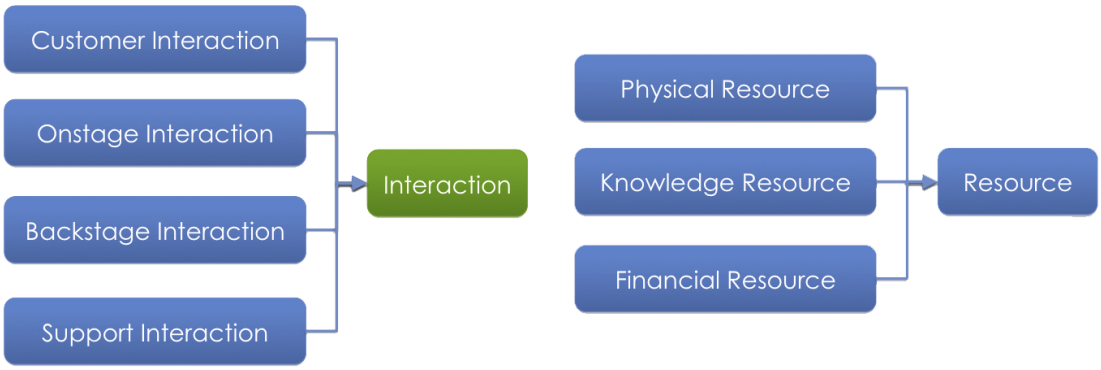


Figure 3.2: Extensions to interaction and resource entities

Since we are targeting this model to interact with the Linked Data Cloud (LDC) we need to build it according to the Semantic Web principles and integrate it with other ontologies. This means that now the connection between entities must have a semantic meaning expressed by an RDF triple, such as "`Interaction isPerformedBy Role`". The integration with th LDC is done by reusing relevant Linked Data ontologies, such as Geonames to give a geographical context to locations or DBpedia to give a better semantic meaning to resources. The resulting model is the proposed service system model of this thesis and is fully described in the next section.

## 3.3 The LSS-USDL model

Due to the aforementioned integration with the LDC, we are modeling Linked Services [76]. Because of that and since our goal, inserted in the USDL research efforts, is modeling service systems, we name our model Linked Service Systems for USDL (LSS-USDL). Its logo, shown in Figure 3.3, is inspired in Linked USDL's logo but, because we are moving from a blackbox to a whitebox model, only the borders are colored and the inside is now white.


Figure 3.3: LSS-USDL logo

This model was developed as an RDF ontology. This ontology is written in Turtle as opposed to XML due to its better readability [12]. The code is open source and is available at `https://github.com/rplopes/lss-usdl` under the Creative Commons Attribution 3.0 Unported License[1], in order to allow future improvements from the open source community.

Figure 3.4 shows the final version of our service model, including semantic relations and elements from external ontologies that are used. This figure does not include interaction and resource subclasses, which can be seen in Figure 3.2.

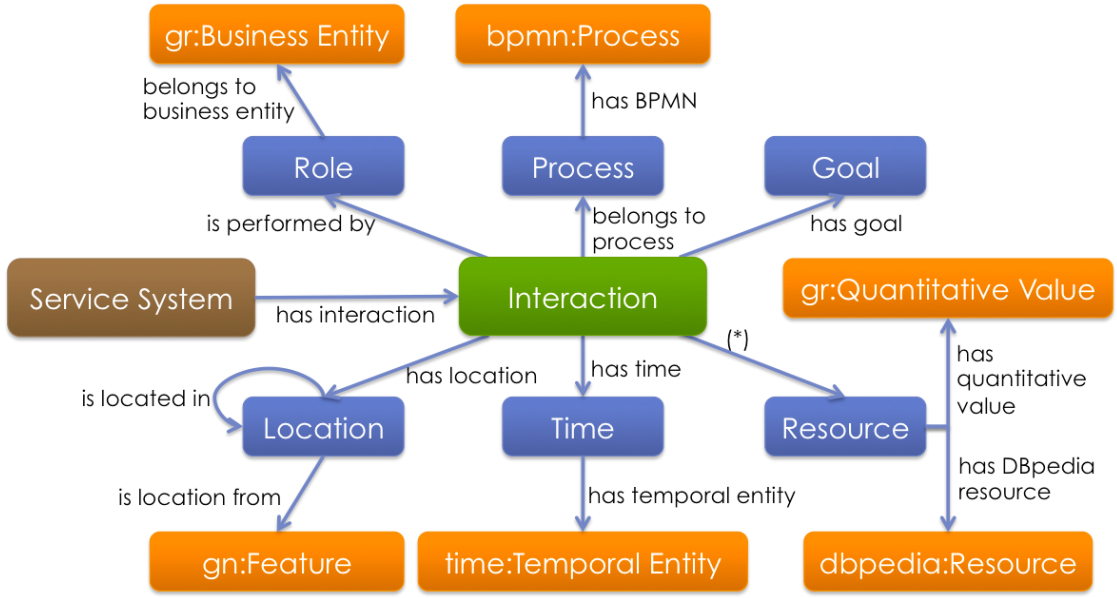


Figure 3.4: Complete service model

We can see the original 6-point interaction star as the core of the model. As such, an `Interaction`, which is considered as a core element of service system modeling, may be characterized by the roles that perform it, the processes that it belongs to, its goals and locations, its temporal description and temporal relation to other interactions and the inputs and outputs of resources. A `Service System` entity was added to the model to group a collection of interactions.

A `Role` might represent customers, managers, computer agents and so on. We can associate a role to their respective stakeholder with the property `belongsToBusinessEntity`. That property connects a `Role` to a `Business`

[1] `http://creativecommons.org/licenses/by/3.0`

`Entity`[2] of the ontology GoodRelations [42]. This ontology was chosen because it is widely accepted as a valuable Linked Data vocabulary for describing products and services [14][41]. A `Business Entity` from GoodRelations "'... represents the legal agent making (or seeking) a particular offering" [1], which makes it an adequate match for expressing the stakeholder of a `Role`.

A `Process` indicates an internal business process of the service system. This entity is particularly useful to filter interaction flows based on certain processes. The usefulness of such an entity can be improved by connecting it to modeled processes. As such, we link it to a `Process` of the BPMN 2.0 ontology [67]. In future work other connections to different process modeling vocabularies may be considered in order to better expand the usefulness of this entity.

A `Goal` expresses a motivation for the occurrence of the interaction. This entity is not connected to any element of the Linked Data Cloud because its meaning is contained in the context of its service system. Moreover, no relevant ontologies were found that could be used to extend the information of this entity.

A `Location` expresses a conceptual location where an interaction occurs, such as a room or a store. Because an interaction that takes place in a certain room also takes place in that room's floor, department, building and so on, the property `isLocatedIn` connects two instances of `Location` to impose a hierarchy level. This enables, for instance, associating an interaction to a room and finding that interaction when querying that room's building. This entity has also the property `isLocationFrom` that connects it to a `Feature` of the ontology Geonames [88]. This enables us to give an unambiguous geographical context to any `Location`, since a Geonames `Feature` represents any city, country, continent and so on and also uses a hierarchy level.

`Time` is the entity that gives a temporal context to an interaction. It is connected to a `Temporal Entity` of the OWL-Time ontology [44]. This connection enables a great level of detail for temporal descriptions, such as the date and time of its occurrence with `DateTimeDescription` or its duration with `DurationDescription`. In addition, it is also possible to define temporal relations between interactions through the use of properties such as `intervalBefore`, `intervalEquals` or `intervalAfter`.

A `Resource` represents some input or output of the service system. As such, an interaction can be related to a resource by four different properties: `receivesResource` when it is being introduced from outside of the service system, `createsResource` when it is created from within the service system, `consumesResource` when it is consumed from within the service system and `returnsResource` when it is provided to the outside of the service system. Resources can be connected to two elements from different ontologies:

[2] `http://www.heppnetz.de/ontologies/goodrelations/v1#BusinessEntity`

`Quantitative Value`[3] from GoodRelations [42], so we may specify quantities, and `Resource` from DBpedia [7], so we may give an unambiguous semantic value.

As we discussed in the previous section, `Interaction` and `Resource` also have subclasses to better express their nature. Their usage, however, is not mandatory, and other subclasses might be used instead if they are a better fit to the service system that is being modeled. That is possible because both `Interaction` and `Resource` are subclasses of `Concept`[4] from the SKOS[5] ontology [46]. This means that they are concepts that can be extended by concept schemes [64]. With that in mind, we can create a `Concept Scheme`[6] from SKOS for `Interaction` and another for `Resource`, create their respective subclasses and add them to their respective concept schemes through the SKOS property `hasTopConcept`[7]. In a similar way, if someone will benefit from a different set of subclasses, they may create a new concept scheme and assign the new subclasses as top concepts. This capability improves the model's adaptivity and capacity to improve.

[3] `http://www.heppnetz.de/ontologies/goodrelations/v1#QuantitativeValue`
[4] `http://www.w3.org/TR/skos-reference/#Concept`
[5] Simple Knowledge Organization System
[6] `http://www.w3.org/TR/skos-reference/#ConceptScheme`
[7] `http://www.w3.org/TR/skos-reference/#hasTopConcept`

# 4

# Tool support

This chapter presents the two proof of concept tools that were developed in order to demonstrate the model's usefulness for real world usage.

The first section briefly explains the motivation for the development of these tools. The second section describes the first tool, a graphical editor for the model. Finally, the third section describes the second tool, a translator to and from a different model, Linked USDL.

## 4.1 Motivation

In the previous chapter we discussed the details of our proposed service system model. However, the LSS-USDL model was developed not only for academic purposes, but also for real world usage. Hence, we need to build a bridge between the theoretical approach presented in the previous chapter and the expected model usage in the field. To do that, we discuss two software tools that aim to demonstrate the applicability of our model for real world usage.

The first tool is an LSS-USDL graphical editor. It intends to prove that it is possible to express modeled service systems in an easily understandable graphical notation and also to edit them using that same notation.

The second tool is a translator to and from Linked USDL [18]. This not only demonstrates the model's ability to generate custom service descriptions such as a customer-based description like Linked USDL, but also proposes an easy transition from service systems expressed in other models to LSS-USDL.

Both tools were developed in Ruby[1] and bundled into one common webapp, because their features bring greater value when combined together. That webapp was developed using the framework Ruby on Rails[2]. In the develop-

[1] `http://www.ruby-lang.org`
[2] `http://rubyonrails.org`

ment environment the application is connected to a SQLite[3] database, and in the production environment it is connected to a PostgreSQL[4] database. The production environment to which the application was deployed was Heroku[5], because it supports all the used technologies and provides a free hosting service. All the code versioning was done in Git[6] and its repository can be found at `https://github.com/rplopes/lss-usdl_editor` as open source with the Creative Commons Attribution 3.0 Unported License[7], in order to allow future improvements from the open source community. A live version was deployed at `http://lss-usdl-editor.herokuapp.com` for demonstration purposes.

The next sections describe in greater detail each of the developed tools.

## 4.2 Graphical LSS-USDL editor

This tool is a prototype that presents a GUI that enables the modeling of service systems without the need to write RDF statements. This is a very important proof of concept not only because service systems are typically designed by managers and business owners, whose technical skills do not usually embrace fluency with programming languages such as RDF, but also because service systems can easily achieve great levels of complexity, so a good UI for data modeling is required to avoid high cognitive work and facilitate rapid data input.

This tool uses the front-end framework Twitter Bootstrap[8] to allow rapidly styling the interface, and Javascript and Coffeescript[9] for small improvements of the user interface.

The application architecture follows an MVC[10] approach, as depicted in Figure 4.1. This means that when a user makes a request to the application, it is received to a router (1) that forwards it to the corresponding controller (2). The controller is where we may find the application logic. The controller then fetches the models it needs (3). A model is a data entity, such as a service system or an interaction. The desired models are fetched from the database (4, 5) and returned to the controller (6). After all the controller's operations, it returns its corresponding view (7), which will generate the HTML page that the user will see (8).

[3] `http://www.sqlite.org`
[4] `http://www.postgresql.org`
[5] `http://heroku.com`
[6] `http://git-scm.com`
[7] `http://creativecommons.org/licenses/by/3.0`
[8] `http://twitter.github.io/bootstrap`
[9] `http://coffeescript.org`
[10] Model-View-Controller

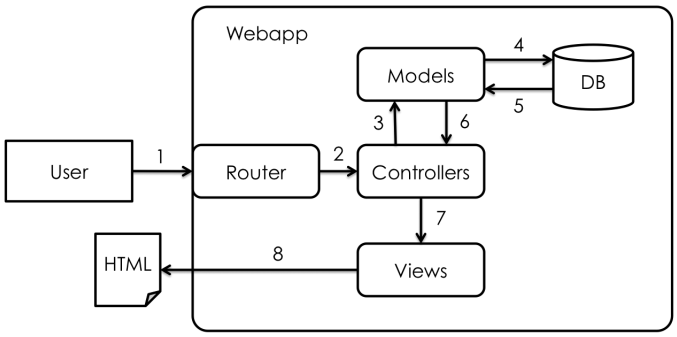


Figure 4.1: Architecture for the graphical editor tool

### 4.2.1 User accounts management

This LSS-USDL editor is an open source tool that any business can download and install in their servers or simply use the provided public version[11]. But because it models internal details of service systems, viewing and editing rights must be taken into account.

Due to that fact, this tool requires user registration to get access to its contents. In the public version of this tool any visitor may create an account, but for business installations the account creation option may be available just for administrators, to block outside access to the service systems models.

In addition, write permissions are only granted to the author of a service system. For a business installation this restriction can be removed by changing the source code. If a different user needs a model of an existing service system but with some alterations, it is possible to export it to a new editor entry and edit it, while still leaving the original model untouched.

Figure 4.2 shows some interfaces available for user accounts management.

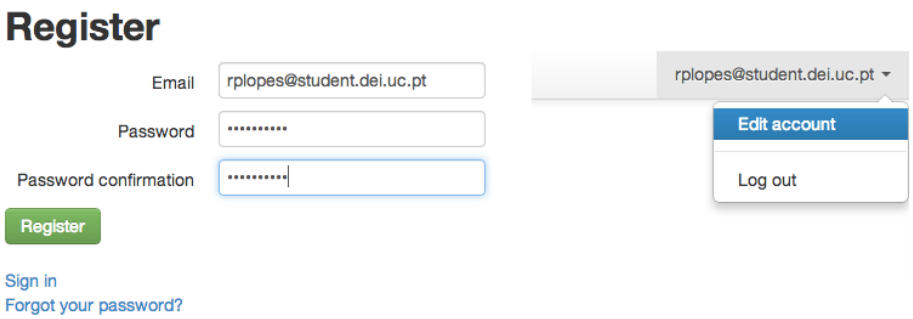


Figure 4.2: Register new account screen and account management menu

[11] http://lss-usdl-editor.herokuapp.com

### 4.2.2 Entities management

The core feature of this tool is the management of service system entities according to the LSS-USDL model. A user may view, create, edit or remove any element of a service system, including the service system itself. An example of editing an entity is shown in Figure 4.3.

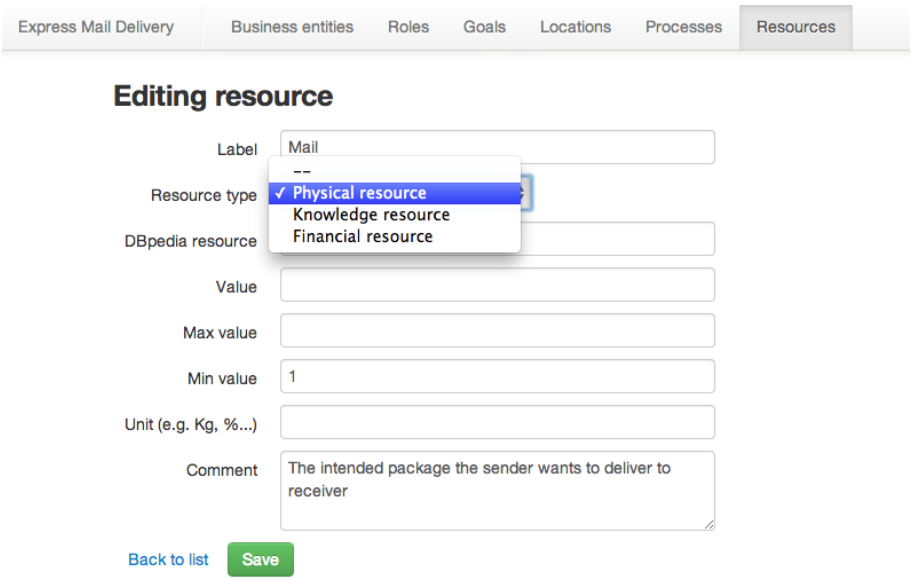


Figure 4.3: Editing a resource entity

All created entities are associated to a single service system. Service systems can be managed in the home screen. When one is selected, it is possible to manage its entities.

In the top navigation bar there are links for managing Business Entities, Roles, Goals, Locations, Processes and Resources. In the front page of a service system it is also possible to manage its Interactions. In the Interaction's form it is also possible to edit its temporal data and connect it to existing service system entities.

For this feature we must consider various compromises. Assigning an entity to a specific service system makes it easier to create scopes and avoid much clutter, but on the other hand it makes it impossible to assign to an interaction entities of another service system (they have to be copied). Moreover, entities from external vocabularies such as `Business Entity` and `Temporal Entity` can hold more data than what this tool manages, but offering a complete solution for those entities would greatly increase the system's complexity with little to no added value to our scope. Because of these compromises we must not label this tool as the best option for modeling any possible service system,

but rather as a proposal among many potential others that is best suited for some scenarios.

### 4.2.3 Connection to the LDC

As we discussed previously, many entities of a service system are connected to elements of the Linked Data Cloud to enable better data description and to facilitate data analysis across different service systems. It is thus important to provide a practical and easy to use interface to enable such connections. An example of such an interface in this tool is shown in Figure 4.4.

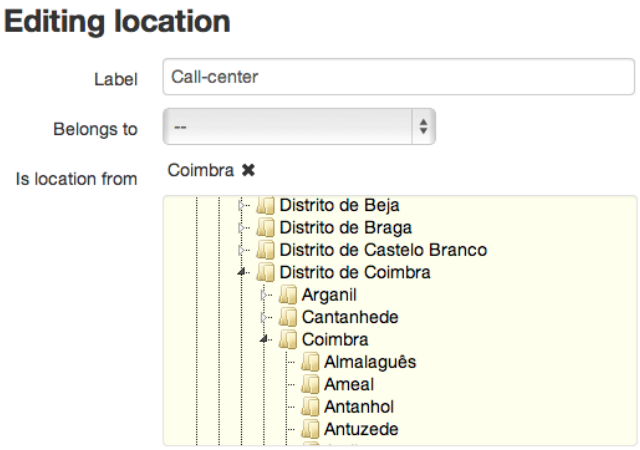


Figure 4.4: Accessing the Geonames Linked Data

When a process has already been modeled using the BPMN 2.0 ontology [67] we can connect it to a service system's `Process` in its form by providing the corresponding URI. In future work this simple approach could be extended by allowing connecting to other process model notations or by parsing the process found and showing some additional information in the tool's UI.

To add a DBpedia [7] `Resouce` to a service system's `Resouce` there is also a field in its form to input its URI. After saving, we can see the name of the DBpedia `Resource` and a link to its unique web page.

Lastly, to add a Geonames [88] `Feature` to a service system's `Location` there is a hierarchical tree view of the elements to choose from. This means that if we want to choose Portugal we just need to select Europe, revealing the list of European countries, and then select our desired option. This greatly improves user input because it does not require typing names (thus avoiding memorization, errors and ambiguities) and it does not overflow the user with meaningless data (e.g. if we want to select a city, we do not see cities from other countries). This interface was adapted from the work of Thomas Haukland [40].

### 4.2.4 Service systems visualization

The core visualization of a service system is the flow of its interactions. When a user selects a service system from the main menu a page is displayed showing its interactions in an extended service blueprint view. This view highlights the temporal dependencies between different interactions and divides them according to their respective areas of action, as required by the traditional service blueprint approach [81].

However, one of the goals of this tool is to prove that service systems modeled in LSS-USDL may have different representations according to a user's needs. Because of that there is an option to change the service system visualization methods. The visualization methods supported in this tool are the aforementioned extended service blueprint and also a plain list that orders interactions based on their temporal relations and displays their connections to the service system's elements. In future work other visualizations can be added.

Finally, the service system's visualization can also be manipulated based on filters. There are filters for Roles, Goals, Locations, Processes and Resources, but more filters can be added in future work. The `Location` filter obeys to the hierarchy found in that class, so if we have an interaction that happens in a room and we filter by interactions happening in that room's building, that interaction will still appear. The use of filters is particularly useful for controlling the data that should be presented to certain stakeholders. Figure 4.5 shows the extended service blueprint view with the list of available filters.

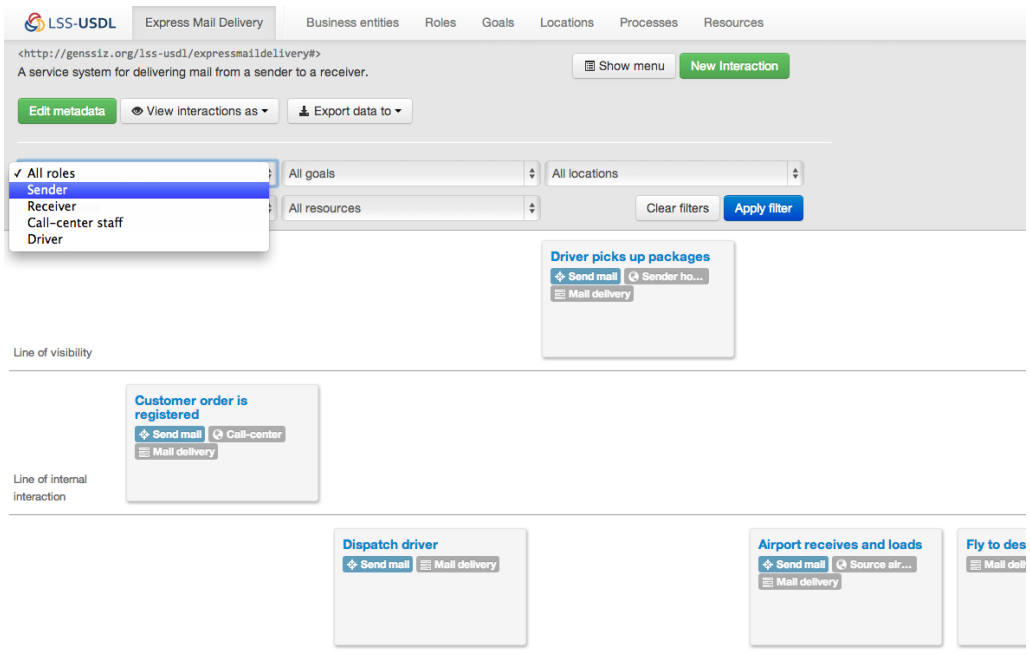


Figure 4.5: The extended service blueprint view and filters list

### 4.2.5 Import/Export from/to RDF files

In order for this tool to be an effective alternative to editing LSS-USDL models directly with RDF code it must provide the ability to seamlessly switch between the RDF representation to the tool's data model. This means that it must be possible to import an existing RDF file containing a service model and to export data to another RDF file. The import interface is shown in Figure 4.6 and the export interface is shown in Figure 4.7.

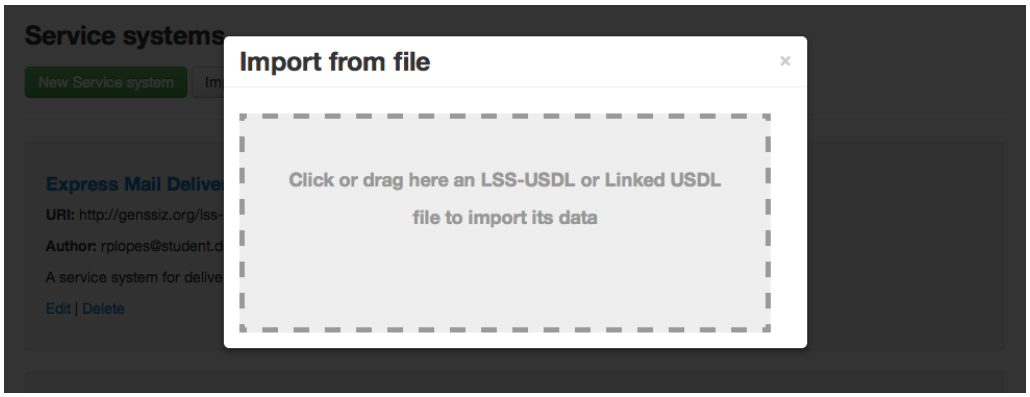


Figure 4.6: File import interface after clicking "Import from file" button

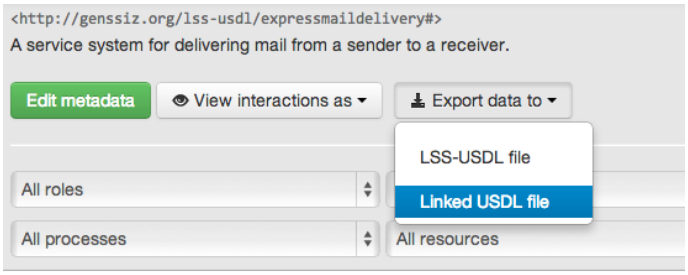


Figure 4.7: Export to Linked USDL added to the export options

This translation to and from RDF is done with the help of the Ruby gems LinkedData[12] for dealing with the RDF semantic graph in Ruby and RDF-Turtle[13] to import/export RDF code in Turtle notation. The Turtle notation [12] was chosen instead of other notations such as XML or N-Triples because of the abbreviations and clean syntax that make it the easiest to write RDF statements manually and to read and understand them.

The import process is done by fetching the semantic graph described in the

[12] `http://rubygems.org/gems/linkeddata`
[13] `http://rubygems.org/gems/rdf-turtle`

file and then applying several queries in which we extract all the information needed to save in the tool's data model. The export process is done by saving to a new semantic graph all the information in the data model of the current service system and then writing the resulting graph to an RDF Turtle file.

The export process also makes use of the applied filters to generate a file with only the requested information. As such, if we want to show to customers only the interactions that concern them, it is easy to generate an RDF file with just that information and thus hide internal data without needing to remove it from the original service model.

Because this tool is a proof of concept, the import process can not find all possible information contained in the RDF file. Elements from external vocabularies such as `Quantitative Value` from GoodRelations [42] or `Temporal Entity` from OWL-Time [44] are able to hold much more information than what is presented in the tool but, as we previously discussed, trying to get all possible data would require a big effort for very little additional value. A more complete import process would, then, be something to consider for future work.

## 4.3 Linked USDL export/import tool

This tool acts as a proof of concept that shows that it is possible to export an LSS-USDL service model into different service descriptions and also to make use of existing service descriptions to rapidly build an LSS-USDL service model. This was added as a feature to the previously described graphical editor because the two tools provide greater value when combined: it makes it possible to edit a service system by importing a Linked USDL file and the export feature is now extended to support exporting both to an LSS-USDL or Linked USDL file.

Figure 4.8 depicts the architecture of this tool, integrated in the webapp. When the user's request for an import or export operation reaches the controller, the special model `SemanticWorker` is called. This model does not represent any entity nor does it have a related database table. It does, however, interact with the database multiple times in order to create new entities from the provided RDF file or create a new semantic graph based on the existing entities. Its response (6) will be a confirmation for imports or a semantic graph for exports. If the user is importing a file, the controller will then call the view of the newly imported service system (7a) to be displayed to the user (8a). If the user is exporting a service system, the controller will return the exported RDF file (7b). The import/export feature of the previous tool also follows this architecture.

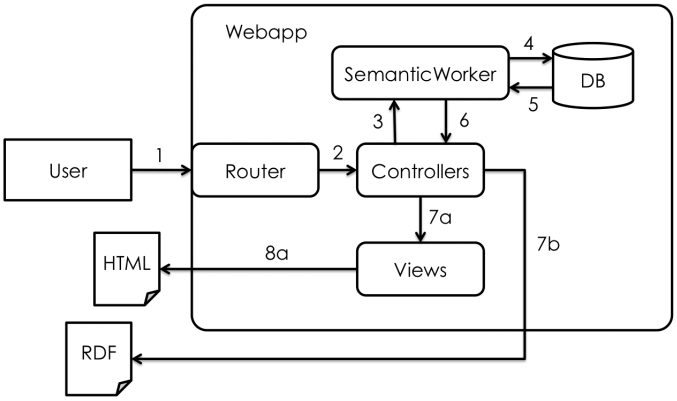


Figure 4.8: Architecture for the Linked USDL export/import tool

### 4.3.1 Mapping between the two models

One of the biggest challenges when creating a tool that translates data from a model to another is to find the appropriate mappings that enable such a translation. Table 4.1 resumes the findings on those mappings.

| LSS-USDL | Linked-USDL |
|---|---|
| Service System | Service |
| Customer Interaction | Interaction Point |
| Role | Interacting Entity |
| Time | Time Spanning Entity |
| Resource | rdf:Resource |
| hasInteraction | hasInteractionPoint |
| isPerformedBy | hasInteractingEntity |
| hasTemporalEntity | spansInterval |
| receivesResource | receives |
| returnsResource | yields |

Table 4.1: Mapping of LSS-USDL elements to Linked-USDL elements

Mapping Service Systems and Interactions to the Linked USDL elements `Service` and `Interaction Point` is straightforward. The other elements, however, require closer attention.

For Roles we may find `Interacting Entity`, which can hold an `Interaction Role` or a `Business Role`. Both these entities can be further specified through the use of SKOS concept schemes [64].

The closest element to LSS-USDL Time is Time Spanning Entity. However, that is a superclass for Interaction Point. This means that the connection to OWL-Time is done directly in the interaction. So while in LSS-USDL we have

an interaction connected to Time connected to the OWL-Time ontology with the property `hasTemporalEntity`, in Linked-USDL we have an interaction connected directly to that ontology with the property `spansInterval`.

A `Resource` is handled in Linked USDL as an RDF `Resource`. Its connection to an interaction can be indicated by the properties `receives` for input and `yields` for output. These are similar to the LSS-USDL properties `receivesResource` and `returnsResource`, respectively. The other two properties for LSS-USDL Resources (`createsResource` and `consumesResource`) are absent in Linked USDL because they concern internal operations.

### 4.3.2 Import/Export from/to RDF files

The work on the translation between Linked USDL RDF code and LSS-USDL objects in the graphical editor's data model builds on the previous efforts to import and export RDF files described in the previous section. We can see integrated functionality in Figures 4.6 and 4.7. This means that the import process is based on queries to the semantic graph and the export process is based on populating a new semantic graph and writing it to an RDF file using the Turtle notation.

As with the previously described efforts, this work is intended to be a proof of concept, not an exhaustive conversion tool. As such, the data it extracts from Linked USDL files might be just a portion of the original data fitting our needs.

Moreover, since the mapping between the two models is not 100% complete, not only do we lose the information that is not present in the mapping, but we also get an incomplete service model with no information on processes, goals and locations and with no interactions besides the ones from customers. However, this incomplete service model can be seen as a "quick start" and a time saver to model services in LSS-USDL.

# 5 Evaluation

This chapter presents the evaluation process used to validate the usefulness and applicability of our model through the use of three different use cases.

The first section briefly describes the motivation for this evaluation process and its approach. The second, third and fourth sections describe how the evaluation was performed by recurring to different use cases and different approaches, one per section. Finally, the fifth section discusses the results of the evaluation.

## 5.1 Evaluation approach

The model presented in Chapter 3 is, as previously stated, an important proposal that aims towards better formalization and standardization of service systems management. It is, thus, of paramount importance to evaluate our findings and proposals so as to understand the feasibility of such a solution to our problem.

In order to do so, three different use cases have been chosen. These use cases are examples of service systems found in the literature. These examples have been originally modeled according to the service blueprint method [81], so it is possible to adapt them to our model and draw conclusions afterwards.

In addition, two software tools, described in the previous chapter, were built in order to demonstrate the applicability of our model to real world usage. The first tool, a graphical editor, is intended to prove that it is possible to express modeled service systems in an easily understandable graphical notation and also to edit them using such a notation. The second tool, a translator to and from Linked USDL [18], intends not only to demonstrate the model's ability to generate custom service descriptions such as a customer-based description like Linked USDL, but also to propose an easy transition from service systems expressed in other models to ours.

Each of the following sections will introduce a different use case. Each use case will be used differently in order to provide a richer evaluation.

## 5.2 Express mail delivery

Simplicity and ease of understanding were the top requirements for the first use case, as our goal is to test our model without losing too much time with complex services and lengthy descriptions. The first use case is, then, an express mail delivery service system that was originally modeled in order to illustrate the features of a service blueprint [39].

Based on the service blueprint of the express mail delivery use case, depicted in Figure 5.1, we may build an RDF file with that information represented as an LSS-USDL service system. This and all other RDF files in this chapter use the Turtle notation, due to its cleanness and high readability [12].

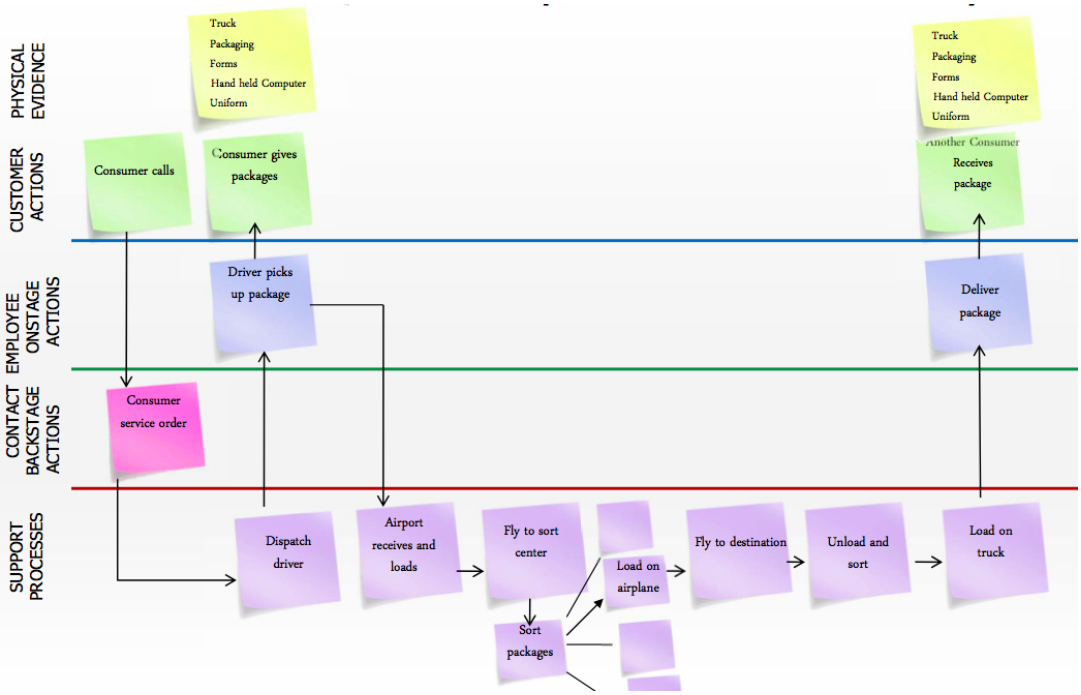


Figure 5.1: Express mail delivery original service blueprint [39]

The complete RDF code for this example can be found in `use cases/1 - Express Mail Delivery.ttl` of the model's code repository[1]. Listing 5.1 shows an extract of that code with the necessary data to describe the first interaction.

**Listing 5.1: Express mail delivery extract**

```
1 :Sender a lss-usdl:Role;
2   rdfs:label "Sender";
3   rdfs:comment "Customer who intends to send a mail to somebody else"
```

[1] https://github.com/rplopes/lss-usdl

```
      .
4
5 :SendMail a lss-usdl:Goal;
6  rdfs:label "Send mail";
7  rdfs:comment "Customer wants to send express mail to a specific
      destiny".
8
9 :SenderHome a lss-usdl:Location;
10  rdfs:label "Sender home";
11  lss-usdl:isLocatedIn :SenderRegion.
12
13 :SenderData a lss-usdl:KnowledgeResource;
14  rdfs:label "Sender data";
15  rdfs:comment "Name, address, etc. of sender".
16
17 :ReceiverData a lss-usdl:KnowledgeResource;
18  rdfs:label "Receiver data";
19  rdfs:comment "Name, address, etc. of receiver".
20
21 :MailDelivery a lss-usdl:Process;
22  rdfs:label "Mail delivery".
23
24 :CustomerCallsTime a time:ProperInterval;
25  time:intervalEquals :CustomerOrderIsRegisteredTime.
26
27 :CustomerCalls a lss-usdl:CustomerInteraction;
28  rdfs:label "Customer calls";
29  lss-usdl:performedBy :Sender;
30  lss-usdl:hasGoal :SendMail;
31  lss-usdl:hasTime [
32   a lss-usdl:Time;
33   lss-usdl:hasTemporalEntity :CustomerCallsTime
34  ];
35  lss-usdl:hasLocation :SenderHome;
36  lss-usdl:belongsToProcess :MailDelivery;
37  lss-usdl:receivesResource :SenderData;
38  lss-usdl:receivesResource :ReceiverData.
```

As we can see, the code looks clean and self-explanatory. For the interaction "Customer calls" we have to define the role "Sender", the goal "Send mail", the location "Sender home", the process "Mail delivery" and the two resources that the service system is receiving, "Sender data" and "Receiver data". We are also stating that this interaction is happening at the same time of interaction "Customer order is registered".

By looking at the complete source code in the repository we can see that the service system is fully described without losing information from the original description (considering that some slight adaptations were done for more cleanness). Furthermore, new data is added that enriches the context information.

In conclusion, this use case shows that this model seems to be a feasible solution for modeling simple service systems.

Now that we have the full service model in RDF, we can import it to the graphical editor. This should validate not only the correctness of that tool but also the graphical representation of the model.

In `http://lss-usdl-editor.herokuapp.com` we can import an existing service model by pressing the "Import from file" button and dragging the file to the new gray area (alternatively, we can click that area and select the desired file). The resulting service system for this use case can be viewed in the entry "Express Mail Delivery" of the deployed graphical editor[2]. Figure 5.2 shows a screenshot of that entry.

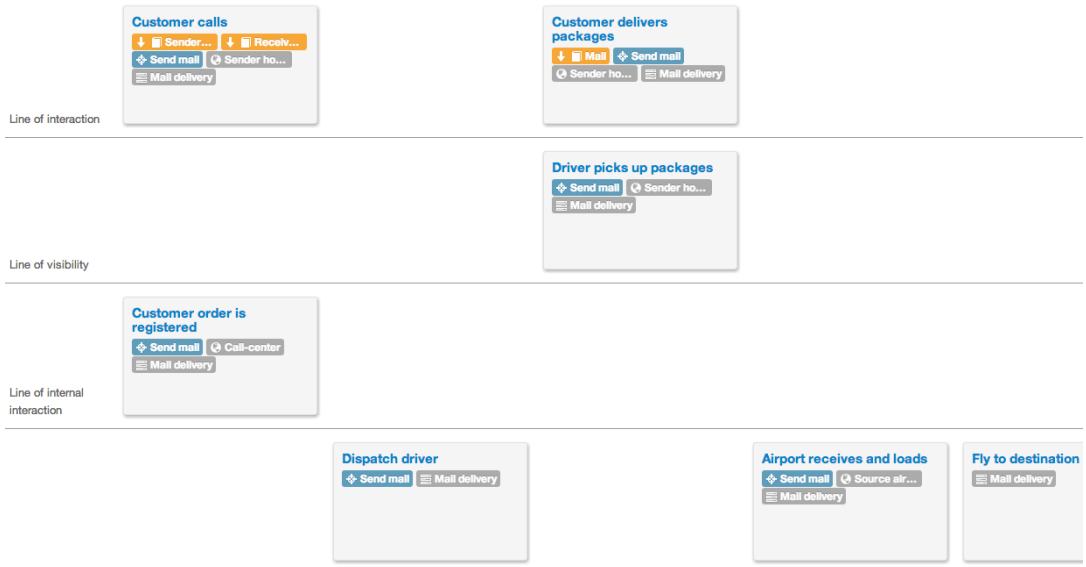


Figure 5.2: Extended service blueprint of the express mail delivery use case

As we can see, the visual result is very similar to the original service model in [39]. The additional information such as resources or locations is also visible.

This experiment allowed us to validate the correctness of the import tool and also proved that our model is not only a feasible solution for modeling service systems, but is also able to represent them in a graphical notation that is easy to understand and use to study the corresponding service system.

## 5.3 Bookstore kiosk

In our second use case we require a more complete service system, in order to evaluate the model's applicability to more complex, detailed scenarios. Because

[2] `http://lss-usdl-editor.herokuapp.com/service_systems/5`

the main field of this research is computer science, it is also desirable to look for use cases that do not alienate that field. Based on these premises, e are going to explore the example of a bookstore kiosk used by customers and employees to achieve their goals [37]. Figures 5.3 and 5.4 depict the original service blueprints for the customer interactions and employee interactions, respectively.

| Physical Evidence | Kiosk | Welcome Screen / Members Card | Member Profile Screen | Member Profile / Search Interface | Book Directions / Map, Coupon | Books | Books / Receipt |
|---|---|---|---|---|---|---|---|
| User Actions | Customer approaches bookstore kiosk | Customer swipes members card, logs in to kiosk | Customer considers suggestions | Customer searches for book | Customer prints book location map and coupon discounts | Customer walks to book locations and retrieves books | Customer discards one book and purchases the rest |
| Front Stage | | Welcome Screen | Members Profile Screen, Suggestions, Promotions | Kiosk search interface | Book location and tailored promotions | | Checkout and registers |
| Back Stage | | Kiosk software queries | Kiosk returns user profile, suggestions, promotions | Kiosk software queries | Kiosk returns book location, bundled promotional discounts | | Systems logs customer purchases |
| Support | | Customer Database | Marketing Database | Inventory DB / Location DB | | | Customer DB / Inventory DB |

Figure 5.3: Customer interactions in the bookstore kiosk original service blueprint [37]

In this use case we start by modeling the service system using the graphical editor. This enables us to evaluate the usefulness of the editing tool and how it compares to modeling by writing the RDF code. The resulting service system is shown in the entry "Bookstore kiosk" of the deployed graphical editor[3]. Figure 5.5 shows a screenshot of that entry.

As we can see, this service system is undoubtedly more complex than the one in the last use case. Using the graphical editor, however, greatly facilitated and accelerated its modeling. It is thus possible to conclude that this tool is not only a valid alternative to coding the RDF statements for people who don't know the RDF syntax, but is also generally faster and easier. However, as we discussed in the previous chapter, this tool makes some compromises between ease of use and data completion, so it is not necessarily the best alternative in

[3] `http://lss-usdl-editor.herokuapp.com/service_systems/6`

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Physical Evidence | | Employee Dashboard | Employee Dashboard | Map / Book | Employee Dashboard | Employee Dashboard | Map / Books |
| User Actions | Employee checks computer screen | Employee sees zombie book action alert | Employee clicks on alert | Employee retrieves book and replaces it | Employee sees restocking action alert | Employee clicks on alert | Employee retrieves copies from stock and replenishes shelf |
| Front Stage | | Employee dashboard alert section | Alert details link | Map with current book location | Employee dashboard alert section | Alert details link | Map with stock room and shelf book locations |
| Back Stage | | Enterprise Service Bus (ESB) integrates RFID data, identifies zombie book | Employee system queries | Kiosk returns zombie book's current location | ESB integrates RFID data, identifies book that needs restocking | Employee system queries | Kiosk returns book's location in stock room and target shelf location |
| Support | | Event Stream Processing (ESP), RFID Tracking tool, Inventory DB | Location DB | | ESP, RFID Tracking tool, Inventory DB | Location DB | |

Figure 5.4: Employee interactions in the bookstore kiosk original service blueprint [37]

all scenarios. Furthermore, since this is a prototype, some interaction elements could be improved, such as mass-assigning common data to many interactions, which currently have to be assigned individually.

With the service model in the editing tool it is possible to apply filters to get different views of the service system. As an example, we may apply a filter that shows the workflow for the customers interface (filtering by process "Customer interactions") and another filter that shows the workflow for the employees interface (filtering by process "Employee interactions"). These two filters show the adapted service models found in Figure 5 and 6 of [37], respectively.

The RDF code for the customers interface workflow can be found in `use cases/2 - Bookstore Kiosk.ttl` of the model's code repository. This file was created by applying the aforementioned filter and selecting the option "Export filtered data to LSS-USDL file". As we can see, the resulting file, which describes just a portion of the original service model, without many line breaks and comments, is larger than the file of the previous use case. By analyzing the resulting file we may conclude that we now have a valid LSS-USDL service model in a valid RDF notation. This means that this tool is indeed capable of building service models and providing them in the semantic notation we desired.

Compared to building service models directly with code, this tool has the disadvantage of outputting code that lacks the organization and comments that


Figure 5.5: Extended service blueprint of the bookstore kiosk use case

we saw in the previous use case. This has the potential of making the RDF code harder to understand by humans. However, when code readability is required, the effort of modeling the service system with this tool and after exporting reorganizing elements and commenting is still much less than the effort of writing all the code without this tool.

## 5.4 SaaS webapp

Since the first use case was a physical service system and the second was a hybrid between physical and digital service system, it is now relevant to explore a completely digital example. As such, a SaaS[4] is a very interesting use case to consider, since it is a growing trend in software with the single purpose of enabling the execution of a service system [87]. The third use case is, then, a SaaS webapp for sharing photos of travel destinations, that was modeled as a service blueprint during its service design phase [55]. Its original service blueprint is depicted in Figure 5.6.

We have previously validated the LSS-USDL model, the graphical editor tool and the import and export features of that tool. We now focus on validating the tool from translating Linked USDL into LSS-USDL and further testing our model in the process.

In order to do so, the service system in this use case has been modeled in Linked USDL. The RDF code of that service description can be found in `use cases/3 - klinkr - Linked USDL.ttl` of the repository. This file was

[4]Software as a Service

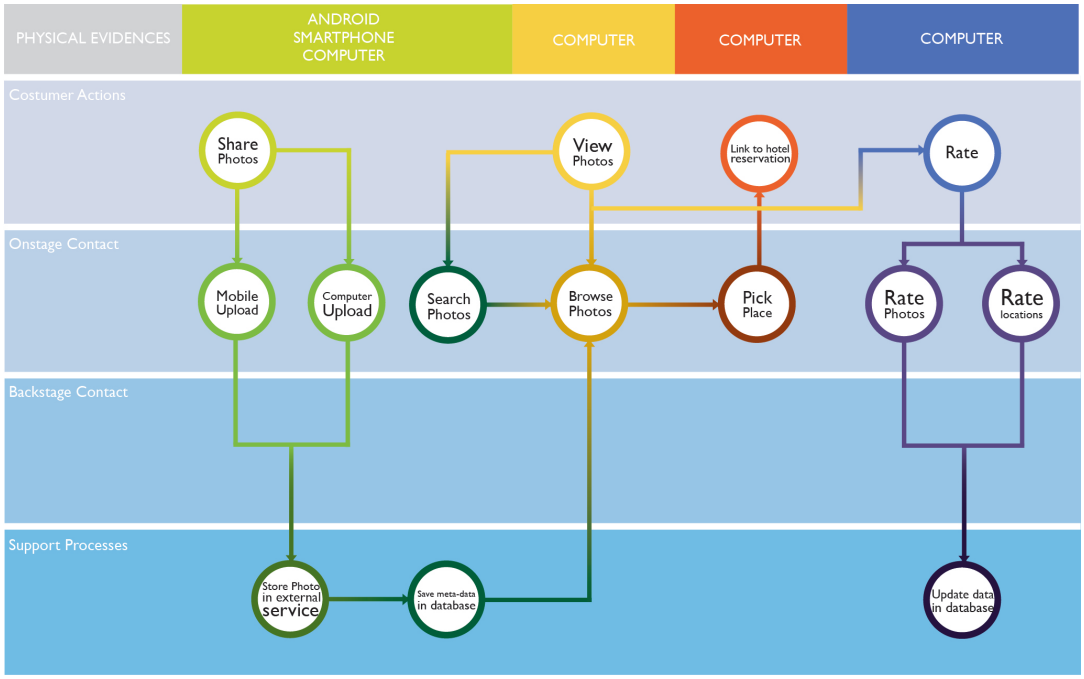


Figure 5.6: SaaS webapp klinkr original service blueprint [55]

then imported into the graphical editor tool, just like the file from the first use case. The tool can detect automatically if the file refers to an LSS-USDL or Linked USDL service system, so the process is exactly the same. The resulting service system is shown in the entry "klinkr" of the deployed graphical editor[5]. Figure 5.7 shows a screenshot of that entry. Note that this service system looks much more incomplete because it was based on a Linked USDL description.

As we can see, the amount of information that can be collected through a Linked USDL service description is not as complete as through an LSS-USDL one. Because Linked USDL is a service description language for customer interactions, all the service interactions that happen in the other areas of action are omitted. Moreover, Linked USDL does not include concepts such as goals or interaction locations.

This means that the resulting LSS-USDL service model will generally lack completeness. However, after this first automatic generation, it is possible to fill the remaining data. Hence, such a tool can be seen as a quick start for modeling service systems in LSS-USDL faster than if no conversion was done.

We can now proceed to export the generated service model into an LSS-USDL file. the corresponding RDF code can be found in `use cases/3 - klinkr.ttl` of the repository. After analyzing the generated code we may conclude that this tool does indeed enable the rapid creation of LSS-USDL service models based on existing Linked USDL service descriptions.

[5] `http://lss-usdl-editor.herokuapp.com/service_systems/7`

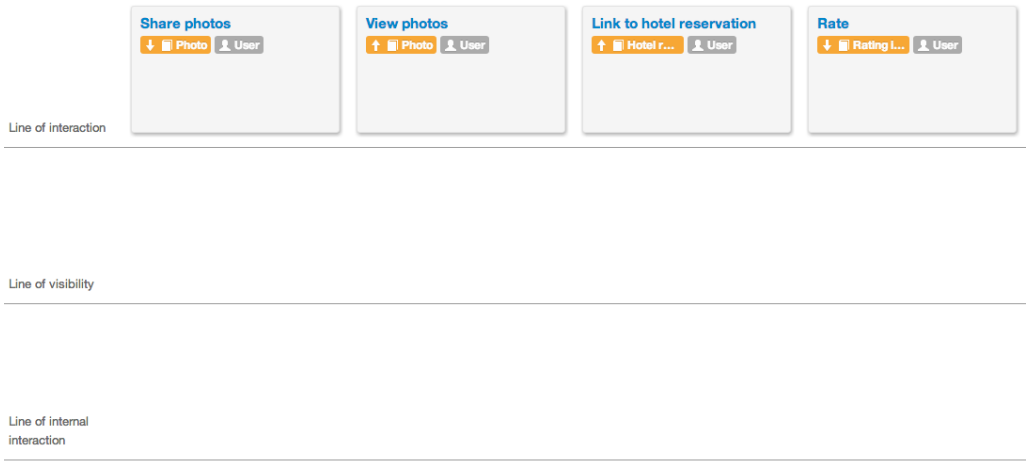


Figure 5.7: Extended service blueprint of the SaaS webapp use case

We may also export the service model into a Linked USDL file. This action is redundant because we are just generating a file with the same information of the original file we imported. However, it demonstrates the correctness of that tool. And since we already have the original file, we may compare the two and conclude that both import and export features are, indeed, working as expected.

## 5.5 Final remarks

The evaluation of our model was done through the use of three different use cases found in the literature. Those use cases were already modeled in a similar notation, the service blueprint, so that we could be able to compare our findings with the authors' work and thus achieve a higher level of confidence of our model's evaluation.

In the first use case we tested the model's ability to describe a simple service system and the graphical editor's ability to import it and display it in an easy to understand visual representation. In the second use case we tested the graphical editor's edition capabilities and its export functionality. Finally, in the third use case we tested the Linked USDL import and export tool, the graphical editor's service systems visual representation and its ability to export data.

In all these use cases we also had the opportunity to see the different service systems modeled in LSS-USDL. Comparing their service models with the original work found in the literature, the results are very encouraging and suggest that LSS-USDL is a viable option to model service systems.

The evaluation of the developed tools also yields positive results. The

migration between LSS-USDL RDF code, Linked USDL RDF code and the editor's data model, despite not being exhaustive, successfully translates the most relevant properties with minimal interaction for the user. The graphical editor, although it could have more elaborate user interaction mechanisms to deal with repetitive data input, is still a great contribution for accelerating the modeling of service systems and for visualizing them in an easier to understand graphical notation. Because these tools serve, themselves, as an evaluation for our model, we can conclude that they indeed prove the model's usefulness and capability for real world usage.

To sum up, after our evaluation this model shows positive signs towards service system modeling and real world usage. Our graphical editor proves it is possible to assign a clear visual representation to our model's service systems and also that it is possible to edit them without prior programming skills. The Linked USDL translator tool proves that this model can provide easy transitions from other models and is capable of generating different service descriptions.

# 6 Conclusions

This chapter presents the conclusions of our work and suggests possible future work to be done.

The first section briefly summarizes the contents of this thesis and the approach that we followed. The second section discusses our findings regarding the results we achieved. The third section describes how this work may benefit our society. Lastly, the fourth section suggests some future work possibilities to improve our model and its tool set.

## 6.1 Summary

Our main goal in this research work was to conceive a model for describing service systems as a whitebox in a computer readable notation, so that we could perform automation tasks such as business simulations, service analytics or automatic comparisons between different service systems.

In our state of the art study we discussed many models for describing service systems. Those models, however, did not produce computer readable information or did not present a whitebox approach, thus they were not capable of fulfilling our goal. However, their study enabled us to identify the most common service system attributes found in the literature.

Building upon the aforementioned attributes, the journalism interrogative pronouns and the service blueprint notation we were able to classify a service system as a flow of service interactions and create the 6-point interaction star depicted in Figure 3.1. This star model would be the core of our final model.

Our final model would also have to use Semantic Web technologies and include external vocabularies from the Linked Data Cloud. We chose to name it Linked Service Systems for USDL (LSS-USDL) because it builds upon the research efforts of the USDL research group and because it uses Linked Data to model the full length of the service system.

The result of combining the Semantic Web principles to the 6-point star model is our final model, depicted in Figure 3.4. This model is our proposal towards the fulfillment of our objectives. It is built in RDF and its code is freely available at `https://github.com/rplopes/lss-usdl` under a Creative Commons Attribution 3.0 Unported License[1].

In order to demonstrate the usefulness of our model in real world usage and also to provide additional value, two prototype software tools were also developed. The first tool was a graphical editor that provided easier to understand visual representations of our machine-readable model's service systems and faster modeling capabilities. The second tool was a converter to and from Linked USDL that provided easy migrations between different service description languages. These two tools were bundled in a webapp that was deployed online at `http://lss-usdl-editor.herokuapp.com`. Its code is also freely available at `https://github.com/rplopes/lss-usdl_editor` under the same license as the LSS-USDL model.

Lastly, we proceeded to the evaluation of this model. To do so, we studied three different use cases with three different approaches and discussed the results. Our findings are further explained in the next section.

## 6.2 Findings

In Section 5.5 we discussed our findings over the model's evaluation process. We found encouraging results that suggested that LSS-USDL is a viable option to model service systems. We also found that the tools yielded positive improvements to the model, despite being prototypes that lack completeness expected in a final product.

Most of the research efforts were guided towards providing a solid model and a serious evaluation with tools and use cases. Due to such evaluation the model's credibility is strengthened. This lets us hold bigger trust in its capabilities for modeling service systems. Thus, we may conclude that, although many improvements are possible upon further research, the proposed model is now a valid tool for organizations to model their service systems.

The fact that this model enables a machine-readable description of service systems brings many other interesting possibilities that could not be further explored due to scope constraints. This area certainly holds many new interesting findings that will not only further validate our model, but also bring unprecedented value to organizations and their stakeholders. Some of that value is further explained in the next section.

[1] `http://creativecommons.org/licenses/by/3.0`

## 6.3 Implication for society

Our ultimate goal when researching for a computer-readable service system model is to bring new value to organizations and their stakeholders and thus making a positive impact in society.

As we discussed in Chapter 1, the services sector is the most influential economic sector of modern society [8][56][85]. Yet, it is still the sector with less scientific understanding [21]. Our model is thus a proposal to enable a much needed better scientific understanding of this dominant economic sector. It is also a catalyst for the Semantic Web and Linked Data, which are yet available only for passive retrieval [68].

Having a tool that enables modeling service systems immediately brings better services documentation and standardization. This has the potential to increase productivity levels at organizations and also enables governments and foundations achieving higher levels of transparency.

However, providing a machine-readable notation for modeling service systems enables a wide range of more significant advancements to science and society. Although our approach did not focus on areas such as service analytics or simulations, the potential of our model is those areas is very interesting. This means that we are setting the building blocks that may enable, among other advantages:

- Standardized services catalogs for customers, through the automatic extraction of customer-relevant information, as demonstrated with the proposed Linked USDL conversion tool. These catalogs will greatly improve consumers' experience of browsing and choosing services.
- Better business decisions based on simulations though our model. This will result in better performance levels for businesses in the services sector.
- Better understanding of how services are operating and where they might need improvement based on service analytics. This enables a better understanding of customers' behavior and possible bottlenecks, which ultimately leads to added value for both parties.

These last improvements to society are now possible to achieve upon further research work. The future work that will enable such improvements is further explained in the next section.

## 6.4 Future work

As we discussed, our model presents a real improvement to businesses and to society in general. However, its biggest value lies within its potential as a

building block to future work.

Below we can find a list of proposed approaches for future work that builds upon our model:

- **Further service systems research:** A great effort was put into service systems research, in order to reach a solid proposal for a service model. However, and since this area spans across many different fields such as economics and management, there is room for improvement, specially if studied in a different context than computer science. Moreover, despite the positive results of our findings, it is advised to expect further research with this model to allow it to reach a better maturity level.

- **Better tool support:** The developed tools acted just as proof-of-concept prototypes to demonstrate the model's applicability and usefulness, and so they lack many interesting features and a maturity level that is expected of final products. It would then be interesting to develop more robust tools. Based on our findings, we would recommend exploring more data customization options and better user interfaces in an editor tool, and a more intelligent converter that could derive new values based on existing information, such as calculating the service price for Linked USDL based on the costs expressed in LSS-USDL.

- **Business intelligence:** Having service systems modeled in this notation, it is possible to apply business intelligence methods based in it. One possible approach would be to use the principles of System Dynamics [52]. This would allow developing simulation experiments [6] and service analytics [20].

To sum up, our findings show very interesting results for businesses and society, but the potential it creates for future research is even more impactful and exciting.

# Bibliography


[1] GoodRelations Language Reference. `http://www.heppnetz.de/ontologies/goodrelations/v1`. Accessed at 31/05/2013.

[2] Linked-usdl. `http://www.linked-usdl.org`. Accessed at 14/01/2013.

[3] Al-Debei, M. The design and engineering of innovative mobile data services: An ontological framework founded on business model thinking. *School of Information Systems, Computing and Mathematics* (2010).

[4] Alt, R., and Zimmermann, H. Preface: introduction to special section–business models. *Electronic Markets 11*, 1 (2001), 3–9.

[5] Alter, S. Service system fundamentals: Work system, value chain, and life cycle. *IBM Systems Journal 47*, 1 (2008), 71–85.

[6] An, L., and Jeng, J.-J. On developing system dynamics model for business process simulation. In *Simulation Conference, 2005 Proceedings of the Winter* (2005), IEEE, pp. 10–pp.

[7] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. Dbpedia: A nucleus for a web of open data. In *The semantic web*. Springer, 2007, pp. 722–735.

[8] Barros, A., Kylau, U., and Oberle, D. Unified service description language 3.0 (usdl) overview, 2011.

[9] Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., et al. Semantic web services framework (swsf) overview. *World Wide Web Consortium, Member Submission SUBM-SWSF-20050909* (2005).

[10] Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., et al. Semantic web services language (swsl). *W3C Member submission 9* (2005).

[11] Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., et al. Semantic web services ontology (swso). *Member submission, W3C* (2005).

[12] Beckett, D., and Berners-Lee, T. Turtle-terse rdf triple language. *W3C Team Submission 14* (2008).

[13] Berners-Lee, T. Linked Data - Design Issues, 2006.

[14] Bizer, C., Heath, T., and Berners-Lee, T. Linked data-the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS) 5*, 3 (2009), 1–22.

[15] Blair, A., Debenham, J., and Edwards, J. Requirements analysis for intelligent decision support systems. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on* (1994), IEEE, pp. 482–486.

[16] Booth, D., and Liu, C. Web services description language (wsdl) version 2.0 part 0: Primer. *World Wide Web Consortium (W3C), June 26* (2007), W3C.

[17] Cardoso, J., Barros, A., May, N., and Kylau, U. Towards a unified service description language for the internet of services: Requirements and first developments. In *Services Computing (SCC), 2010 IEEE International Conference on* (2010), IEEE, pp. 602–609.

[18] Cardoso, J., Pedrinaci, C., Leidig, T., Rupino, P., and De Leenheer, P. Open semantic service networks. In *International Symposium on Services Science (ISSS'12), Leipzig, Germany* (2012).

[19] Cardoso, J., Winkler, M., and Voigt, K. A service description language for the internet of services. In *Proceedings of ISSS* (2009).

[20] Chen, Y., Spohrer, J., and Lelescu, A. Three factors to sustainable service system excellence: A case study of service systems. In *Services Computing, 2008. SCC'08. IEEE International Conference on* (2008), vol. 2, IEEE, pp. 119–126.

[21] Chesbrough, H., and Spohrer, J. A research manifesto for services science. *Communications of the ACM 49*, 7 (2006), 35–40.

[22] Chinnici, R., Moreau, J., Ryman, A., and Weerawarana, S. Web services description language (wsdl) version 2.0 part 1: Core language. *W3C Recommendation 26* (2007).

[23] Christensen, E., and Meredith, G. Web services description language (wsdl) 1.1.

[24] De Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Oren, E., et al. Web service modeling ontology (wsmo). *Interface 5* (2006), 1.

[25] Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., and Horrocks, I. The semantic web: the roles of xml and rdf. *Internet Computing, IEEE 4*, 5 (sep/oct 2000), 63 –73.

[26] Dhanesha, K., Hartman, A., and Jain, A. A model for designing generic services. In *Services Computing, 2009. SCC'09. IEEE International Conference on* (2009), IEEE, pp. 435–442.

[27] Dumas, M., O'Sullivan, J., Heravizadeh, M., Edmond, D., and ter Hofstede, A. Towards a semantic framework for service description.

[28] Faber, E., Ballon, P., Bouwman, H., Haaker, T., Rietkerk, O., and Steen, M. Designing business models for mobile ict services. In *Workshop on concepts, metrics & visualization, at the 16th Bled Electronic Commerce Conference eTransformation, Bled, Slovenia* (2003).

[29] Fensel, D., and Bussler, C. The web service modeling framework wsmf. *Electronic Commerce Research and Applications 1*, 2 (2002), 113–137.

[30] Fensel, D., Fischer, F., Kopecký, J., Krummenacher, R., Lambert, D., and Vitvar, T. Wsmo-lite: Lightweight semantic descriptions for services on the web. *W3C Member Submission 23* (2010).

[31] Ferrario, R., and Guarino, N. Towards an ontological foundation for services science. *Future Internet–FIS 2008* (2009), 152–169.

[32] Ferrario, R., Guarino, N., Janiesch, C., Kiemes, T., Oberle, D., and Probst, F. Towards an ontological foundation of services science: The general service model. *Wirtschaftsinformatik, Zurich, Switzerland February* (2011), 16–18.

[33] Fielt, E. Alternative business model canvasses: A Partnering Canvas example, 2010.

[34] Fielt, E. An Extended Business Model Canvas for Co-Creation and Partnering, 2010.

[35] Fielt, E. To what extent is the Business Model Canvas constraining? A Co-Creation Canvas example, 2010.

[36] Fliess, S., and Kleinaltenkamp, M. Blueprinting the service company: Managing service processes efficiently. *Journal of Business Research 57*, 4 (2004), 392–404.

[37] Glushko, R. Seven contexts for service system design. *Handbook of service science* (2010), 219–249.

[38] Glushko, R. J., and Tabas, L. Designing service systems by bridging the "front stage" and "back stage". *Information Systems and e-Business Management 7* (2009), 407–427.

[39] Gremler, D. D. Service blueprinting: Designing service from the customer's point of view. In *Phonak Practice Development Conference* (2011).

[40] Haukland, T. jeoquery: Browse geonames hierarchy methods. `http://tompi.github.io/jeoquery/earth.html`. Accessed at 21/03/2013.

[41] Heath, T., and Bizer, C. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology 1*, 1 (2011), 1–136.

[42] Hepp, M. Goodrelations: An ontology for describing products and services offers on the web. In *Knowledge Engineering: Practice and Patterns*. Springer, 2008, pp. 329–346.

[43] Hill, T. On goods and services. *Review of Income and Wealth 23*, 4 (1977), 315–38.

[44] Hobbs, J. R., and Pan, F. Time ontology in owl. *W3C working draft 27* (2006).

[45] Hong, J., and Bae, D. Software modeling and analysis using a hierarchical object-oriented petri net. *Information Sciences 130*, 1 (2000), 133–164.

[46] Isaac, A., and Summers, E. Skos simple knowledge organization system primer. w3c working group note. *World Wide Web Consortium* (2009).

[47] Kaner, M., and Karni, R. Design of service systems using a knowledge-based approach. *Knowledge and Process Management 14*, 4 (2007), 260–274.

[48] Karni, R., and Kaner, M. Teaching innovative conceptual design of systems in the service sector. *Technological Forecasting and Social Change 64*, 2 (2000), 225–240.

[49] Karni, R., and Kaner, M. An engineering tool for the conceptual design of service systems. *Advances in Services Innovations* (2007), 65–83.

[50] Kinderen, S. D., and Gordijn, J. e3service: An ontological approach for deriving multi-supplier it-service bundles from consumer needs. In *Proceedings of the 41st annual Hawaii international conference on system sciences* (2008), p. 318.

[51] Kinderen, S. D., and Gordijn, J. Reasoning about substitute choices and preference ordering in e-services. In *Advanced Information Systems Engineering* (2008), Springer, pp. 390–404.

[52] Kirkwood, C. W. System dynamics methods. *A quick introduction* (2001).

[53] Kruchten, P. *The rational unified process: an introduction*. Addison-Wesley Professional, 2004.

[54] Lausen, H., and Farrell, J. Semantic annotations for wsdl and xml schema. *W3C recommendation, W3C* (2007).

[55] Lopes, R., Duro, J., Chicória, R., Mateus, A., and Raposeira, P. klinkr - using concord: A customer centered service design approach. 2012.

[56] Luczak, H., Gill, C., and Sander, B. Architecture for service engineering — the design and development of industrial service work. In *Advances in Services Innovations*, D. Spath and K.-P. Fähnrich, Eds. Springer Berlin Heidelberg, 2007, pp. 47–63.

[57] Maglio, P., Srinivasan, S., Kreulen, J., and Spohrer, J. Service systems, service scientists, ssme, and innovation. *Communications of the ACM 49*, 7 (2006), 81–85.

[58] Maglio, P., Vargo, S., Caswell, N., and Spohrer, J. The service system is the basic abstraction of service science. *Information Systems and e-business Management 7*, 4 (2009), 395–406.

[59] Magretta, J. Why business models matter. *Harvard business review 80*, 5 (2002), 86–93.

[60] Maier, M., Emery, D., and Hilliard, R. Software architecture: Introducing ieee standard 1471. *Computer 34*, 4 (2001), 107–109.

[61] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. Owl-s: Semantic markup for web services. *W3C Member submission 22* (2004), 2007–04.

[62] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. Wonderweb deliverable d18, ontology library (final). *ICT Project 33052* (2003).

[63] McGuinness, D., Van Harmelen, F., et al. Owl web ontology language overview. *W3C recommendation 10*, 2004-03 (2004), 10.

[64] Miles, A., Matthews, B., Wilson, M., and Brickley, D. Skos core: simple knowledge organisation for the web. In *International Conference on Dublin Core and Metadata Applications* (2005), pp. pp–3.

[65] Mora, M., O'Connor, R., Raisinghani, M., Macías-Luévano, J., and Gelman, O. An it service engineering and management framework (its-emf). *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET) 2*, 2 (2011), 1–15.

[66] Mora, M., Raisinghani, M., Gelman, O., and Sicilia, M. Onto-servsys: A service system ontology. *The Science of Service Systems* (2011), 151–173.

[67] Natschläger, C. Towards a bpmn 2.0 ontology. In *Business Process Model and Notation*. Springer, 2011, pp. 1–15.

[68] Norton, B., Krummenacher, R., Marte, A., and Fensel, D. Dynamic linked data via linked open services. In *Workshop on Linked Data in the Future Internet at the Future Internet Assembly* (2010), pp. 1–10.

[69] Oberle, D., Barros, A., Kylau, U., and Heinzl, S. A unified description language for human to automated services. *Information Systems 38*, 1 (2013), 155 – 181.

[70] Oberle, D., Bhatti, N., Brockmans, S., Niemann, M., and Janiesch, C. Countering service information challenges in the internet of services. *Business & Information Systems Engineering 1*, 5 (2009), 370–390.

[71] OGC. *The Official Introduction to the ITIL Service Lifecycle*. ITIL Series. Stationery Office, 2007.

[72] OMG. Introduction to OMG's Unified Modeling Language (UML), 2012.

[73] Osterwalder, A. The business model ontology: a proposition in a design science approach. *Institut d'Informatique et Organisation. Lausanne, Switzerland, University of Lausanne, Ecole des Hautes Etudes Commerciales HEC 173* (2004).

[74] Osterwalder, A., and Pigneur, Y. *Business model generation: a handbook for visionaries, game changers, and challengers*. Wiley, 2010.

[75] Osterwalder, A., Pigneur, Y., and Tucci, C. Clarifying business models: Origins, present, and future of the concept. *Communications of the association for Information Systems 16*, 1 (2005), 1–25.

[76] Pedrinaci, C., and Domingue, J. Toward the next wave of services: linked services for the web of data. *Journal of Universal Computer Science 16*, 13 (2010), 1694–1719.

[77] Petrovic, O., Kittl, C., and Teksten, R. Developing business models for ebusiness. *Available at SSRN 1658505* (2001).

[78] Poels, G. The resource-service-system model for service science. In *Advances in Conceptual Modeling–Applications and Challenges*. Springer, 2010, pp. 117–126.

[79] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D., et al. Web service modeling ontology. *Applied Ontology 1*, 1 (2005), 77–106.

[80] Selic, B. UML 2.0: Exploiting Abstration and Automation, 2004.

[81] Shostack, G. L. Designing services that deliver. *Harvard Business Review 62*, 1 (1984), 133 – 139.

[82] Söderström, E., Andersson, B., Johannesson, P., Perjons, E., and Wangler, B. Towards a framework for comparing process modelling languages. In *Advanced Information Systems Engineering* (2006), Springer, pp. 600–611.

[83] Speiser, S., and Harth, A. Towards linked data services. In *Proceedings of the 9th International Semantic Web Conference (ISWC)* (2010).

[84] Spohrer, J., and Maglio, P. Service science: Toward a smarter planet. *Introduction to service engineering* (2009), 3–30.

[85] Spohrer, J., Maglio, P., Bailey, J., and Gruhl, D. Steps toward a science of service systems. *Computer 40*, 1 (2007), 71–77.

[86] Timmers, P. Business models for electronic markets. *Electronic markets 8*, 2 (1998), 3–8.

[87] Turner, M., Budgen, D., and Brereton, P. Turning software into a service. *Computer. 36*, 10 (2003), 38–44.

[88] Vatant, B., and Wick, M. Geonames ontology. `http://www.geonames.org/ontology`, 2012. Accessed at 31/05/2013.

[89] W3C. Web services glossary, 2004.

[90] Zachman, J. Enterprise architecture: The issue of the century. *Database Programming and Design 10*, 3 (1997), 44–53.

[91] Zachman, J. John Zachman's Concise Definition of The Zachman Framework™, 2008.

[92] Zolnowski, A., Semmann, M., and Böhmann, T. Introducing a co-creation perspective to service business models. In *Enterprise Modelling and Information Systems Architectures (EMISA 2011)* (2011), p. 243.

[93] Zolnowski, A., Semmann, M., and Böhmann, T. Metamodels for representing service business models.

# Appendices

# A
# Schedule

In the end of the first semester of this thesis a schedule was elaborated to define the work that should be done in the second semester. The following is the list of tasks that were identified and their temporal allocation. This schedule may also be visualized in the Gannt diagram in Figure A.1.

- Development of the service model using RDF: February
- Creation of model instances in RDF and model evaluation: February and March
- Script that generates Linked USDL code: March
- Evaluation of the Linked USDL generator script: April
- Software tool to act as an abstraction layer for the service model users: April and May
- Creation of model instances using the tool and tool evaluation: May
- General improvements and late work: May and June
- Service model and tool documentation: June
- Preparation for the final presentation: July
- Final presentation: July

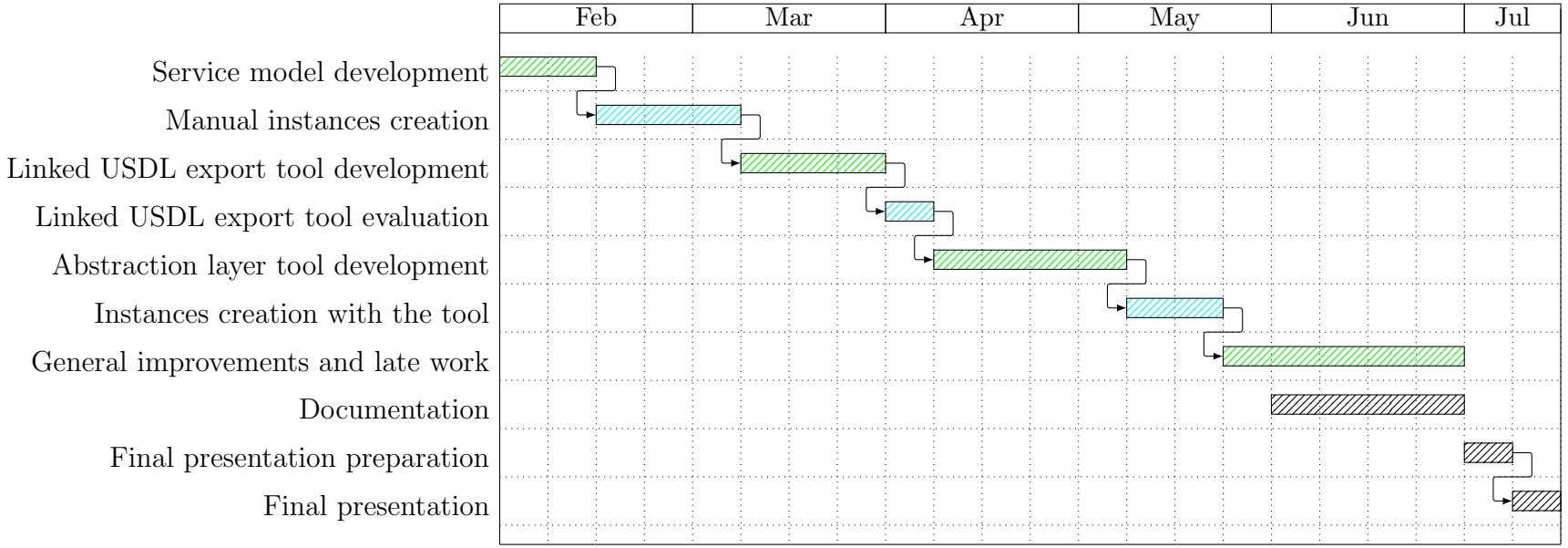


Figure A.1: Gantt diagram of the project planning. Development tasks are shown in green, evaluation tasks are shown in cyan and generic tasks are shown in gray.

The actual temporal execution of these tasks followed some adaptations, but it was always possible to meet the proposed deadlines.

Firstly, after some consideration, the creation of the visual abstraction tool was considered to have a higher priority than the Linked USDL mapping tool, not only because it acted as a stronger proof of concept, but also because its graphical representation of service systems was needed to get a better understanding of possible flaws of the model. Therefore, the development and evaluation of the abstraction tool was switched with the development and evaluation of the Linked USDL mapping tool.

Furthermore, the development of the abstraction tool was divided into two parts: the graphical editor and the import/export feature. The evaluation was also divided into two parts because there was a need of evaluating the graphical editor before moving on to the import/export feature.

Lastly, in the original schedule a week was expected in the Linked USDL converter tool to set up the required tools for parsing RDF files in Ruby and exporting data to files. Because the development of that tool was changed to be after the development of the graphical editor, and because the graphical editor also had a feature to import and export RDF code, there was a planned week that changed from the Linked USDL converter tool to the import/export feature of the graphical editor.

Regarding the proposed tasks, the only relevant change was in the Linked USDL converter tool. In the original schedule it was planned to be a tool for exporting LSS-USDL models into Linked USDL, but after further considerations that task was expanded to also include the feature of creating LSS-USDL models based on Linked USDL code.

The resulting schedule, which corresponds to the actual execution of the tasks, is shown in Figure A.2.

Figure A.2: Gantt diagram of the actual execution of the tasks. Development tasks are shown in green, evaluation tasks are shown in cyan and generic tasks are shown in gray.

# B

# Project documentation

This appendix shows the documentation of the two code repositories created in the scope of this thesis. The first section is the documentation of the LSS-USDL model's repository[1] and the second section is the documentation of the graphical editor's repository[2]. The links in the documentation are depicted here as footnotes.

## B.1 LSS-USDL



Linked Service Systems for USDL (LSS-USDL) is an ontology for modeling service systems in RDF. This brings many advantages to organizations that make use of this ontology:

- The resulting service models may be used as documentation for the service operations or to generate service descriptions for various stakeholders
- A freely available and complete service description presents a solid evidence of an organization's effort towards transparency
- After modeling a complete service system it is possible to identify previously unknown bottlenecks and fail points and to study how to overcome them
- After all operations of a service system are identified it is possible to execute automation tasks that can greatly reduce costs
- It is also possible to run simulations based on the service model, which aid managerial and operational decisions

[1] https://github.com/rplopes/lss-usdl
[2] https://github.com/rplopes/lss-usdl_editor
[3] http://creativecommons.org/licenses/by/3.0/

- Because this ontology uses Semantic Web tools and integrates with the Linked Data Cloud, a strong data integration is also ensured
- Service models may create custom service descriptions aimed at customers that could be used in a generic online services marketplace

## B.1.1 Model Explanation

### B.1.1.1 6-Point Interaction Star

A service system can be expressed as the flow of its interactions. Service interactions take place when any actor interacts with the service system. We use the journalism interrogative pronouns (who, how, why, where, when, what) to give a better context to service interactions.

The elements that represent each of those answers are, respectively, the role of the actor that is interacting, the process that describes how the service works, the goal behind why such an interaction is taking place, the location, the time and the resources that enter or leave the service system during that interaction.

This contextualization of service interactions is the core of LSS-USDL. It is called 6-point interaction star and can be viewed in the images directory of the project.

### B.1.1.2 Extending Interactions and Resources

For those familiarized with the service blueprint, it is also possible to further describe an interaction based on its area of action. This means that an interaction in LSS-USDL can be classified as customer interaction, onstage interaction, backstage interaction and support interaction.

Resources may also be further described according to their nature. A resource in LSS-USDL can be classified as a physical resource (such as a package), a knowledge resource (such as a customer's profile) or a financial resource (such as a payment).

These classifications are depicted in the images directory of the project. Note, however, that their use is not mandatory, and because they are defined as SKOS concept schemes, you can replace them with your own.

### B.1.1.3 Ontology Elements

A graph of the full ontology may be viewed in the images directory of the project. This subsection explains its elements and their relations.

`ServiceSystem` is the entity that represents the service system that is being modeled. A `ServiceSystem` is connected to an `Interaction`, which represents service interactions, through the property `hasInteraction`.

Interaction is connected to the following elements:

- Role, which describes the role of the actor interacting with the service, through the property isPerformedBy
- Process, which describes the process of the interaction, through the property belongsToProcess
- Goal, which describes the goal behind the service interaction, through the property hasGoal
- Location, which describes the interaction's location, through the property hasLocation
- Time, which holds the interaction's temporal data, through the property hasTime
- Resource, which describes the resources interacting with the service system at that moment, through the properties receivesResource (if it's an input from outside), createsResource (if it was created internally), consumesResource (if it was consumed internally) and returnsResource (if it's an output to outside)

These core elements can also be connected to other elements of the Linked Data Cloud, in order to provide richer information and better disambiguations.

A Role can also be connected to a BusinessEntity of the ontology GoodRelations[4] through the property hasBusinessEntity. This makes it possible to explain which company or other stakeholder is responsible for that role.

A Process can be connected to a Process of the BPMN 2.0[5] ontology through the property hasBPMN. This enables linking the service model with the previously defined process model.

A Location can have a broader location through the property isLocatedIn. This enables a hierarchy that lest us express knowledge such as "Room A is located in Building 1, therefore an interaction happening in Room A is also happening in Building 1". A Location can also be connected to a Feature of the Geonames[6] ontology through the property isLoationFrom to assign it an unambiguous geographical meaning.

Time is connected to a TemporalEntity of the OWL-Time[7] ontology through the property hasTemporalEnity. This enables a very rich temporal

[4] http://www.heppnetz.de/ontologies/goodrelations/v1
[5] http://www.scch.at/en/Page56-8330.aspx
[6] http://www.geonames.org/ontology/documentation.html
[7] http://www.w3.org/TR/owl-time

description, such as the duration of the interaction, its date or its temporal relation to other interactions.

A `Resource` is connected to a `QuantitativeValue` of the ontology GoodRelations through the property `hasQuantitativeValue` to assign it values such as quantities, units, etc. It is also connected to a DBpedia[8] `Resource` through the property `hasDBpediaResource` to get an unambiguous semantic meaning.

### B.1.2 Getting Started Tutorial

A service system modeled in LSS-USDL is represented by RDF statements. We will use the Turtle notation because it's cleaner and easy to read and edit.

The first step is to create a file that will hold the service model. For an express mail delivery service system we may create the file maildelivery.ttl. These are the RDF prefixes we need for this tutorial (you may add others and remove any that you might not use):

```
1 @prefix : <http://genssiz.org/lss-usdl/expressmail#> . # this is the
      prefix for our example, change the URL to match yours
2
3 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
5 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
6 @prefix lss-usdl: <http://genssiz.dei.uc.pt/lss-usdl#> .
```

The first element to add is the service system. We can use RDF properties to give any element a label and a comment:

```
1 # Add the service system
2 :ExpressMailDelivery a lss-usdl:ServiceSystem;
3   rdfs:label "Express Mail Delivery";
4   rdfs:comment "A service system for delivering express mails".
```

Note that, if we were using the RDF/XML notation, the same data would look like the following:

```
1 <lss-usdl:ServiceSystem rdf:about="#ExpressMailDelivery">
2   <rdfs:label>Express Mail Delivery</rdfs:label>
3   <rdfs:comment>A service system for delivering express mails</rdfs:
      comment>
```

[8] http://dbpedia.org/

```
4 </lss-usdl:ServiceSystem>
```

Now we can start adding interactions. Every time we add an interaction, it should be added to the service system's list of interactions:

```
1 # Change the service system
2 :ExpressMailDelivery a lss-usdl:ServiceSystem;
3   rdfs:label "Express Mail Delivery";
4   rdfs:comment "A service system for delivering express mails";
5   lss-usdl:hasInteraction :CustomerCalls .
6
7 # Add the interaction
8 :CustomerCalls a lss-usdl:CustomerInteraction;
9   rdfs:label "Customer calls" .
```

That interaction still has no information, so the next step is to provide more useful context data. For new entities we need to create them and make the interaction point to them:

```
1 # Change the interaction
2 :CustomerCalls a lss-usdl:CustomerInteraction;
3   rdfs:label "Customer calls";
4   lss-usdl:hasGoal :SendMail;
5   lss-usdl:isPerformedBy :Sender;
6   lss-usdl:hasLocation :SenderHome .
7
8 # Add the goal
9 :SendMail a lss-usdl:Goal;
10  rdfs:label "Send mail" .
11
12 # Add the role
13 :Sender a lss-usdl:Role;
14  rdfs:label "Sender" .
15
16 # Add the location
17 :SenderHome a lss-usdl:Location;
18  rdfs:label "Sender's home" .
```

For entities that have already been created we only need to point at them. So if we create a new interaction performed by an actor with the role "sender" we only need to point to it:

```
1 # Change the service system
2 :ExpressMailDelivery a lss-usdl:ServiceSystem;
```

```
 3   rdfs:label "Express Mail Delivery";
 4   rdfs:comment "A service system for delivering express mails";
 5   lss-usdl:hasInteraction :CustomerCalls,
 6     :CustomerDeliversPackages .
 7
 8 # Add the interaction
 9 :CustomerDeliversPackages a lss-usdl:CustomerInteraction;
10   rdfs:label "Customer delivers packages";
11   lss-usdl:isPerformedBy :Sender .
```

The code we have so far should now look like this:

```
 1 @prefix : <http://genssiz.org/lss-usdl/expressmail#> .
 2
 3 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
 4 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
 5 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
 6 @prefix lss-usdl: <http://genssiz.dei.uc.pt/lss-usdl#> .
 7
 8 :ExpressMailDelivery a lss-usdl:ServiceSystem;
 9   rdfs:label "Express Mail Delivery";
10   rdfs:comment "A service system for delivering express mails";
11   lss-usdl:hasInteraction :CustomerCalls,
12     :CustomerDeliversPackages .
13
14 :CustomerCalls a lss-usdl:CustomerInteraction;
15   rdfs:label "Customer calls";
16   lss-usdl:hasGoal :SendMail;
17   lss-usdl:isPerformedBy :Sender;
18   lss-usdl:hasLocation :SenderHome .
19
20 :CustomerDeliversPackages a lss-usdl:CustomerInteraction;
21   rdfs:label "Customer delivers packages";
22   lss-usdl:isPerformedBy :Sender .
23
24 :SendMail a lss-usdl:Goal;
25   rdfs:label "Send mail" .
26
27 :Sender a lss-usdl:Role;
28   rdfs:label "Sender" .
29
30 :SenderHome a lss-usdl:Location;
31   rdfs:label "Sender's home" .
```

This was just a short getting start guide to explain how to use the LSS-USDL ontology to model service systems. The full code of this express mail delivery example is available in the use cases directory of the project, among with other examples to help you understand how to use this ontology.

### B.1.3 Useful links

- LSS-USDL Editor[9]: Open source repository of the LSS-USDL graphical editor.
- USDL Incubator Group[10]: LSS-USDL is part of the research for service systems by the USDL research group.
- Linked USDL[11]: Similar project, focusing on service descriptions for customers. The third use case found in LSS-USDL's repository shows a service system modeled both in LSS-USDL and Linked USDL.
- Linked USDL core[12]: Repository for the core module of Linked USDL. The other modules may be found under the same Github profile.
- Semantic Web[13]: Technologies such as RDF are a core component of LSS-USDL.

## B.2 LSS-USDL Editor



Linked Service Systems for USDL (LSS-USDL)[15] is an ontology for modeling service systems in RDF. This is a graphical editor for LSS-USDL instances developed in Ruby on Rails. Its goal is to provide an abstranction to model service systems without having to edit RDF code and also to present a visual representation of modeled serivce systems. A deployed version for demonstration purposes is available at http://lss-usdl-editor.herokuapp.com.

### B.2.1 How to set up

This webapp was developed in Ruby, using the framework Ruby on Rails. So if you don't have Ruby installed in your computer, you should install it. Follow this link[16] for all the information on how to install Ruby on your platform.

[9] https://github.com/rplopes/lss-usdl_editor
[10] http://www.w3.org/2005/Incubator/usdl
[11] http://www.linked-usdl.org/
[12] https://github.com/linked-usdl/usdl-core
[13] http://semanticweb.org/wiki/Main_Page
[14] http://creativecommons.org/licenses/by/3.0/
[15] https://github.com/rplopes/lss-usdl
[16] http://www.ruby-lang.org/en/downloads/

This application is versioned in Git, so if you don't have Git installed, you should also install it. Follow this link[17] for all the information on how to install Git on your platform.

If you want to run this application on your computer and not on a production server, you also need to install the database SQLite, to store the information even when you exit the editor. Follow this link[18] for the installation instructions. If you are configuring a production environment, then you should install PostgreSQL[19] instead.

The first step to set up this app on your computer is to clone the Git repository. To do so, type the following in your terminal:

```
1 git clone git@github.com:rplopes/lss-usdl_editor.git
```

This will copy all the necessary files to the directory `lss-usdl_editor`. To go to that directory:

```
1 cd lss-usdl_editor
```

Now you need to install the required dependencies. If you don't have the Bundler gem installed:

```
1 gem install bundler
```

Now to install all other required gems just type:

```
1 bundle install
```

In order to save data we need to have a database and the right schema. We use the SQLite database because it is great for lightweight usage. If you are setting up a production environment, then the database is PostgreSQL. The required commands to generate the database and schema are:

```
1 rake db:create
2 rake db:migrate
```

[17] http://git-scm.com/
[18] http://www.sqlite.org/
[19] http://www.postgresql.org/

Now everything is set. To start the application type:

```
1 rails server
```

### B.2.2 Useful links

- Linked Service Systems for USDL[20]: Open source repository of the LSS-USDL model.
- USDL Incubator Group[21]: LSS-USDL is part of the research for service systems by the USDL research group.
- Linked USDL[22]: Similar project, focusing on service descriptions for customers. The third use case found in LSS-USDL's repository shows a service system modeled both in LSS-USDL and Linked USDL.
- Linked USDL core[23]: Repository for the core module of Linked USDL. The other modules may be found under the same Github profile.
- Semantic Web[24]: Technologies such as RDF are a core component of LSS-USDL.

[20] https://github.com/rplopes/lss-usdl
[21] http://www.w3.org/2005/Incubator/usdl
[22] http://www.linked-usdl.org/
[23] https://github.com/linked-usdl/usdl-core
[24] http://semanticweb.org/wiki/Main_Page