A Service Register as an Information Intermediary

Towards a structured design approach for service registers

Assigner

Ordina ICT B.V.
Business Unit ICT Management & Consultancy

Supervisors

First supervisor
dr. A. B. J. M. Wijnhoven
Information Systems and Change Management, University of Twente

Second supervisor
dr. L. Ferreira Pires
Software Engineering, University of Twente

Ordina supervisor
Ing. W. van Pruijssen
Solution Architect, Ordina ICT

Student

Mr. A.R. van Oostrum
Student Business Information Technology
a.r.vanoostrum@gmail.com

Abstract

Motivation – need for another service register design approach

Service Oriented Architecture and all forthcoming concepts are popular topics in both business and current research. A possible (and often required) component of the SOA governance is the service register, a 'catalogue' of services that exists in a SOA architecture. The service register is used to store the specifications of the 'real' services in a central place so they can be searched and found by other users.

At the moment, the design of service registers is primarily focused on the inclusion of the service register in the system architecture, and is therefore often focused on technical data. However, there is a need for specification information for other types of users. In order to fulfil this need, the service register should be differently designed.

Approach – Information Intermediary design methodology

This research uses the design methodology for Information Intermediaries to the design of service registers in order to apply a structured approach to service register design. This methodology takes various stakeholder views into account, and identifies the different information types needed by the stakeholders.

The methodology defines layers and aspects to create a design space with elements that are to be considered in the design. This research applies this design space to the service register. The result is an architectural design that indicates the different views, needs, processes and means necessary to fulfil the needs. However, this design is too large for implementation in order to validate the methodology.

Therefore, a small part of the service register (the interaction with the end-user) is further specified and constructed in a prototype as a proof of concept.

Results

The research provides a structured approach to service register design and discusses the steps for creating the final design. As there is no real case scenario used for the construction, the final design is broad and meant as a reference model. It explicitly leaves several design choices open to be made in a real case design.

This approach to service register design is a different way of designing a service register. The approach focuses on the fulfilment of the information needs of the end users and determines various information types that satisfy different end-users. The design concentrates on the match of the search request of the end-user and the content that is available in the service register whereas other design approaches concentrate on the technical interaction with other applications.

The contribution of this research is twofold. Firstly, it adds another design approach for service registers to the current research knowledge base. This thesis can be seen as a starting point for further research that focuses on providing the right information to the end-user, in a way similar to the delivery of an information good by an Information Intermediary. This comprises a focus on the *content* that is stored about a service (i.e. the service specification), the way of content

delivery to the user using *soft features* and –when relevant- the way of revenue generation using the service register

Secondly, the information needs of the users are centrally placed in the design and exploration of the service register. This addresses the key problem of current service registers as encountered by organizations, that these are not able to satisfy the information needs of end-users.

To evaluate our design approach and design, we used existing service descriptions in a prototype. A group of professionals is asked to simulate the use of an end-user in a case study research, where the end user satisfaction is measured. The results indicate that this part of the design does satisfy the needs of the end-user.

Future empirical research should be done to completely validate the methodology as a design approach for a service register. Until then, this research shows that the methodology encompasses several views to include all information needs and that it is a viable design approach.

Preface

About a half year has passed since I walked into the headquarters of Ordina TCT for the first time. A half year full of searching, typing, discussing, deleting, searching again, adjusting etc, that resulted in this thesis. Well, at least it kept me occupied but I hope to have written a report that interests certain audiences.

An audience that probably does not keep their interest when reading this report is about 95 percent of my social life. They friendly nodded along with me when I explained what I was doing and derived the conclusion: "So it is something like the Yellow Pages". However they also supported me during the past half year, and I thank them for having the interest and patience.

Furthermore, I would like to thank my supervisors Willem van Pruijssen (Ordina TCT), Fons Wijnhoven and Luis Ferreira Pires (both of Twente University) for their advice, contributions and invested time. It would have been difficult to write this thesis without their ideas and directions. Also the other colleagues at Ordina TCT who were friendly and willing to help are greatly appreciated.

Have fun reading,

Arjen van Oostrum

Nieuwegein/Utrecht, June 08

TABLE OF CONTENTS

1 Introduction

This chapter gives an introduction to this thesis.
Section 1.1 motivates the research topic.
The problem statement for this research is formulated in Section 1.2.
Section 1.3 formulates the objective of this research, by explaining how the problem statement is addressed.
Section 1.4 presents the structure for the rest of the report.

1.1 Motivation

For years, ICT is supporting business processes in various ways. However, although ICT was meant to be an enabler for business to change their business processes, in today's fast changing business environment it is often holding back the changes (Tigthart *et al.*, 2006). Bringing new products to market, reacting on governmental policy changes, or changing an existing process often takes a lot of time and expenses. The reason for this problem is that the existing systems are often specifically designed to support an entire process, and changes in the business process therefore have to be reflected in changes in the system. For instance, when a governmental regulation changes, several different systems have to be changed.

With implementing a *Service Oriented Architecture (SOA)* within a company, the promise is that with the offering of services the business process can be changed (and composed) more rapidly (Diesen & Namiri, 2006). A successful implementation of SOA depends on various factors. A structured approach to identify and implement *services* within an organization is important, as well as management support and *SOA governance*. Without SOA governance, it is difficult to reach the objectives of a SOA implementation, as measuring and steering are limited. Without organizational policies which are enforced by the organization, the SOA architecture results in a decentralized offering of services with no overview of which services are available with which policies/restrictions. This may for instance result in redundant offering of services or services which are not used.

A SOA architecture can be implemented within an organization more or less parallel to the existing IT infrastructure. Services can be implemented on top of existing applications to offer a specific functionality, while these applications continue to run. The implementation of SOA can therefore be done incrementally. The number of services in an organization over time typically shows this incremental nature, for example in Figure 1 for Credit Suisse (Krafzig *et al.*, 2004). The number of services in a SOA architecture can grow fast, making it more difficult for the involved stakeholders to find services and manage them. Without a service register which centralizes and standardizes the storage of information about the available services, there is an increased chance of inefficient use of the services.

Mahajan (Mahajan, 2006) identifies the governance of SOA and the governance of the (use of) service registers as a success factor for implementing SOA. According to Mahajan, one of the reasons for the low success of using Object Orientation for creating reuse objects in a market environment was the lack of a central register and processes to ensure the use of the register by IT and business. As a successor of different design architectures, the SOA architecture should

learn from the lessons learnt of the past, and therefore the use of a service register is recommended, and commonly accepted as a means for SOA governance. This service register is the topic of this research and therefore we will develop a definition of this concept here.

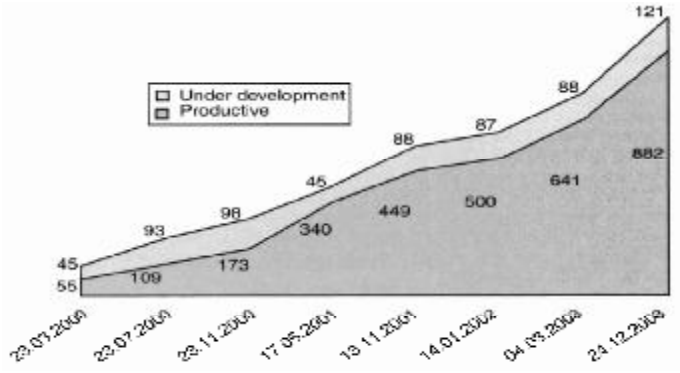


Figure 1 Development of number of services within Credit Suisse (Krafzig et al., 2004)

A commonly used description of a service register is "the Yellow Pages for services", which provides the basic functions for a creator to publish and consumer to find services. The service register can be implemented in an organization in different ways. A set of describing documents or a datasheet may already be considered to be a service register. Before we give a definition, it is useful to understand when consumers may need the information in the service register. There are two main purposes: to use at *runtime* and to use at *design time*.

Service specifications are used at runtime by applications that want to use the service itself. They query the service register for the service specification, which they use to bind with the service. The information which is needed here is primarily technical, containing location information and interface descriptions. For these purposes, service register implementations based upon UDDI and ebXML have been created. We name the user that connects to the service register at runtime (very likely to be represented by a software agent) a 'runtime user'.

At design time, service specifications are used by for example enterprise architects, application developers and business process engineers. These consumers need different information than runtime users, like the goal of the service or its business value, i.e. meta-information about the service. The information is meant to be interpreted by human users, and we name this human user at 'design time' to be an 'end-user'.

An accurate service specification is equally important for both purposes. Therefore, a service register should support both purposes and provide service specifications which reflect the real service as good as possible. This is why the service register can not be seen as 'just a system'. To be able to deliver high quality service specifications, the service register needs organizational processes to maintain the service specifications.

Providing accurate service specifications for use at runtime and design time thus requires more than a software system, organizational structures to support the provision of information have to be put in place. We define the service register as follows: *"The processes and systems that an organization puts in place to ensure that the storage, maintenance and retrieving of service specifications is done in a structured way, to facilitate the matching of a consumer request with a service specification of a creator".*

Today, organizations are looking for ways to fill, manage and disclose the information in the register in a structured way. The importance of having a service register is understood, but they are struggling with what information should be stored about services and how this information should be stored and retrieved. Both aspects are influenced by the context (organizational environment) wherein the register has to function. This context partly determines the goals, and therewith the requirements of the register.

This research focuses on how the information can be stored, managed and disclosed in the service register. To do so, a design is needed that addresses both the technological and business means, necessary to implement the service register. We do so by taking the perspective of an *Information Intermediary*, and use the design propositions from this field. Information Intermediaries transfer *Information Goods* from supplier to consumer; in this case the Information Goods are the service specifications.

1.2 Problem statement

While the SOA way of thinking is past the slope of enlightening of the hype curve (Gartner, 2007), it becomes more clear what the SOA abilities are. Although there is little experience with implementing SOA in organizations, some best practices can be derived. One of these is to use SOA governance. The SOA governance is becoming more important, especially when the number of services is growing. There is a common understanding that a service register can be useful for coordinating services, and initiatives are taken to create them.

Most existing service register implementations are based upon UDDI or ebXML technologies, which provide little support for other information than technical information via nested templates with information (Ferdke & Loos, 2003; Luo *et al.*, 2006; Turowski *et al.*, 2002). Therefore, the service registers are primarily useful at runtime, when an application asks the register for the address of a certain service. However, access at design time, providing information about the service is at least as important, e.g. to prevent redundant services.

Consultants in the area of SOA implementations experience customer requests for information about "how to fill registers with what information, and how to manage and disclose the stored information". Two main aspects have a role here, linked to each other:

1. What information is stored about services. This question is not answered by this research. However, some knowledge of "what information" is required to describe how it is handled. Therefore, for this research the seven levels of service specification as described by Turowski (Turowski et al., 2002) are adopted as a guideline for what information is stored. The levels indicate different areas of information which should be stored about a service, ranging from explicit technical information to the business goals the service contributes to. This is further explained in Section 3.3.
2. How is a service register designed so that it supports the information needs of its users? This is the subject of this research, and combines both technical aspects with organizational issues. The information has to be stored in such a way that it can be filled, managed and disclosed, which is mainly of technical nature. Furthermore, changes in the

organizational structure and processes have to be defined to support the working of the register itself and to maintain the quality of the specifications in the register.

Figure 2 shows a problem tree which traces a problem (an unsuccessful SOA implementation) back to its plausible root problems. The root problems (in the figure shown in bold) are to be addressed when solving the main problem

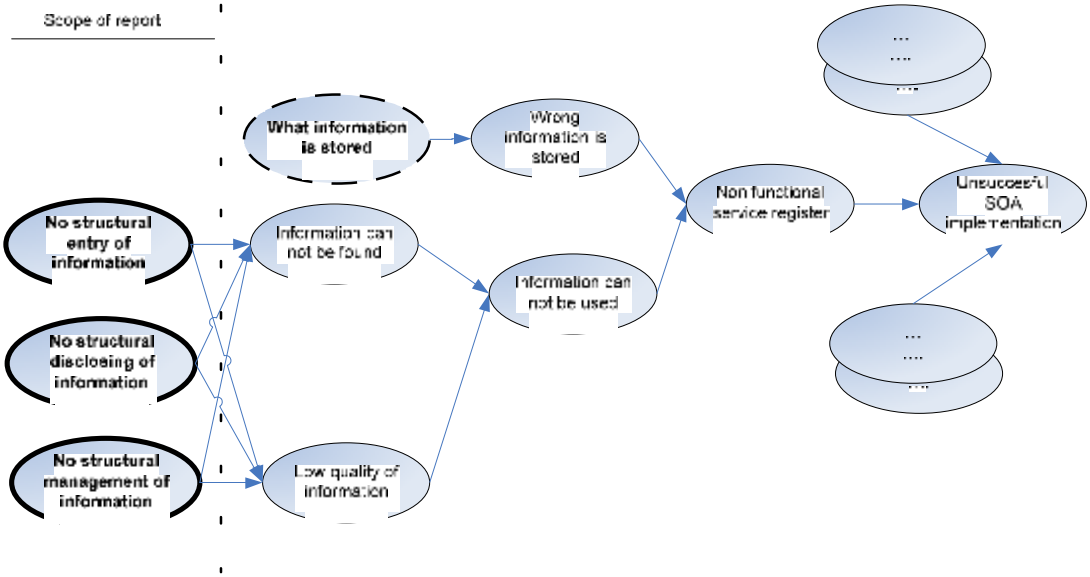


Figure 2 Problem tracing to research topic

The problem statement for this research is formulated as follows:

P1. *How should a service register store information about services and how can the register be implemented within an organization to structurally fill, manage and disclose service information in order to deliver high quality service specifications.*

1.3 Objective

In addressing this problem statement, different perspectives can be taken. Up-to-date, service register designs have been primarily created for runtime use (i.e. querying the register to obtain the location of a specific service). Therefore, service specifications are stored in low level information schemas. Due to the technical nature of the information, management of information is focused on database management (data structures), less on the service specifications itself.

This research has taken another perspective, and approached the service register as an Information Intermediary to apply a structured methodology to the design process. This enables the use of design theories which not only focus on the technical aspects of delivering the right description to the user, but also the organizational processes and procedures to support this. The result is an attempt to create an effective design for a service register as an Information Intermediary.

'High quality service specifications' is difficult to asses. According to (Reeves & Bednar, 94), 'quality' has at least four dimensions: Excellence, Value, Conformance to specifications and Meeting and/or exceeding customers' expectations. When designing the service register, these dimensions have to be kept in mind. The latter dimension, 'Meeting and/or exceeding customers expectation' is about the perception of the product by the end-user, i.e. the end-user satisfaction. High end-user satisfaction is recognized to be a key factor for success of Information Systems (Doll & Torkzadeh, 1988; Thong & Yap, 1996)

The objective of this research has first been to provide a meta-model for a service register using design science of Information Intermediary. The design is already focused on the delivery of high quality service specifications to end-users, for a high end-user satisfaction. Different views are taken into account (i.e. management, creators, maintenance and consumers) and we discuss the various aspects (i.e. content, use features and revenue) and layers (i.e. business, process, infrastructure), applicable to the design.

Validating this complete model would have required the implementation of the service register within an organization, with a development and implementation project which is not in the scope of the assignment. Therefore, we focused on the interaction of the human end-user, searching the register for service specifications. A design is defined to be effective if the end-user satisfaction is high, and a prototype is created for this part of the model to check the satisfaction.

The contribution of this research has been the result of this design approach, by providing a model consisting of a technical design and descriptions of organizational processes and procedures which should be created. This contributes to the service register research topic by providing a structured way of designing a service register. Furthermore, the objective has been to identify elements which are to be considered when designing a service register.

1.4 Structure of report

This report is further structured as follows:

Chapter 1 and Chapter 2 provide the context of the thesis. Chapter 1 has introduced the research topic and explained the motivation and problem statement. Chapter 2 sets out how the research is approached and what context is defined. Furthermore, the research questions are defined.

To create a theoretical background and become familiar with the concepts used in the research, concepts are defined in Chapter 3. The main concept (the service register) is explained separately and more in-depth, discussing the various views on a service register.

Chapter 4 explains the design methodology that is used in this research, and discusses the various aspects of this methodology for the service register situation.

The design methodology is followed in the subsequent chapters of the thesis. It consists of various layers that are represented by these chapters. Some of the layers are combined into one chapter, as is the case with the layers that deal with requirements.

Chapter 5 starts with a brief introduction of the design problem and agenda, the first of the design layers. The remainder of the chapter is meant to identify the requirements of the service register. The functional requirements of a service register are identified, by discussing the

information needs of the actors which are part of the business layer (i.e. what is transferred for what reason). These are to be supported by processes, which are discussed. Furthermore, the non-functional requirements are briefly discussed using the FURPS+ method (Tarman, 2002).

To address the meta-requirements, Chapter 6 indicates infrastructural needs that are to be used. An integrative model is constructed that shows the service register.

The next layer in the methodology is the prototyping of the design. However, due to scarce resources, we only focus on the interaction part of the human end-user with the system in Chapter 7 and specify this part in more detail and describe the construction of this part in a prototype.

The final layer is the exploitation of the service register, which is again limited to resources. A research case is introduced in Chapter 8 which is used to measure the user satisfaction. The remainder of the chapter describes typical exploitation issues.

Chapter 9 discusses the results and reflects on the design guidelines that are used.

2 Research Design

This chapter explains how the research is designed to address the problem statement in the previous chapter.
The first section, Section 2.1 describes the design science which is followed when performing the research.
Section 2.2 defines research questions which are answered during the research and shortly describes the approach of the research by describing the process steps.
Section 2.3 explores the context of the service register by introducing a case which will be used throughout the research as a reference case. This context scopes the design of the service register.

2.1 Information System Design

The design of Information Systems has a short history of scientific research compared to other disciplines. As Nunamaker & Chen argue in their article (Nunamaker & Chen, 1991), the design of Information Systems should be considered as scientific research in the applied science category. Today, the importance of Information System design is well understood and multiple authors have attempted to guide the development of Information Systems. Hevner et al (Hevner *et al.*, 2004) provide 7 guidelines which can be followed when designing an Information System. These guidelines recognize the importance of the environment for which the system is designed.

The above mentioned insight shows that when designing an Information System it is not enough to design a technically well-designed system. The environment wherein the system has to function is at least as important. To do so, various techniques have been developed to support the design of the Information System. Traditionally, the so-called waterfall method is popular, in which the design of an Information System is divided into various sequential phases with feedback flows. More recently, methods like the rapid prototyping model or spiral model are used, although they do not replace the existing models. In order to decide which model to use, one has to consider the context of the proposed system.

The ideas of Nunamaker & Chen (Nunamaker & Chen, 1991) match with the design guidelines of Hevner et al (Hevner et al., 2004), which focus on using existing research for developing a system design that contributes to the knowledge and is relevant for the real world. Our research adopts these thoughts and uses the guidelines from Hevner et al and the research process as proposed by Nunamaker & Chen. Figure 3 shows the combined theories, where the research process of Nunamaker & Chen (thick solid line) is embedded in the model of Hevner et al. In addition to the design process of Nunamaker & Chen, the design guidelines of Hevner et al make an explicit connection between the design process and environment/knowledge base. Therefore, by following the design guidelines of Hevner et al. the design process is followed, and is relevant for the environment and knowledge base.

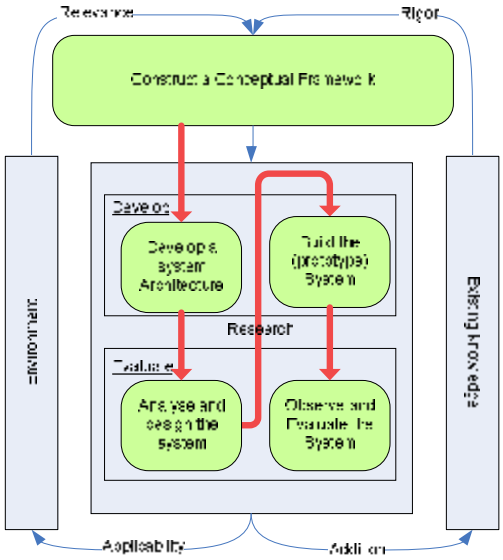


Figure 3 Information System design model

In Table 1, the guidelines from Hevner et al. are further explained and the way they are used in this research project for designing a service register is discussed.

Table 1 Guidelines Hevner

Guideline 1	*Design as an Artifact*
Description	Design science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Applied in research project	The project has delivered two artefacts, a model for a service register as an Information Intermediary and a prototype implementation showing a part of the model.
Guideline 2	*Problem Relevance*
Description	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Applied in research project	Although the SOA concept is growing, it is still a relatively new way of thinking. As it is, there are some gaps to be filled; one of them is subject to this project. The results can be used in the practice of SOA implementations as a basis when considering the use of a service register.
Guideline 3	*Design Evaluation*
Description	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well executed evaluation methods.
Applied in research project	The model is discussed with professionals in the topic area. Developing and implementing the complete service register is out of the research scope. A part of the design is further developed and checked with a prototype, further evaluation and testing of the model are left for future work.
Guideline 4	*Research Contributions*
Description	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.

Applied in research project	The results of the project provide a new model for designing service registers as an Information Intermediary. The design is contributing to the research area as another way of designing a service register, which is not applied before. The scope of the service register as used in this research narrows the applicability of the design.
Guideline 5	*Research Rigor*
Description	Rigor design science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Applied in research project	The design is based upon a literature review and uses relevant design methods for an effective design. It uses an existing, structured, methodology to create a design for the service register.
Guideline 6	*Design as a Search Process*
Description	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Applied in research project	The approach of using Information Intermediaries design theory for an effective service register design consists of searching for a well fitted design. Knowledge about the problem environment is retrieved by regular discussions with professionals. These discussions have indicated a need for a structured approach to service register design (see chapter 1). This thesis is in fact a search for such an approach by applying a structured method from other design literature to the service register to propose a new design methodology for service registers.
Guideline 7	*Communication of Research*
Description	Design science research must be presented effectively both to technology-oriented as well as management-oriented audiences.
Applied in research project	The thesis has been presented at Ordina and the University of Twente, and published in the University library. The ambition is to publish in an external magazine or journal afterwards.

2.2 Research questions and approach

The main goal of the research is to provide a possible solution to the problem statement as stated in Section 1.2 by designing the service register as an Information Intermediary. The main question is therefore formulated as follows:

MQ *Are Information Intermediary design principles a possible structured method for service register design?*

The 'possible' part is added for two purposes. For one, it indicates that this research is a search for a structured method. Others may be proposed, this view on service registers proposed here as a possible view. Secondly, the (time) resources for this research are too small for a complete validation of the method. As the complete method cannot be evaluated well enough, it remains a possible approach to service register design and not a proven approach.

We first need to clarify concepts and relations between concepts that apply to Service Oriented Architectures and the service register to create a background for the research. The associated research questions are the following:

Q1 What are the benefits of using a service register in Service Oriented Architectures?

Q1.1 What are general reasons for implementing SOA?

Q1.2 How can a service register help in reaching the advantages of SOA?

The motivation for using a service register is discussed in Chapter 1. The concepts, relations and goals associated with SOA are discussed in Chapter 3.

Having introduced the research topic, the question is how to design a service register as an Information Intermediary. A literature review answers the following questions, related to the design purpose of this research.

Q2 What techniques are available for designing a service register as an Information Intermediary?

Q2.1 How to use general Information System Design research?

Q2.2 What design process for Information Intermediaries is used?

Next to the design method, the server register itself is examined. As stated in the motivation, the service register is more than a software system alone, nevertheless this is an important part of the register. It is useful to know for what purposes the service register can be used.

Q3 What characteristics does a service register have?

Q3.1 What information is requested by various views?

Q3.1.1. Which views are considered?

Q3.1.2. What information types are needed by the different views?

Q3.2 Which ways of service register implementations can be distinguished?

The answers give a design context in which the service register is designed. Next, an Information Intermediary perspective is taken and answer is given to the following questions:

Q4 What are the design constructs of Information Intermediaries (content, use features, revenue) in a service register setting?

Q4.1 Which design scenario is used for service register design?

Q4.2 What are the aspects *content, use features, revenue* of an Information Intermediary when applied to a service register?

These meta-requirements are taken into account in the meta-design in which we address the following questions. These are based upon the design methodology for Information Intermediaries.

Q5 What is the design for a service register, designed as an Information Intermediary?

Q5.1 What is the design problem and agenda

Q5.2 What values are exchanged? *(requirements)*

Q5.2.1 Which stakeholder roles are identified?

Q5.2.2 What interactions do the stakeholders have?

Q5.2.3 Which additional requirements do the stakeholders have?

Q5.3 What processes support the requirements of the stakeholders? *(requirements)*

Q5.4 Which technical and organizational infrastructure support the processes? *(design)*

Q5.5 How is this used in practice *(prototype)*

Q5.6 What are further exploration issues?

With answering these questions, the design of a service register as an Information Intermediary is formulated.

The research questions are answered in different chapters of the report. Figure 4 illustrates the following description of what has been done in this research.

We provide the reader with an illustrative case in the next section, which is used as an example for the use of a service register. The described usage reflects the normal use of the register as acknowledged by field professionals, though the described organization is fictitious. This case is the context for which the design is created.

Together with the design science literature for Information Intermediaries and literature about service registers, this provides the conceptual framework. From this conceptual framework, we define meta-requirements and start the meta-design, following a design theory for Information Intermediaries. Relevant chapters here are Chapters 1, 2, 3 and 4.

Based upon this knowledge, we create meta-requirements and a meta-model for a service register as an Information Intermediary. The model covers the meta-requirements and explicitly deals with the organizational means necessary for supporting the service register. This is done in chapters 5 and 6.

We continue the application of the design method with a prototype that satisfies the end-user of the service register. Creating a complete model is out of the scope, therefore we focus on an important part of the model; the interaction with the end-user which demands information at design time. This component is further developed into a prototype for a real case situation, which is checked with the criteria for end user satisfaction of the system. Chapters 7 deals with the design and development of the prototype.

End-user satisfaction measurement of information systems is described by Doll & Torkzadeh, (Doll & Torkzadeh, 1988) who revised the instrument of Ives et al. (Ives *et al.*, 1983). The instrument of Doll & Torkzadeh is well established, and tested in different settings over the last decade. It is also acknowledged to be useful in both data warehouse (Chen *et al.*, 2000) and knowledge management (Ong & Lai, 2004) settings, which are connected to the field of service registers. As an Information Intermediary is likely to run via the internet, the main access point for the user is via a web browser. To measure the web-performance, indicators have been

identified which indicate the performance. However, these measure comparable areas as does the instrument of Doll & Torkzadeh.

Accordingly, we measure the user satisfaction of the prototype in the areas as identified by Doll & Torkzadeh, being *Content*, *Accuracy*, *Format*, *Ease of use* and *Timeliness*. This is discussed as part of the exploration chapter in Chapter 7.

We reflect on the research to provide feedback to the body of knowledge and environment by evaluating and discussing the contribution to the design guidelines used and provide conclusions in chapter 9.

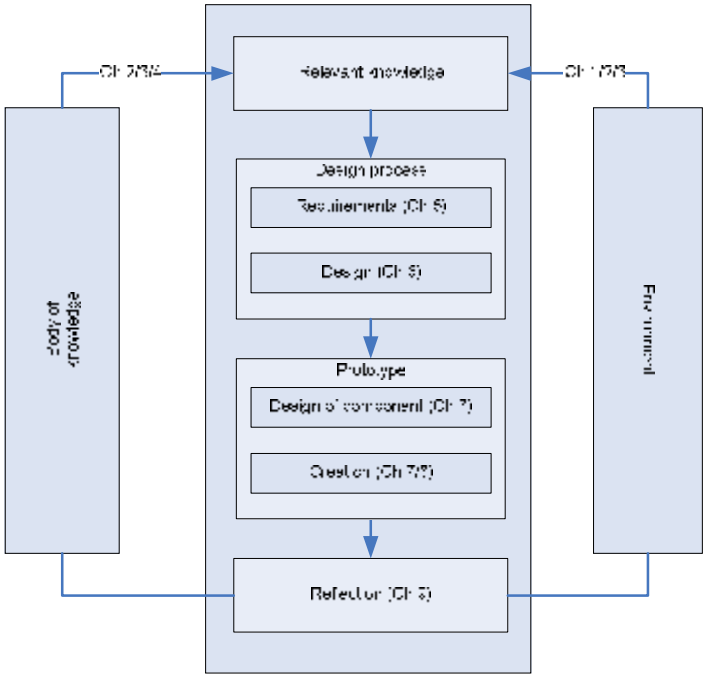


Figure 4 Research process model

2.3 Context exploration

Among other variables, requirements of an Information System also depend on the context where it is placed in, i.e. the type of organization, other systems, assigned responsibilities etc. This section creates such a context for the service register, and is used throughout the report. To motivate the use of a service register, a scenario shows the difference between a situation with and without a service register.

2.3.1 Scoping of service register

A first scoping is the target user group of the service register. Some initiatives are taken to create public accessible service registers, in which any provider may publish its service and any consumer may look for a service to consume. Those initiatives were not successful, as the stored service specifications could not be trusted. Some services which were described didn't exist anymore or behaved different. Therefore, the first scoping is that the service register will be used

in private setting. The private setting may be a single organizational unit, an organization itself or a network of trusted organizations.

The service register is one of the tools which can be used for SOA governance. All of these tools may -or may not- be put in place in a specific SOA architecture. Some tools can have overlapping functions with other tools, so it is important to determine the responsibilities of each tool. The service register is a central tool, with a primary task and supporting tasks. The primary task is to deliver (certain parts of) service specifications to users at design time (meta information) and runtime (technical information). Furthermore, it supports other tools by providing information which is needed. For instance, an invoicing tool might need the price information of a certain service.

In future scenarios, applications will be able to dynamically determine the right service to use for their purpose. To achieve this 'semantic web' requires well-constructed semantic matching processes which are not yet available. Therefore, the service register design will not be designed for use in the semantic web.

2.3.2 Case introduction

There is no industry for which a SOA is not useful, but in some industries the benefits of using SOA are reached earlier. For our case, we choose the insurance industry, as a lot of checks are made in business processes, and (governmental) regulations have to be honoured strictly.

A large insurance organization, named "Sure 4 U", traditionally provides car insurances. After the recent merger with "Fit your body", also health insurance products are offered. At the time of the merger, the management board decided to adopt SOA to integrate the Information Systems of both companies.

The technical environment is a mix of various systems, with an internal IP-based network. A radius server provides authentication of users for applications throughout the organization. Functionalities of the systems are offered through interfaces as web services and connected to a central Enterprise Service Bus (ESB). Sure 4 U has just one IT organization which is responsible for maintenance and creation of the services which are offered. A separate unit is running the service register, and coordinates the provision of service specifications to both design- and runtime users. Management has decided that the costs of the service register unit should be covered by revenues created by the service register.

The two organizations will function as cost centres, and the costs of using a service is calculated and invoiced to the user. Additional to the fee which is paid for the service, the user has to pay a fee for searching the service register to find the service specification. However, to stimulate the use of the service register, the management sponsors the service register by providing a subsidy which covers the expenses to use the register for a large part. The end-users also pay a fee for searching in the register, which is subscription based by their unit.

2.3.3 Service register

With the service specifications stored in the service register, an overview is available of which services can be used. For each service, the users are identified and data is available of which runtime users are subscribed to a service.

The service register ensures that the information that is stored is up-to-date. As the application queries the register for the address of the service at each occurrence, recent changes in the address are available for the application.

Other parts of the service specification are used by users at design time (i.e. the end-user). These users query the service register to search for existing services, for instance based upon their goals. Other usages by end-users include querying the life cycle states of the service or business information such as ownership and boundaries.

The service register itself is also offering services which can be used to retrieve, store or alter service specifications. Offering these services is costly, and Sure 4 U has determined that the service register should be cost-efficient, and therefore will have to earn its own costs by generating revenues.

3 Background literature

This chapter provides a relevant body of knowledge about this research. To start with, the concepts which are used throughout the thesis are discussed and defined in section 3.1. Section 3.2 first discusses how service registers can be implemented. The subsequent Sections 3.3, 3.4, 3.5, and 3.6 discuss why the viewpoints of an Information Intermediary (respectively creator, user, management and maintenance) use a service register. The goal of a service register as we see a service register is discussed in 3.7. The earlier discussed actor views are split-up in several smaller actor roles in Section 3.8. Section 3.9 shows various matching techniques for the matching of service specifications.

3.1 Concepts

This section provides definitions for relevant concepts used in the thesis. The service register concept is already introduced in the motivation section 1.1 and is extensively further discussed in section 3.2.

3.1.1 Service Oriented Architecture (SOA)

The subject of the research (service register) is part of the SOA way of aligning business and IT. Through the years, much attention from different views is given to the notion of SOA, and as a result there are many different interpretations of the SOA concept. However, there is some consensus that SOA is not an architecture in itself, but is an architectural pattern, providing the opportunity to create Service Oriented Architectures (Lewis *et al.*, 2007).

Other authoritative sources which provide a definition of SOA (Gartner, 2003; OASIS, 2006) have similar thoughts about the meta-level of the SOA concept, in that it can be used to describe an architecture of resources. These resources can be *services* which offer certain functionality but also the organizational resources and means to be able to measure and steer these services.

Thus, SOA is more than a technological architecture, and more than an overview of business processes. It provides the means to integrate Business and IT with the use of services.

For this research, the concept of SOA is defined as follows:

A Service Oriented Architecture is an architectural pattern that provides means to describe the use of measurable distributed resources for aligning business requirements with a defined application topology.

Introduction to SOA

Before continuing with defining other concepts, we will briefly discuss how the SOA concept developed and is used today.

When organizations started using IT to automate parts of business processes, the used IT was very specific for its purpose and location (monolithic applications). Through the years, with the development of technology, innovations in IT resulted in more structured and flexible IT structures. Via client/server and n-tier architectural styles, the (re)use of software components became popular and was popular because it enabled organizations to create flexible integration with business processes. However, this had its shortcomings like that application programmers

had to know which component should be used in advance. Also different vendors used different interfaces for their components, which made it difficult to bind between different systems and between organizations.

The concept of SOA was first described in the late 90ties but it was not very popular until the industry introduced web services and corresponding standards. With these standards, the different interfaces of vendors can be used with less effort, which addresses one of the main issues of component reuse. Another promise of SOA is the runtime choice of what component can be used, although this promise is not yet fulfilled (Vrijkorte, 2006).

In contrast with component reuse, SOA is more than only reusing software components. A full SOA implementation is another way of looking at the alignment of **IT** with the business in an organization. In the vision of SOA, only software reuse is not enough for agile business processes.

The main components of a SOA architecture, the services, can be located in all parts of the organization. This organization-wide focus compared to departmental focus differentiates SOA from the earlier architectures like Client Server and Object-Oriented Programming (Mahajan, 2006). Using services provided by other organizational units reduces overall development and maintenance costs.

Another benefit of SOA is the opportunity for incremental development, deployment, maintenance and extension of business applications (Gartner, 2003). SOA can be implemented parallel to existing architectures and systems, not necessarily replacing them. It makes use of existing systems by encapsulating parts of the system them as services and provides access to these services via interfaces that are agreed upon and published. This reduces the number of interfaces necessary to provide access to systems, which facilitates the maintenance of the system. Maintenance of new or changing (governmental) regulations is also facilitated, as these have to be implemented in just one or few services, not in every application.

Other benefits include the reuse of the business components in other business processes, clarity of application topology and lower costs assembly of new processes, as existing services can be used to form new services. Therefore, also small organizations are able to profit from services, by using just the services they need, so they do not have to buy a complete enterprise system(Vertke & Loos, 2003).

However, SOA is no silver bullet and does have its limitations. Among others, some false expectations of SOA are (Gartner, 2003; Lewis et al., 2007):

- *SOA is simple software engineering* – Although it facilitates a little, SOA does not automatically support automated or simple software engineering.
- *SOA provides free integration and/or interoperability* – Integration and interoperability are facilitated, but still have to be done.
- *SOA creates total Technology and Vendor independence* – Although the technical implementation of the services is decoupled of the performed action, it still has to be

implemented on software and hardware platforms, and the integration technology also has to be delivered by a vendor.

- *SOA provides complete architecture for the modern organization* – SOA can be implemented in many ways, and consists of multiple entities which are topologically linked together. A lot of decisions have to be made, which may result in any number of different architectures. Just adopting SOA is thus not enough.
- *SOA can integrate legacy systems with little effort* – Although SOA makes it possible to use legacy systems, these systems still have to be changed to support the offering of services. With specially designed legacy systems, this may be difficult.
- *SOA is primarily about technology* – SOA aims at supporting business processes/functions by providing IT support in the form of services. By adopting SOA, organizational changes have to be made to support the new way of working, which are often underestimated.
- *Use of standards automatically provides interoperability* – There is no such thing as one standard which will be used for years to come. For each standard is an alternative, and no standard is stable for years in the future, as new technological developments are reflected in standards.
- *SOA applications just add services in the right order* – This oversimplification of the development process of a SOA-based application does not consider the efforts which have to be spent to, for instance, find, compose and use the services. When services are composed, it is highly unlikely that the output of service A can directly be used as input for service B, but this output has to be transformed to meet the preconditions on the input of service B.
- *SOA services can be used by anyone* - Although services are designed for reuse, they often have limitations because of the context and granularity.

Initiatives have been taken to guide the creation of a SOA architecture such as, for instance, a reference model (OASIS, 2006), an UML profile for SOA (Heckel *et al.*, 2003) and implementation guidelines (Lewis et al., 2007; Legthart et al., 2006; Mahajan, 2006) for structured implementation.

The discussion above shows that SOA is based upon other IT architectures, but has additional promised advantages and should be considered as a new way of using IT for business processes. The main services which provide functionality can be (re)used in business processes. This loosely coupled architecture creates opportunities for a quick reaction towards changes in the business environment, for instance, when new products are offered or binding between organizations have to be realized. However SOA implementation should be done with care to avoid misconceptions and in order to achieve the promised advantages.

3.1.2 Service

The name Service Oriented Architecture and the discussion of SOA in section 3.1.1 show that the concept of service has a central role in the SOA architecture. A service in the context of SOA is defined by various authors, from which two important elements can be derived:

a) *The service is about the observable behaviour.* The internal working of a service is not relevant. The service is considered to be a black box, which delivers a certain function.

b) *The environment of the service has a role.* The environment of the service (other systems/organization) constrains the service as it expects certain behaviour. The degree in which the service fulfils this expectation (partly) determines the quality of the service.

As the observed service behaviour should satisfy the expectations of the environment, it is important that the environment has a correct perception of the service. Therefore, the description of the service should be stored and maintained carefully and completely. The definition of the concept of service in this research is: *"The observable behaviour of a mechanism that provides access to one or more capabilities via a prescribed interface to its environment"*.

3.1.3 SOA governance

As the discussion of SOA in section 3.1.1 showed, a successful implementation of SOA requires more than just create services. To ensure that SOA is done in accordance with agreed architectural principles, best practices and regulations, processes are to be put in place (Manes, 2005). These processes are a combination of technology, procedures, processes, information and tools and are defined as SOA governance. This SOA governance is identified to be a key factor for success of SOA (Lewis et al., 2007; Ligthart et al., 2006; Mahajan, 2006).

SOA governance thus includes a wide range of aspects that have to be considered, and various questions that can be answered. To ease SOA governance, a service register is used to centralize the service specifications and governance policies.

3.1.4 Information Intermediary

As shortly mentioned in the motivation section 1.1, we take an Information Intermediary perspective for a service register design. While this choice is further motivated in section 3.2, we explain the concept of Information Intermediary in this section.

A particular type of information system is an Information Intermediary[7], which transfers Information Goods from a creator to a user (Wijnhoven, 2008). Such a system generally has multiple users and contributors who exchange Information Goods via the Intermediary. To facilitate the matching of the user's request and the creator's offer, an Intermediary makes use of various processes and tools. Among others, these include changing the Level of Conceptualization and Level of Representation to offer the best suitable Information Goods.

[7] An Information Intermediary is often called an Information Service, but the term Information Intermediary is chosen here because Information Service might be confusing, as in this context, a service specification is stored in the Information Intermediary.

Domain ontologies facilitate the use of the Intermediary (Wijnhoven *et al.*, 2003; Zimmermann *et al.*, 2002). Information Intermediaries can be designed in various ways; each context and organization provides its own specific design requirements. To support the process of designing Intermediaries a design process theory can be used (Wijnhoven, 2008; Wijnhoven & Kraaijenbrink, 2008).

A schematic model of an Information Intermediary is given in 3.1.4, in which an Information Good described in a vocabulary/ontology, is transferred by the Information Intermediary to the user. Ontologies are used to semantically match the user's request with the Information Good of a creator. Management and maintenance activities and needs influence the stored Information Good.

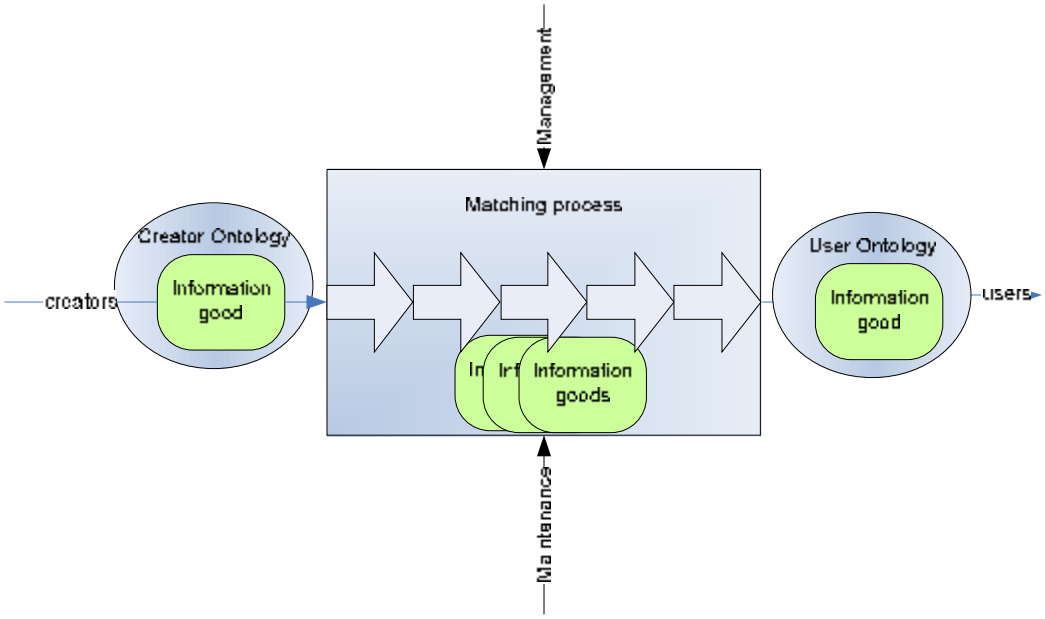


Figure 5 Schematic representation of an Intermediary

An Information Intermediary can also be considered as a type of Information and Knowledge Management system. A Knowledge Management System (KMS) facilitates at least few of the following aspects of the definition of Knowledge Management: "*the generation, representation, storage, transfer, transformation, application, embedding, and protecting of (organizational) knowledge*" (Earl, 2001; Schultze & Leidner, 2002). The connection between this definition and an Information Intermediary is straightforward: Information Intermediary use several of these processes to deliver the Information Good from creator to user.

3.2 Service registers Implementations

This section discusses the current research of service registers and presents the viewpoint of this research. Although a great share of available literature is focused on semantic discovery of services for the 'semantic web', we ignore this view as it is outside the scope of the research to design a service register capable of semantic discovery.

Using a service register makes it easier to have knowledge about what services exist, where they are implemented and with what agreements they are used. Especially for large SOA architectures, a service register is necessary to keep track of the available services.

A service register is defined to be more than a software system alone, and also needs organizational structures and means (see section 1.1). However, a software system is an important part of the service register. This section discusses various ways to implement IT support for a service register. The actual service register implementation depends on the ambition level and/or number of services of the organization in which the service register is to function.

In its simplest implementation, the register is a flat database (for example an excel sheet) or a set of documents which contain specifications of services. Searching and maintaining such a service register is difficult, however can be enough for small organizations or organizations with just a few services. It is not possible for runtime users to retrieve information of the services to check where the service is located. The location therefore has to be implemented hard-coded in the application which will invoke the service, which restrains updates and location changes. End-users can read the specification, although it can be difficult to find a service specification.

Organizations which want to be more flexible in the use of services (or have more ambition and/or services) can use a register system which stores primarily technical specification of the services and provides a shell with functions to approach the data. The application developer can query the service register for the desired service, and use the technical information to let the program bind to the service. The service location which is stored is used, making the program more flexible than the first simple implementation. This is the most basic dynamic form of binding (Lewis et al., 2007). However, as the stored information is primarily meant for runtime inclusion, it is difficult for a human user to retrieve information about the service such as service lifecycle information (further discussed in Section 3.4).

An implementation which is capable to store additional information about the service specification (e.g. to support the service lifecycle (Section 3.4), consumer-focused information (Section 3.3)) is another way of service register implementation. Additional to the runtime querying support, the service register system offers human users more possibilities to search service representations and use the information for various goals.

A further extended implementation adds richer machine-interpretable meta-data to the service specification, and is able to reason with and about the service specifications. The discovery of services can be automated, giving advantages such as dynamically binding of application and service. Depending on for instance the goal of the desired service, the service register returns a list of services which may be used to reach that goal. It is also possible that the service register returns a set of services complementing each other, and are able to achieve the goal. This automatic processing is the goal of the semantic web, in which applications and services are able to reason with knowledge and negotiate about the usage. However, it is a misconception that the finding of services is, or can be, fully automated with existing technologies (Lewis et al., 2007). Recent studies show that the various ways of discovery (keyword-based, description logic and rule based logic) lack possibilities or can not be implemented with the existing tooling (Cabral *et al.*, 2004; Vrijkorte, 2006).

Table 2 summarizes the ways of implementing the service register system. It is not meant to be complete, however to indicate various ways of implementing a service register. The column 'Useful at' indicates for whom the implementation can be used (see Section 1.1)

Table 2 Different ways of service register implementations

Implementation	Ambition	functionality	Advantages	Drawbacks	Useful at	
					Runtime	Design time
1	Low	Storage of service information in flat files or simple database	Storage of information for future reference	No possibility to quickly search the information	No	little
2	Medium	Possibility of querying the register, primarily for technical information	Consistent way of storing. Possible to search the information.	Use of vocabulary is important to be able to find services again.	Yes	little
3	Medium	Possibility of querying the register (at runtime and design time) and support for meta-information for lifecycle and registration purposes.	Consistent way of storing service specification. Possible to store meta-information. Enhanced possibility to search the information.	Use of vocabulary and working with standards is important to be able to find services again. Requires more data maintenance.	Yes	Yes
4	High	Possibility of adding rich semantic information.	Description of services in other way. More flexible description, possible to search for service based on capability.	Advanced adding of services, difficult to maintain.	Yes	Yes

Today's solutions for service registers (e.g. based upon UDDI and ebXML) provide little more functionality than basic querying of technical information and owner information. Compared to the seven specification levels of Turowski et al (Turowski et al., 2002), some levels are missing or not explicitly implemented, e.g. the marketing and goal levels. This is the reason why organizations seek for alternative methods to store service specifications which they find, for example, in using MS Word-based templates for service specification.

The UDDI-based service register supports keyword-based search that match the keyword with words in the service description. This has the drawback that synonyms and plurals are considered as different terms. 'Wild' characters such as '%' or '*' give more results but require more human intervention. Therefore, initiatives are taken to extent the possibilities of the service registers to

support both the storage of additional information(for example goal and marketing information) and maintain the possibility of automated search and usage of service (Studer *et al.*, 2007). This is mainly done by adding ontology support (Colasuonno *et al.*, 2006; Luo et al., 2006; Studer et al., 2007). The use of ontologies is proven to be more effective in data matching solutions, and we will include this in the design, as discussed in Sections 3.9.1 and 3.9.2.

We recognize the importance of the end-users to use the service register, and will therefore focus on the third implementation which provides support for these users. However, as we will approach the service register as an Information Intermediary, we also pay attention to the other actor views that are previously indicated in Section 3.1.4. In the remainder of this chapter, the main actor view groups of the Information Intermediary (see Figure 5) are discussed for what purposes they use the service register.

3.3 Consumers view

The consumers are the end-users of the service register. These may be humans in an organization, or an application which is represented by a software agent. These users have different demands for information, and therefore we make a distinction between them. First, the register can be used at design time, where mostly meta-information (information about the goals, application etc) is requested. We stated before that this (human) end-user has the main focus of our design approach. Secondly, users at runtime have a demand for technical data.

In order to design a service register that stores and transfers service specifications, we should first discuss what information is stored in a service specification. Before storing the services in the service register, the services first have to be identified and specified. There are multiple ways to identify services such as, for example, based upon stakeholder/communication patterns (Geurts & Gielhoed, 2004; Klose *et al.*, 2007). However, the identification methods generally focus more upon the matching of the business functions in the business domain with IT components in the IT landscape (An *et al.*, 2006; Levi & Arsanjani, 2002; Shishkov *et al.*, 2006). This indicates the need to describe the services at both business and IT level.

Hubbers et al (Hubbers *et al.*, 2007) define 10 methods for the identification of services, ranging from business oriented to technical oriented approaches. Most likely, different methods can be used when identifying the services, depending on the context. Although the variety of existing methods may make it difficult to choose, it is important to follow a structured method for identifying services. Otherwise, the search for services may result in services which are non-feasible, not useful or with the wrong design granularity. Finding the right design granularity of the services is of high importance. A low granularity results in services which can only be used in specific situations, whereas a high granularity results in a surplus of services which have to be maintained.

After identifying the services, the services have to be specified. Various authors have defined service aspects which should be specified, for example (Quartel *et al.*, 2006; Turowski et al., 2002). The seven specification levels of Turowski (Turowski et al., 2002) describe both technical and business aspects of a service, and are developed from different consumer views. These levels thus represent different information needs of consumers, and are shown in Figure 6.

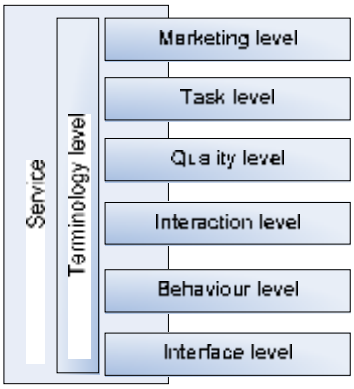


Figure 6 Levels of service specification, adapted from (Turowski et al., 2002)

The terminology level is identified as a separate level in the model of Turowski et al, however it describes concepts and elements which are used in all layers. As it does not explicitly adds specification information of the service, it is doubtful to include it as a separate specification layer. However, it is surely important to describe the concepts and other terms in a structured way to avoid confusion. Therefore, the terminology level is included as an integrating layer, to show it is facilitating the other layers and is not a specification layer itself.

These levels represent the information needs of the design end-user and –with the interface level- from the runtime user. We refer to the combined levels as the 'core information' in other sections.

Throughout this chapter different information types which are to be stored in the service register are identified. Figure 7 starts the identification with the information type recognized in this section, the core information.

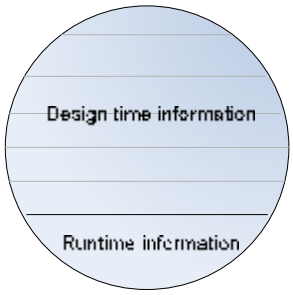


Figure 7 Representation of core information

The core information, here consisting of the specification information, is represented using a circle that is divided into design time information and runtime information. The runtime is the lowest level of the Turowski model, the interface level. This level describes the location and interface of the service that is represented and is primarily meant for automated inclusion by software agents. The other levels describe the use of the service, and although parts of these descriptions can be made readable for machines, the main purpose of the information is to provide the human user information. An exemplary usage scenario is a designer that uses the specification to determine if it suits his needs.

3.4 Creators view

The creator can be an organization, organizational unit, or any other entity providing an Information Good, with the purpose of enabling the use of its service by others, by publishing the service specification in the service register. Creators have the responsibility of maintaining the service according to the service lifecycle.

In analogy with other IT elements, services have a lifecycle which they follow. Without paying attention to the service lifecycle, services may lose their purpose early or do not fit the architecture. A service register helps to enforce service lifecycle policies and provides an overview of the status of the services. A service specification is registered in the register from the first phase of the lifecycle and records all information. It is important to begin the storage of life cycle information from this point onwards to know with what purpose the service is created. The stakeholders which are involved in the service lifecycle use the service register to retrieve information, which results in different information needs.

Combining the descriptions of a service lifecycle in (Afshar, 2007; Larsen & Wilber, 2005; Zwaan & Steenbakkers, 2007), the lifecycle can be divided into the phases as shown in Figure 8. Each phase consists of various activities. To perform these activities, the actors need information from earlier activities. For example, the testing activity needs information about the service design which is stored before.

Changes in business needs or technical developments may cause a request to change the service. During its lifecycle, the service encounters several change requests, which may improve its value when implemented. The service is then evaluated whether its value is high enough to invest in changing it.

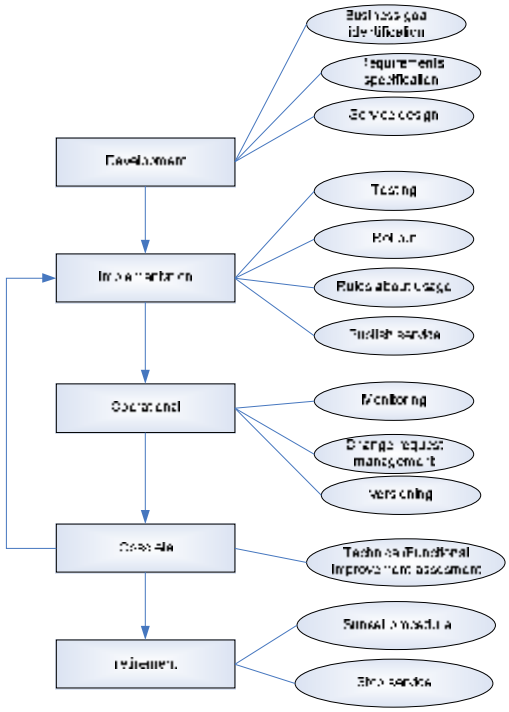


Figure 8 Service lifecycle

The service register can support a service's lifecycle by providing information about the status of the service in the lifecycle, change requests or historic data. To do so, additional information about a service should be stored in the service register. To support the information needed for service lifecycle management, another information type is introduced, the 'supportive information'. Supportive information of a service is information directly linked to a specific service; however it does not contain specification information.

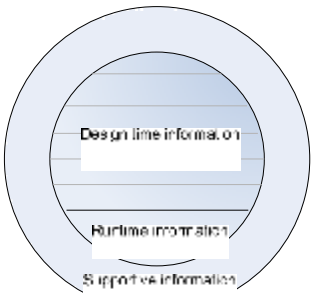


Figure 9 Representation of (service lifecycle) supportive information

3.5 Management view

The management is involved in the positioning of the service register within the organization and is mostly concerned with reports. The management can also have a 'financial sponsor' role, e.g. when the organization makes use of an internal costing system.

Services offer functionality that can be used by the organisation. This is comparable with the usage of software in an organization, which also provides functionalities to users. In analogy with the application governance in an application portfolio, services may be governed in a service portfolio for a manager to be able to measure and steer. This knowledge of which services are available and/or desired in an organization gives the ability to minimize the costs associated with deficiencies such as redundant services, services which are not used etc. In Figure 10, based upon software registrations (Zwaan & Steenbakkers, 2007), differences are showed that may occur. For example, the arrow between administrated services and used services represents an administration difference. If there are more services used than administrated, the costs of maintaining the services are higher than expected. Goal of managing the services in a portfolio is to minimize the differences, and therewith the costs.

Ideally, all services which are bought or made are created according to standards (so that they can be integrated easily), published (available for use) and used (no unused services) with no redundant functions (no extra costs of maintaining a functionality twice), while having the optimal contracts (for example discounts of costs-per-invocation). To be able to steer the IT in reaching this optimal situation, the service register creates a central place where this information (contracts, status, service consumers, standards etc) is stored. Along with organizational policies, this is essential to perform at high level at the service portfolio and minimize costs.

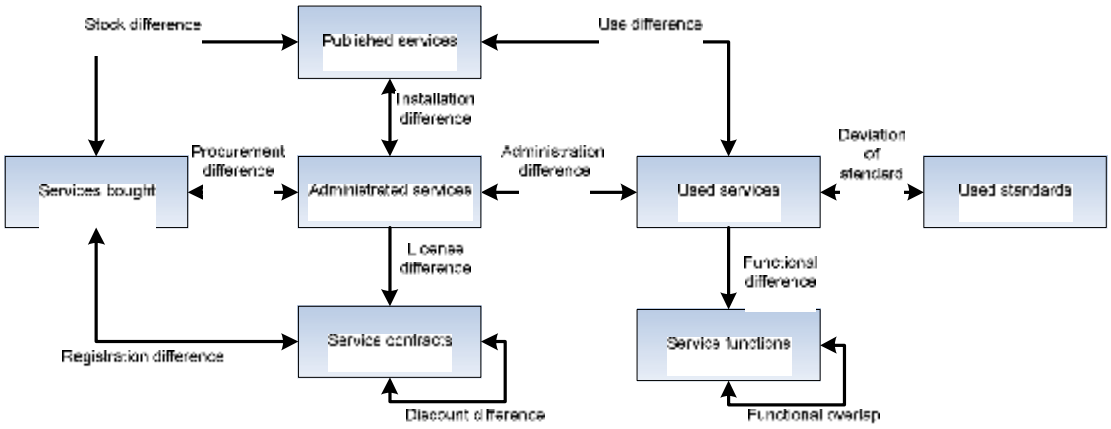


Figure 10 Service registrations, adapted from (Zwaan & Steenbakkers, 2007)

Portfolio information of the service specification is considered to be part of the supportive information as it is directly linked to a certain service specification. It supports the organization in making decisions, by containing information such as:

- With what licenses the services are used
- Who are the consumers of any service
- Who is the owner of the service
- Is the service created following standards set by the organization.

Implementation of this information is subject to the chosen service register implementation and the goals of the service register. Furthermore, management should decide whether they want to use the portfolio management method.

In addition, the management is interested in information about the use of the service register itself, especially when the users of the service register system pay for the possibilities of the service register.

This type of information, not directly linked to a certain service specification, is called 'Coordination Information' and contains information about the number of users, related costs etc.

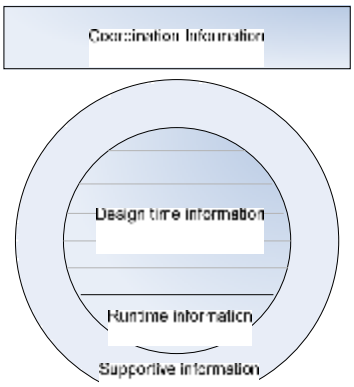


Figure 11 representation of coordination information

3.6 Maintenance view

The maintenance view can be divided in maintenance of the register as an Information System, and the maintenance of the (quality of) service specifications that are stored in the register. The first group is concerned with processes for functional, application and exploitation management of the service register software system. The organization wherein the service register is implemented might have existing structures to perform these maintenance tasks in a structured manner. The service register can facilitate their work by providing relevant log files and other information about the system, depending on the requirements set by the organization.

This information is needed for the exploitation of the service register. This 'exploitation information' is not directly related to a certain service specification, and is thus not part of the core information or supportive information.

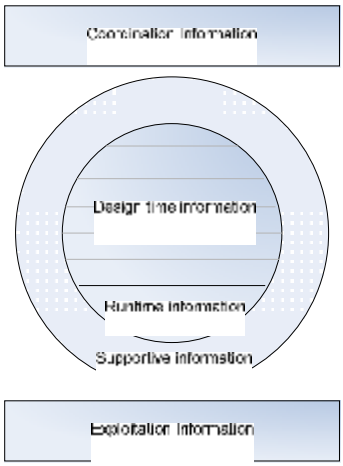


Figure 12 Representation of exploitation information

The second group of users is concerned with the quality of the service specification, i.e. how well the real service is represented by the service specification. The quality of the service specification is important to ensure that the consumer has the right expectation of the service, so it can be included in its process. These users concentrate their efforts to improve the core information and supportive information of the consumer and creator views.

A way to improve the matching of service requests with service specifications is to store the query and results of the requests in a knowledge base. Using the results of the previous search action, new search actions can be improved by learning from earlier results. By assessing the way how users are searching for information, the service specification maintenance can improve the service specification to be found.

As stated before, the terminology level of a service specification describes terms of the service specification as used in other layers (see Section 3.3) When combined, these terms can be used to create a domain ontology. This ontology can be used for searching or determining the relevance of a search query. The creation and maintenance of such ontology cannot be done entirely automatically, but requires user involvement.

The maintenance of the service specifications is a key factor for a successful service register. Attempts to create a public service register have proven that some control over the content has to be put in place. If not, consumers cannot rely on the service specification and are distrusting to use the service.

3.7 Goal of the service register

The discussion of the main actors is combined in the goal of the service register. The primary responsibility of the service register is to store service specifications which can be accessed for design purposes and runtime inclusion in business processes. Additionally, the service register maintains information about services that can be useful for other purposes (for example service lifecycle management), which must be retrievable.

The goal of the service register can be divided in two categories of processes. There are some essential processes, which define the basic goal of the service register to store and disclose service specifications to users. Other processes are optional to include in the goal of the service register. These processes support systems or tools with providing extra information. We briefly discuss these processes.

Essential processes

- *Facilitate the storage of service specifications* – The service register stores the service specifications and make sure that the services are described uniformly.
- *Facilitate the managing of service specifications* – The service register provides the possibility to adjust/update the specifications of services.
- *Facilitate the disclosing of service specifications* – The service register responds to service specification requests by different types of users. Technical access data for runtime users and meta-information for designer purposes.
- *Facilitate service lifecycle management* – The service register functions as a central place where information about the lifecycle of a service is stored.
- *Facilitate service portfolio management* – The management of services in a portfolio is supported by generating reports and service specifications storage.

- *Facilitate maintenance processes* – Provide maintenance users with the information which is needed for system maintenance.
- *Authorize access to service register System by actor roles* – The service register makes sure that it is only accessed by authorized actor roles.

Optional processes

Optional processes are useful processes that add extra functionality to the service register. These processes are included in our approach.

- *Support the SLA's by storing SLA information* –Part of the service specification is SLA information, which can be retrieved for evaluation purposes.
- *Support the storage of information about the service consumers* – Storing information about the service consumers is useful for analysis purposes, and enables the creators to contact the consumers in case of changes.
- *Support an invoice process by storing information about costs of a service* – The costs of using a service are stored with the service specification to be able to calculate usage costs and provide this information to a invoice process.
- *Support management reports* – Management reports use logged data about changes, invocations etc to create reports.

Excluded processes

These processes are identified to be useful; however they have a great impact on the design of the service register and are not included.

- *Support the coupling with other service registers in peer-to-peer network of organizations* – Storing service specifications at a single central place is in contrast with the SOA view of distributed systems. Distributed service registers are particular useful in situations with distributed networks, for instance in a chain of organizations.
- *Support service usage regression* – Service specification might be coupled with business process engineering tools, enabling statistics of which service is used in which process.

These processes show what the service register system has to support.

3.8 Actor roles for views

For each of the four main actor groups, actor roles are identified which are directly involved with the service register and are depicted in Table 3 below. It is likely that some of the actor roles are done by the same actor instantions, for instance the combination of subscriber and runtime user may be done by a single organizational unit.

We keep the granularity of the actor roles low for transparency reasons. For example, we acknowledge that the producer role of a creator can in fact be split up into several roles in service lifecycle management. However, we are primarily interested in the interactions with the service register.

Table 3 Actor roles

Actor group	Responsibilities	Actor role name	Description
Management	- Providing financial and authority resources - Steers the service portfolio management	Manager	Manages service portfolio, measures, reports
		Sponsor	Makes (financial) resources available
Creators	- Provide a working service - Provide SLA - Submit the service specification to the service register maintenance - Updates service version information in the service register - Notify the consumers of the service when changes occur, using consumer data in service register - Updates contact information in the service register	Producer	Creates services according to standards.
		Submitter	Submits asset to register maintenance
		Maintainer	Updates services according to lifecycle management, maintains quality
Consumer	- Subscribes to the service, therewith committing to the SLA and costs - Queries service register for service specification for each use - Notifies service creator in case of malfunctioning service using creator data in service register - Updates contact information in the service register	Subscriber	Subscribes to service, receives notifications
		Runtime service consumer	Searches register for technical information
		End-user consumer	Searches register for services, requires information at different levels
Maintenance	- Ensure a common taxonomy of specification data - Reviews the submitted service information, check for redundancy according to organizational policy - Periodically checks if services are up - End-responsible for changes in service specifications	System maintainer	Uses system information to perform maintenance tasks
		Coordinator	Organizes the service register efforts and activities
		Reviewer	Verifies services, checks quality
		Librarian	Publishes services, creates taxonomy, measures

3.9 Service specification matching

The most important function of an Information Intermediary, and in our view of the service register, is the matching of a (human) request with available content. This section explores appropriate matching techniques.

3.9.1 Data matching solutions

The main purpose of the service register for end-user consumers is to answer correctly to requests for service specifications. In order to do so, the request for a service specification has to be matched with the stored service specifications. In terms of Information Intermediaries, the request for an Information Good has to be matched with the stored Information Goods to provide the user with the best possible result. A mediator helps the matching of the Information Good with the request of the user to improve user satisfaction and matching.

The matching of data has been a research topic for a long time. Although focused on the integration of various data sources, Wijnhoven et al (Wijnhoven et al., 2003) distinguish different useful ways of matching solutions. The relevant matching solutions are shown in Table 4. Research shows that the use of an ontology based mediator facilitates the user satisfaction by giving more correct results.

Table 4 matching solutions, based upon (Wijnhoven et al., 2003)

Matching solution	(Additional) Advantages	(Additional) Disadvantages
Database	- Useful for simple reference base - Technically easy to implement. - Creates a central place for common storage of description	- Low flexibility in querying
Mediator	- No need for transforming source systems data. - Connects different domains.	- Technically rather complex to realize because of variety domain. - Executing queries may give high loads.
Mediator with ontology	- The user is not constrained by a given language, and is able to specify requests in his own language. - Each group of users may be served by their own language and worldview. New users are easily connected via the ontology.	- requires the creation and maintenance of ontologies for source systems.

The use of ontologies for matching a service request with a service specification is well understood in literature, especially when focused on automated matching in the semantic web (Alevizou & Plexousakis, 2006; Pokraev *et al.*, 2004; Studer et al., 2007), where it can be used to add additional information to the service specifications or specify different domain ontologies.

In the following section, the ontology matching is further elaborated upon.

3.9.2 Ontologies

An ontology is an explicit description of concepts, its properties and restrictions to define a (machine-interpretable) vocabulary (Noy & McGuinness, 2001). To deal with different users and the application data itself, multiple (slightly) different information domains are to be brought together, as different users of the register have different information domain subsets. Bringing different vocabularies together is both difficult and necessary. To create a common information model, the different ontologies have to be mapped to each other to make it possible to match a request with stored information.

Mapping of different ontologies can result in conflicts at naming and structural level. Naming conflicts arise from different people (or applications) using the same term to mean different things or, reversely, from people using different terms to mean the same thing. Structural conflicts arise when conflicting modelling constructs are used to represent the same concept (Vrijkorte, 2006). However, the creation of qualitative domain ontologies which describe services is difficult and costly.

As it is unlikely that different users have the same conceptualizations of a concept, mediating between different conceptualizations is important to get the best results of matching with stored data (Wijnhoven et al., 2003). Within the Knowledge Modelling research, ontologies have been developed to facilitate knowledge sharing and reuse. They can be used to communicate this knowledge between people and systems (Cabral et al., 2004). When ontologies are used for knowledge representation, they can be divided into four categories (Jurisica *et al.*, 2004):

- Static ontologies, describing static aspects of the world, i.e., what things exist, their attributes and relationships.

- Dynamic ontology, describing the changing aspects of the world in terms of states, state transitions and processes
- Intentional ontologies, describing the world of things agents believe in, want, prove or disprove, and argue about.
- Social ontologies, describing social settings; agents, positions, roles, authority, permanent organizational structures or shifting networks of alliances and interdependencies.

In the context of service registers, ontologies facilitate the common understanding of concepts by describing 'static aspects' of the world (the services), and therefore mainly fall into the first category. A more advanced service register would require intentional ontologies to reason about the relevance of a specification (e.g. in the semantic web).

Literature identifies several techniques for representing knowledge. They can be divided into five main categories; First Order Logic, Semantic Networks, Frame-based Logics, Description Logics and Rule-Based Logics. Implementations from these techniques are checked against requirements of modelling languages to see which implementation can be used best for knowledge representation in (Vrijkorte, 2006). This shows that Rule-Based Logics is the most appropriate technique for representing knowledge, although results of implementations are not known in theory.

Although its importance is evident, the creation of a common information model is difficult. Not only is the mapping difficult due to the conflicts described earlier, but the creation of the ontologies can not be fully automated. Although some first attempts are made based upon the description of the services, domain experts have to be involved in the creation (Sabou, 2005).

4 Information Intermediary design methodology

In addition to the concepts introduced in the previous chapter, this chapter explains the design theory that is used in this thesis. Section 4.1 gives an introduction to the theory. Section 4.2 discusses various categories of Information Intermediaries, that influence the choice for a design scenario in Section 4.3. Section 4.4 discusses the aspects of the Information Intermediary design in the service register context.

4.1 Design science for Information Intermediaries

In general, the design of an Information Intermediary has three main aspects (Wijnhoven, 2008). The first aspect is the delivery of the *content* to the user, and all associated functions (e.g. acquisition, aggregating). The second is the delivery of *use features* for a better use and value experience of the Information Good. The third aspect is the stream of *revenues* to be realized for the Information Intermediary to be viable for the content owners.

The third aspect, the stream of revenue, is not applicable for the design of a service register when used within a company with no intention to have a cost effective information system. If the service register has to earn its own costs (as in the case), revenue can be earned in several ways, which will be dealt with later on. The arrangement of revenues can lead to a more efficient and effective working Information Intermediary, and is recommended to include in some way.

To support these aspects throughout the design, a layered approach can be used to design the aspects at different levels. The top layer depicts why the Intermediary should be used, the second layer depicts what is needed to support the Intermediary, and the third layer depicts how to create an infrastructure to support it. Wijnhoven (Wijnhoven, 2008) divides these layers into:

- Design problem and agenda
- Business requirements
- Process requirements
- Infrastructure

Following these layers may result in a working Information Intermediary, but for a successful and enduring system, building a prototype and exploiting/evaluating are also part of the design layers.

It is important to notice that the layers as mentioned above are for both IT and Organization aspects. For example, the infrastructure layer does not only comprise the design of the IT infrastructure such as databases and networking, but also the structuring of employees and tools to support them.

The resulting design space consists of the three aspects supported by the six layers, is given in Figure 13. Each design, for each type of Intermediary and its instances, is highly likely to be different from others, due to the context the Intermediary has to be placed in. Therefore, when designing a service register as an Information Intermediary, this has to be abstracted to a meta-design. This gives an overview of the elements of the Intermediary to which an instance should comply, and does not contain any specific implementation details.

The eighteen blocks in Figure 13 formed by the crossings of the six rows (layers) and three coloured columns (aspects) indicate which areas have to be filled (or at least considered) when designing the Intermediary.

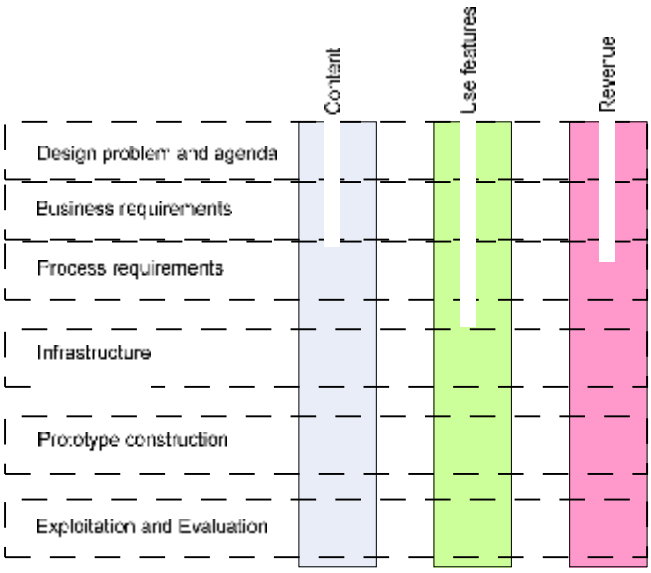


Figure 13 design space for Information Intermediaries, adapted from (Wijnhoven, 2008)

There is no specific order of considering the blocks, this is left for a specific design style. Top-down, bottom-up and middle-out can all be used, as long as the blocks are aligned with each other. The design layers in Figure 13 are grouped according to the chapters in which they are discussed in the next part.

4.2 Categories of Information Intermediaries

To deliver the right Information Good from the provider to the requestor, an Information Intermediary has issues with matching and maintaining the Information Good. The field of Information Intermediaries, backed by the more general Knowledge Management systems, has developed methodologies to try to deal with these problems.

There are different categories of Information Intermediaries, each one requires a specific focus on a design aspect of Figure 13. The assignment of the service register to a category can be important when making design decisions. The categories are (Wijnhoven, 2008):

- Corporate web-sites – *Communicate controlled messages to an audience, with a focus on the content aspect*
- Content aggregators – *Aggregate Information Goods of a specific area, with a focus on the content aspect*
- Community builders – *Helping the development of communities with meta data processing, with a focus on the use features aspect*
- Data integrators – *Consolidate data of different sources by a common data model, with a focus on both content and use features*

The revenue aspect is not a specific focus of a category, as the content and use features are essential for creating revenues. The focus is therefore on these aspects, and the revenue is created around them.

The service register is a central place for aggregated description of services, and could therefore be placed into the second category. However, it is more important that the various service specifications are stored with some sort of common data model. Therefore, a service register can be placed in the fourth category, while the second may have elements which can be of help. The consequences are discussed in the next section.

4.3 Design scenario

When designing an Information Intermediary, several design scenarios are identified which can be followed. Theoretically, each of the cells in Figure 13 can be taken as starting point, followed by another cell. This results in a shear amount of different design scenarios, from which Wijnhoven (Wijnhoven, 2008) extracted seven scenarios which are closest to existing design theories.

These scenarios vary in layer and/or aspect starting point and provide different ways of developing an Information Intermediary. Most of the scenarios provide designs in which (part of) the Intermediary is launched in an early stage to collect usage information to improve the design. These scenarios are not particularly useful when designing the service register, and although an iteration based on experiences is useful and should be done, the basic waterfall scenario is useful to follow. This explicitly does not mean that iterative steps are impossible to take. In contrary, the subsequent steps may result in other insights which make it necessary to adjust previous steps. As a central information system in the SOA architecture, the service register should function well right from the beginning, making risky design scenarios less appropriate.

The waterfall scenario follows the layers of Figure 13 downwards:

- *Business requirements* – What information good is transferred with which use features, and what are the perceived values
- *Process requirements* – What processes are needed to fulfil the business requirements
- *Infrastructure* – How to support the processes.
- *Construction* – Create the Information Intermediary
- *Exploitation* – Bring in production and analyse performance

This will also be the design scenario which is used in this research. The other dimension of the grid, the design aspects (content, use features and revenue), are further explained in the next section. The use of the waterfall model may give the perception that the different phases have to be completed before proceeding. However, this is explicitly not the intention, as the feedback from phases is used to improve the design.

4.4 Design aspects

4.4.1 Content

The content of Information Intermediaries is the Information Good (i.e. the service specification), supplemented with optional other delivery of information (the supportive information). These digital goods are difficult to match with the user's demand, as the sheer volume of variations of the Information Good make it difficult to present the user with a few choices. The Information Intermediary has therefore the task to reduce the amount of possible Information Good variations to present the user with meaningful information.

The Information Intermediary is aware of the content it is transferring. This means that the Intermediary has the control over the selection and provision of the content. By pre-selecting the content, the Intermediary provides a service to the user. The intermediary experiences costs (such as cluttering costs) to provide this service, for which the user is usually willing to pay.

Generally, an Information Intermediary transfers static content which the user consumes. However, the service specification in an Information Intermediary is in essence a piece of process information, used in business processes. A sound consequence is that the service specification should be of high quality and well-described. As previously mentioned, the use of defined and accepted standards is needed for high quality, and is a prerequisite for exchange of process information (Aalst & Kumar, 2003).

The Information Intermediary may transform the Information Good to meet the request of the user by delivering just what the user needs. It can do so by altering two of the dimensions of an Information Good, the Level of Representation (LOR) and the Level of Conceptualization (LOC). By increasing or decreasing either or both levels, the Information Good is 'customized' for a user.

Service specifications consist of both runtime and design time information, with added supportive information (see section 5.2.3). Not every specification detail is useful for all users, and therefore the LOR is increased to present the user with the information that he needs, e.g. only the interface description is presented as a result. Additionally, the user may be presented with information on how the service specification is used in other situations.

4.4.2 Use features

To increase the value experience (user satisfaction) for information users, use features can be provided by the Information Intermediary. Use features may be realised by delivering additional content interaction options such as advanced search options and extra about the content. Extra information may be the provision of quality perceived by others.

By providing use features, the Information Intermediary makes itself more attractive to use. This is specifically important when other service registers exist which may provide the same information good. However, this is not true in the case when a service register is placed within (a network of) organizations, which is the scope of this research. Providing use features like credence characteristics or quality judgements from other users (which may be helpful in deciding

whether to use the Information Intermediary) is therefore not necessary as the user has the obligation to use it.

Also the provision of experience characteristics (to 'test' the information good) will not be useful in a service register setting, as this is meant to persuade the user of using the Information Good.

However as the objective is to design a service register with a high end-user satisfaction, use features may be added to increase the satisfaction. Examples are to help the user with formulating his search query or provide the user with alternatives.

4.4.3 Revenue

The provision of an Information Good is costly. First, the Information Good has to be created and second, be available and transferred to users. Although the digital nature of an Information Good makes the distribution cheap, costs are made with creating, maintaining and providing the good.

As the quality of the service specifications is identified as a key factor, the maintenance of the service specifications needs extra human attention. Revenue streams have to be created to cover the costs of creators and the service register organization. To identify what the revenue streams are, a first step is to determine what institutional form is suitable for a service register.

Womack (Womack, 2002) identifies optimal institutional forms for Information Intermediaries which function in a certain environment. Based upon five 'tests', the optimal institutional form can be derived, and revenue streams can be identified. These five tests are discussed here for the service register.

1. Is the information of primarily private or of primarily social benefit?

The information in a service register is meant to be used by the consumer only. However, by using the information, the consumer is able to perform actions which are beneficial for the entire (chain of) organizations, i.e. the social environment. Therefore, the information which is transferred by a service register is of primarily social benefit.

2. Is the information transparent or opaque?

The distinction between transparent or opaque information is mainly the time span between retrieving the information and knowing what value it has for the consumer. The information which is retrieved from the service register can almost immediately be evaluated and is therefore transparent information.

Opaque information would require the provision of extra use features to the consumer to express the quality of the information.

3. Is the information provided by many intermediaries or few?

As there probably is just one service register in an organization, this question is answered with 'few'. More Information Intermediary would indicate that there is some sort of market at which the consumer may choose, but in a normal organizational setting, the choice of service register is established via power structures.

4. Is the information in demand by many or few?

Whether there are many consumers or not influences the level of customization of the Information Good. The information in the service register is meant to be useful for a lot of consumers. Although there may be few consumers of a specific Information Good, this might grow in the future and is focused on a demand by many.

5. Is the information useful to paying clients or clients who have little ability to pay?

As the consumers are most likely part of an organization with financial resources, they are likely to be able to pay for the use of the service register.

According to the classification of Womack (Womack, 2002), the optimal institutional form of a service register would be "Antitrust regulation + subsidy to client; or non-profit/government provision". As the setting of a service register is restricted to use in an organization, antitrust regulation and government provision is not applicable, leaving subsidy and non-profit institution. This classification is relevant if an organization decides to arrange revenue streams for the service register. It shows that the organization should create a unit that is run as a non-profit organization.

The above answers of the questions are general choices, leaving the option for other outcomes in different situations. The resulting form, a non-profit institution with a possibility for subsidization, gives a guideline for how to position the service register in an organization. For revenue streams, this means that the organization provides subsidy for the consumers, while an organizational unit runs the service register as a non-profit organization.

Running the service register as a non-profit requires the consumers to pay for their use of the service register. Possible payments can be based upon subscriptions or actual use of the services of the service register. This is further discussed in the section about the business layer, Section 5.2.

5 Service register requirements

This chapter defines requirements for the processes which are supported by the service register. This research focuses on delivering the right Information Good to the user, and providing the actors with the information which they need. Therefore, the functional requirements specification is focused on the interaction of the actors and the quality requirements for maintaining the register.

We define the requirements at a meta-level, i.e. the global requirements for an architectural design which is created in Chapter 6. The component for interaction with the end user is then further specified in Chapter 7 3.4, as this component is important, though often missed in current service register implementations.

Based upon the goals of the system, requirements are identified which are to be implemented to reach the goal of the system. The case mentioned in section 2.3 serves as reference case. As the case is fictional, there is no information about the exact technical environment.

This chapter discusses requirements of the service register and discusses the first three layers of the design methodology as discussed in Chapter 4.
Section 5.1 discusses the design problem and agenda that is to be set before the design project is started. Section 5.2 discusses the business requirements for the service register. It states what values are exchanged for what returns and defines information requirements set by the user. Section 5.3 determines what is needed to support the business requirements with processes. Section 5.4 briefly discusses additional requirements that are not covered in previous sections.

5.1 Design problem and agenda

As the service register is part of the SOA architecture, roughly speaking the same stakeholders are involved with the service register. As is the case with all Information Systems implementations, the stakeholders should have a general positive attitude towards the system and acknowledge its added value. To communicate the necessity with the stakeholders, a causal tree such as presented in Figure 14 can be used.

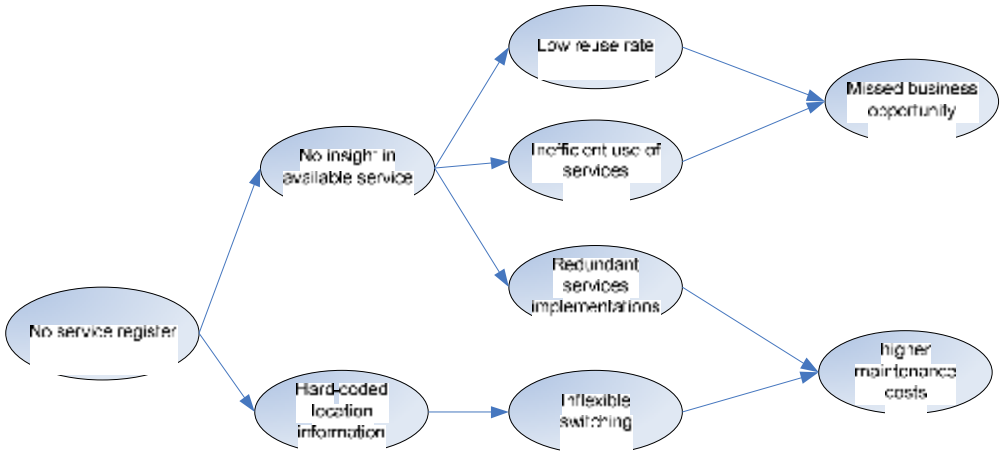


Figure 14 Causal problem tree; showing why a service register is used.

The four main stakeholder groups (creators, consumers, management, and system/information maintenance) are to be identified in the organization. The stakeholders have to be aligned with each other and the goals of the service register, which can consist of the primary processes supplemented with any of supportive processes as explained in Section 3.7. A causal tree like Figure 14 can be used as a tool to create a positive attitude towards the service register by the stakeholders by showing the consequences of not implementing a service register.

5.2 Business layer requirements

5.2.1 Introduction

The second layer of the design methodology indicates the values that are exchanged with the service register. The interactions with the service register generate requirements for the design of the service register. This Section first identifies the values that are exchanged and then list functional requirements for the service register.

5.2.2 E3value modelling

The business layer depicts the values which are transferred between stakeholders. The service register should be able to enable value exchanges between stakeholders to satisfy the stakeholders, so that the service register is used independent from obligatory regulations in the organization. A value exchange can be either of the design aspects *content*, *use feature*, or *revenue*, and specifies what stakeholders deliver in return for what.

The business layer is about *what* content, use features and revenues are exchanged for what reason by which stakeholders. How these exchanges are realized by the stakeholders is hidden at this point. The exchanges between the stakeholders can be modelled using the e3value method of Gordijn and Akkermans (Gordijn & Akkermans, 2001). The resulting business model shows only interactions and no internal processes.

In addition to the stakeholders who are directly involved in exchanging content, the *revenue aspect* as identified in Section 4.4.3 shows that the presence of a sponsor is needed. Overall the exchanges of revenues should result in a service register which costs are covered. To test if this is true, the e3value model can also be used.

There are several ways in which the exchanges can be modelled (for instance, choices can be made how the subsidy is organized), the following value model in Figure 15 shows one of the possible ways. In this high level scenario, the sponsor (which may be the management itself) subsidizes the service register directly. In a more detailed model, the value exchanges can be further specified, and also internal value exchanges can be modelled. In another possible model, the subsidy is given to the users, giving the service register more incentives to perform efficient, as users are to be satisfied in order to make money.

The e3value model in Figure 15 has four value exchange starting points. They follow a value path, depicted by the thick (red) line. Firstly, the creator may provide a service specification to the service register, which is willing to pay for the specification in return. Secondly, the runtime user can initiate a value exchange, which is directly split into separate value paths. The runtime user pays a specification fee, and gets a (part of) service specification in return. The sponsor is willing

to pay for (part of the) specification fee, for instance in return for feedback. The sponsor may also offer its subsidy purely altruistic, the specification fees of the consumers are then not existent.

The end-user also pays a specification fee and gets several value objects in return. First the service register offers various use features and search assistance to help the end-user with retrieving the design specification. Second, the service register delivers a specification. The delivery of the specification is the core value of the service register, and its task is to deliver the specification as good as possible. To deliver the specification and meet the expectations of its clients, the specification should be complete, of high quality and truly represent the real service.

As this high level model does not include the internal value offerings of the service register (e.g. by maintenance/support personnel), the net profit can not be determined with this specific model.

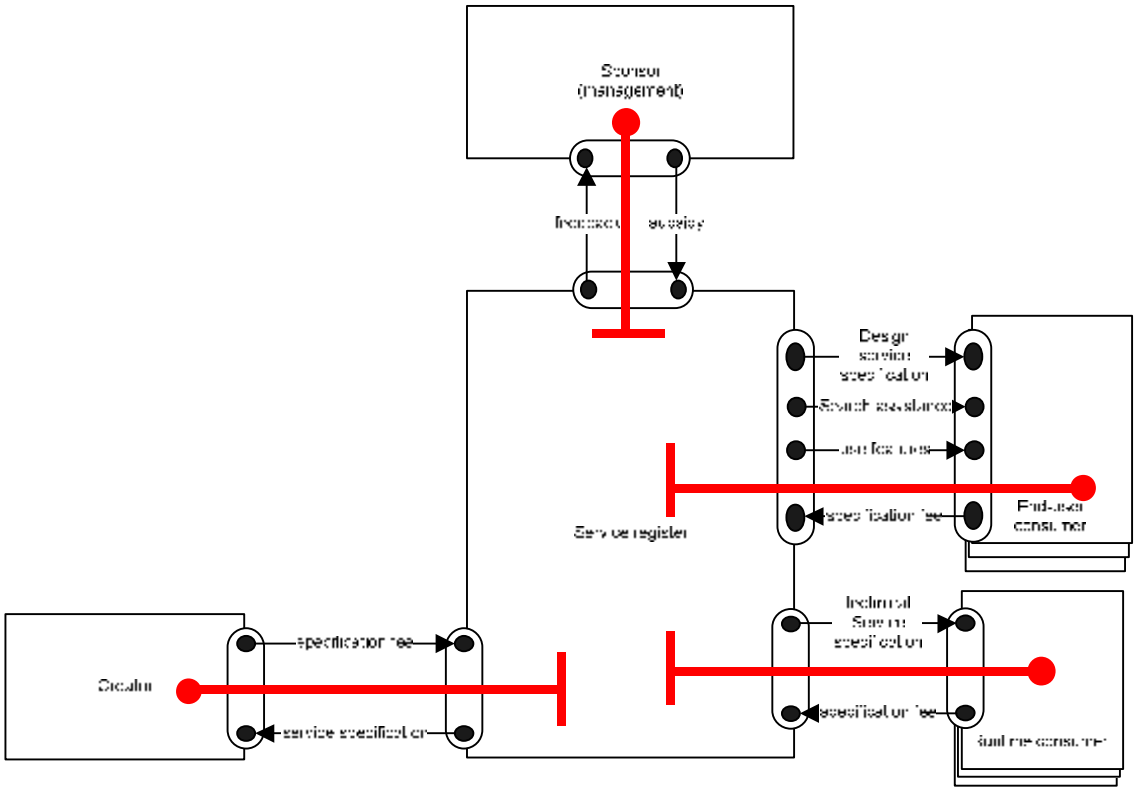


Figure 15 high level e3value model for "Sure 4 U"

The exchanges are depicted by arrows between the stakeholders and the service register. As stated before, the internal exchanges (e.g. the maintenance/support) of the service register are not shown. The exchanges represent the three main aspects of an Information Intermediary design, content (e.g. "service specification"), use features (e.g. "search assistance"), and revenue (e.g. "specification fee").

5.2.3 Actor interaction requirements

The actor roles that are defined in Section 3.8 and the discussion of the actor groups depict information needs of the users of the service register. The service register has to fulfil these information requests, and they are therefore functional requirements of the service register software system.

Use cases are a main source of extracting requirements of an Information System. By assessing each identified actor role for their information needs, use cases are created which describe the expected behaviour of the service register. This research does not discuss these use cases in depth.

Important information sources for this assessment are existing written sources, for example procedures or log files, and interviews with the actors, which are to be done when determining the requirements for a real implementation case (Hepworth, 2004). However, this is not applicable in this research

Assessment model

Use cases can be derived using various methods. The traditional, scenario driven approach can be applied to elicit use cases. A variation of this approach takes the responsibility of an actor and checks what interaction takes place in fulfilling the responsibility.

As developed in the discussion of the main actor groups in Section 3.2, users of a service register demand different types of information:

- Coordination information (e.g. management reports, statistics)
- Exploitation information (e.g. error logs)
- Core information
  - design time information (e.g. purpose of service, goals)
  - runtime information (e.g. interface, location)
  - supportive information (e.g. owner, creator, consumers, lifecycle information)

The different information types are developed in Chapter 3 and again graphically represented in Figure 16. The core information (which is essentially the service specification complemented with other, supportive information) is the central Information Good. The other information types are put in place to facilitate and coordinate the provision of the service specification. A data model is provided in Section 6.2.2.

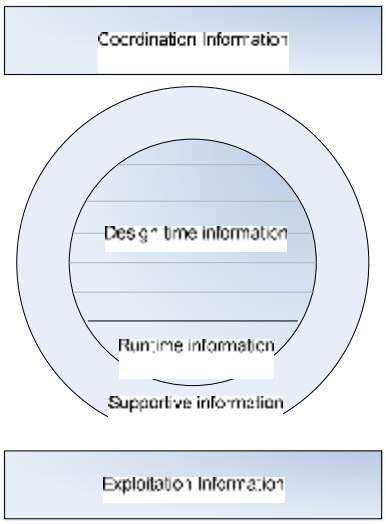


Figure 16: Graphical representation of information types

Both coordination and exploitation information provide information about the service register as a system, whereas the core information is the information good itself. Actors may have information needs for these information types. Hence, combined with the actors which are identified earlier, this leads to the framework in Table 5.

Table 5 actor interaction assessment

User / Information type		Management	Creators	Consumers		Maintenance	
				End-users	Runtime	Service register maintenance	Information Good maintenance
Coordination							
Exploitation							
Core	Design time information						
	Runtime information						
	Supportive information						

The shaded elements of this framework indicate an information need and are briefly discussed below.

Management

The manager is primarily interested in how the service register is used and not in the service specifications itself. Therefore, the information need of the management is limited to the coordination type of information, consisting of various management reports. Which exact reports

are generated varies over the implementations, but may include usage reports to determine the usage and the costs for consumers of the service register.

Furthermore, we can determine functional requirements from the discussion of the management view in Section 3.5. The management of a service portfolio requires the service register system to provide several reports. Although not intended to be complete, the following reports are needed:

- Which services are administrated
- Which services are published
- Which services have what contracts
- Who are the providers of the services
- How many users do the services have
- What are the functions provided by the service
- What standards are used for the service

Creators

The creator has different roles and responsibilities. A primary responsibility is to provide the runtime consumers of the service register with a service. Two main aspects are important in providing the service. First the service must be up and running and reachable, and second the service should behave like it is specified in the service register for the satisfaction of the user. From this view, the three roles as described in Table 5 are discussed.

The *producer* implements the service according to the needed functionalities and standards. To do so, it might need information from the service register to determine within what boundaries (standards) the service has to be implemented. The producer is further responsible for testing of the service and creating a service specification for both the design time and runtime information.

The *submitter* is a first review step before the specification is made available in the service register and is the communication channel for changes in the service register. The submitter checks whether the service specification meets the service register requirements, and eventually submit the specification to the service register.

The *maintainer* needs access to the service specification in the service register for both design and runtime information. He might also use supportive information to retrieve information about the number of different runtime consumers, the status in the lifecycle or history. When the maintainer decides to change the service, he is responsible for updating the service specification, and to submit the specification via the *submitter*.

We discussed in Section 3.4 that the activities of the creator for the service lifecycle are to be supported by the service register. The service lifecycle starts with the identification of the service and ends when the service is discontinued. The service register supports the lifecycle by storing information about the service and create reports. Requirements of the service register are shown (although not meant to be exhaustive) in the following sequence of service register system activities to support the life cycle.

- Create a new service specification
- Change a status of an service specification (possible statuses include 'to be reviewed', 'requires a requirement specification', 'requires approval', 'completed development', 'is tested' etc)
- Create lists of service specifications of a creator depending on status
- Alter or add information to a service specification

Consumers

Consumers are essentially the end-users, the users which actually use the information good in the service register

The *runtime service consumer* is typically an organization or organizational unit which desires to use the service which is described in the service specification in the service register. This organization is most likely represented by a software agent (an application which will eventually use the service) who queries the service register for a specific service description. Therefore, the runtime service consumer is primarily interested in the runtime components of the service specification.

The service specification may change in the service register. This may affect the working of the service, and hence the working of the application which invokes the service. To prevent the change of the specification from going unnoticed by the runtime consumer (and therewith an unreliable end-application), the runtime end-user should be notified when a change occurs. To facilitate this, the service register may choose to store contact information as a *subscriber* of the service. The creator can use this information to contact the subscribers.

The *end-user consumer* is typically a human user, who uses the service register to retrieve service specifications. This user has other information needs than the runtime user, and is likely to need e.g. goal, purpose or condition information. A plausible usage scenario is a business architect which needs a certain service and checks whether there exists a service which meets his requirements.

Following the levels of service specifications in Section 3.3, the service register supports the following:

- Retrieve marketing information of a service specification
- Retrieve task information of a service specification
- Retrieve quality information of a service specification
- Retrieve interaction information of a service specification
- Retrieve behaviour information of a service specification
- Retrieve interface information of a service specification
- Retrieve definition information of a service specification
- Search in all information layers

- Support the search process

Maintenance

As discussed before, the maintenance users can be split up into the maintenance of the service registry system itself and the maintenance of the stored information good, i.e. the service specification

The maintenance of the service register system itself is done according to the available maintenance structures (ICT organization, outsourced maintenance) in the organization. For these structures to function, information about the service register is needed to monitor the system. Therefore, the service register system should keep a log wherein relevant maintenance data is kept. What exact data is kept depends on the maintenance structures.

As the *content* is one of the factors which determine the end-user satisfaction and should be of high quality, other maintenance users are assigned to make sure that the service specifications are well-specified.

The *coordinator* is a central role which organizes the maintenance efforts. In a discussion of the future of SOA, Natis et al (Natis *et al.*, 2006) acknowledge the importance of having a central entity to administrate the coarens of the service register. This role, comparable with a database administrator, coordinates the activities around a service register system

Activities of the coordinator can be compared to the activities around a Configuration Management DataBase (CMDB, used in the ASL maintenance processes to ensure a standard way of storing configurations of applications). An important task is to set out the standards and conventions which are to be used when specifying the services, and communicate this with creators and consumers.

To improve the results from the end-user queries, the coordinator may use search logs to improve the matching of the end-user search request.

The *reviewer* receives the service specifications submitted by the submitter of the creator and checks if the specification meets the requirements set by the coordinator. Furthermore, the reviewer may check whether the specification is a true representation of the service itself, so the user has the right expectation about the behaviour of the service.

The *librarian* is responsible for the entries in the service register and can be consulted by end-users in need of assistance. A further task may be to periodically check the correctness of service specifications or when notified by the system.

The service register should thus support:

- Storage of guidelines and requirements of specifications that can be used by creators
- Retrieve the guidelines and requirements
- Logging of system actions and search queries and possibilities to retrieve the logs
- Review procedure
- Alter service specifications

- Notification

5.2.4 System interactions

Although the primary goal of the service register is to interact with users which demand information of a specific service specification, it also should be able to provide supportive information to other systems. These interactions are optional, as they are not part of the essential part of the service register. However, as a service register probably provides one or more such services, it is taken into account here.

Interactions with other systems are offered by the service register via services. The context diagram in Figure 17 (on the right side) shows (some of the) systems with which the service register interacts. Which systems are to be connected depends on the specific implementation. To stay in line with the SOA architectural thoughts, the connecting lines represent a communication via the SOA infrastructure for loosely coupling.

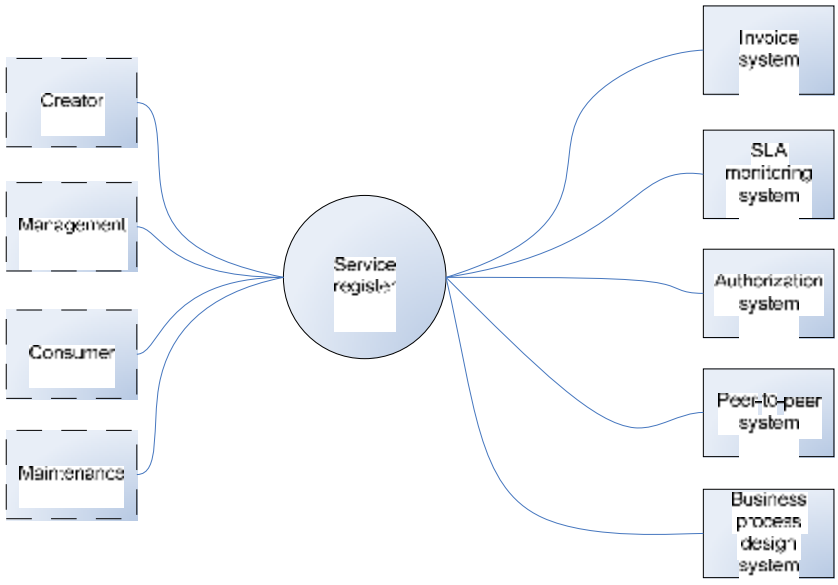


Figure 17 Context diagram service register

5.2.5 Use cases

From the interaction assessment above, a use case diagram for the exemplary case can be derived which is shown in large in Appendix B. From these identified use cases, the use case "search for design service specification" is taken as focus. This use case is the interaction between the service register system and the end-user, and therefore the interaction which is important to achieve user satisfaction. This use case is further specified in Section 7.1.3.

5.3 Process layer requirements

Where the business layer does not define internal processes, the process layer defines activities required to deliver content, facilitate use of the content and to process transaction costs. We discuss these three activity groups here, and focus on the interaction with the end-user.

5.3.1 Deliver content

A service register delivers a service specification to a consumer. It does so by aggregating the available service specifications and stores these specifications. Consequently, the service register should have stored all content necessary to answer the request of the user. Therefore when answering the request, no connections have to be made with creators or other actors. However, when the organization wherein the service register is functioning is part of an interconnected chain of organizations which work together, the service register can reroute the request to another service register when it is not able to answer the request. As this is not the case in "Sure 4 U" this is not included in the following.

The content is thus delivered by the service register using available service specifications. The delivery can be customized to meet the request of the human end-user, for instance by increasing the level of representation. The service register is than not just delivering the entire service specification, but makes a selection of the available information to present to the end-user. When possible, it can also increase or decrease the level of conceptualization by adding application domain information.

The content is delivered by the creator to the service register. The end-users demand a high quality of the service specification, to achieve a high end-user satisfaction. Therefore, before the service specification –or change in the specification– is added to the content base, the service specification follows a series of activities to ensure that the content is meeting the standards set by the service register.

5.3.2 Use content facilitation

Along with the content, use features are delivered to increase the factors which lead to a high end-user satisfaction. Recalling the factors which lead to end-user satisfaction, the focus is on *Content*, *Accuracy*, *Format*, *Ease of use* and *Timeliness*. Activities to support the use content facilitation are:

- *Content* – The server register has to deliver the precise information needed by the user. Therefore, the needs of the user have to be determined, e.g. by helping the user to formulate his information need. The service register is than able to deliver the exact content, and at least provide sufficient information to fulfil the needs.
- *Accuracy* – A related factor to the content is the accuracy with which the service specification is delivered to the user. An accurate delivery is the presentation of the best matching service specification(s) to the user. The service register can increase the accuracy by providing alternatives to the users. Using ontologies, the service register can determine the relevance of the presented results and suggest other specifications which are close to the presented ones.
- *Format* – The presented results are to be showed in a clear and useful format. The service register can assist this by increasing the operational use, e.g. by providing different formats so the user can choose the appropriate one. This means that the same information is presented in different ways, e.g. in HTML, XML, .pdf, .xls etc.

- *Ease of use* – The ease of use is increased by providing user friendly interfaces with help functions and the use of standard information presentation.
- *Timeliness* – The timeliness is influenced by different factors. The technical factor determines how fast a request is processed by the service register system. The speed of delivery is subject to the available computing power of the system. The other factor is how old the information is, and thus how trustworthy the information is. Where the first factor can be influenced by the design of the service register system, the second can be influenced when results are presented to the user. When the service register system determines that the specification has exceeded a specified period of time, maintenance personnel can be signalled to check the specification.

The various use features are added at the same time the content is delivered, and are thus enriching the experienced value of the content.

5.3.3 Transaction processing

The transaction processing is –depending on the chosen subsidy way- relatively clear. The revenue processes to be used are the standard subscription and pay-per-unit methods. Other transaction processing possibilities (e.g. advertising, syndicating, additional merchandise, pay-per-data-packet, see (Wijnhoven & Kraaijenbrink, 2008)) are based upon a more open market for content delivery, which is not applicable in an organizational setting. Activities to support the transaction processing are thus to check the subscription status of the user in the case of the end-user.

5.3.4 Process model

The process model in Figure 18 shows the process requirements described in the previous sections. It assumes that the maintenance of a service specification is part of the service register unit.

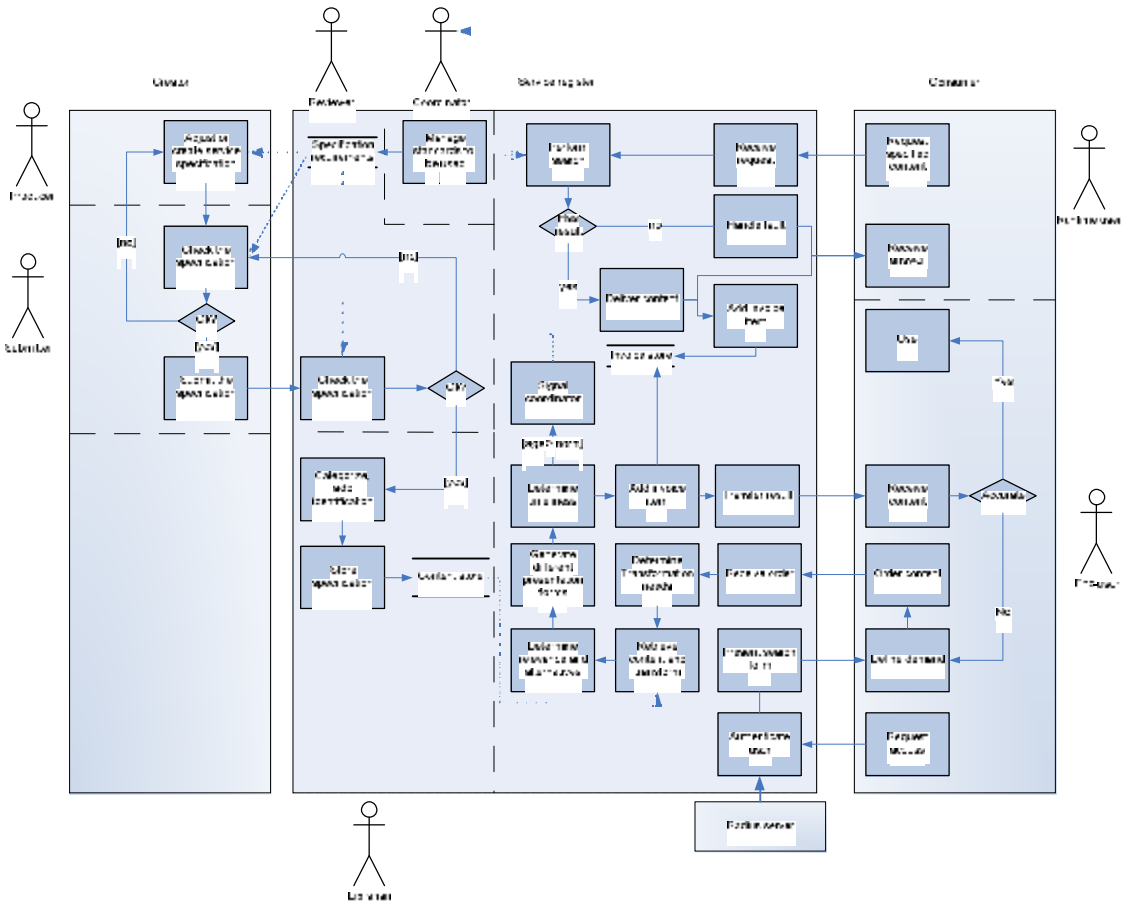


Figure 18 Process activities chart

A slightly larger version of the process chart is found in Appendix A. The relevant actor roles here are put on the side of the figure, and have their own part of the model which is depicted by the striped lines. Dotted lines indicate a stream of information whereas a solid line depicts a process step (which may contain information elements). More detailed process steps can be drawn by zooming in on process elements

This process model only shows the delivery of content by creators and to the consumers. Other processes do not have our main attention, but can be drawn in a similar way. In the Archimate model in Section 6.3, these processes are also mentioned.

5.4 Additional requirements

Next to the functional requirements which are discussed in the previous section, an Information System has additional requirements. The FURPS+ method for requirements determination distinguishes six different requirements areas (Larman, 2002), which is in essence an abstract of the ISO norm 9126 for software quality. The Functional requirements are already discussed, the Usability, Reliability, Performance, Supportability and the '+' (other requirements on hard- and software, documentation, operational issues and business) will be briefly discussed in the next subsections, and are meant for a quick scan when implementing a service register

5.4.1 Usability

The usability requirements specify how easy the system can be used. These requirements specify how the functionality of the system is perceived by the user in an interaction. In (Lauesen & Younessi, 1998), five factors form the usability (or ease-of-use) of a system:

1. *Ease of learning* - The system must be easy to learn for both novices and end users with experience from similar systems.
2. *Task efficiency* - The system must be efficient for the frequent end user.
3. *Ease of remembering* - The system must be easy to remember for the casual end-user.
4. *Understandability* - The end-user must understand what the system does.
5. *Subjective satisfaction* - The end-user must feel satisfied with the system.

Where satisfaction is identified as a factor of usability here, it can also be said that the satisfaction depends on the other factors of usability, which thought is adopted here.

The aforementioned factors are focused on the human-system interaction, where the service register also has to deal with usability for system-system interaction. A possible requirement resulting from this interaction is to use certain standards.

5.4.2 Reliability

A reliability requirement specifies a required amount of reliability, the degree to which something operates without failures. The objective of such requirements is to ensure that the system is working properly and minimize disruptions.

The definition of a 'properly working service register' depends on the chosen form of the register. For a service register meant to support only the design processes, the reliability is less important than for a service register which is integrated in a business process.

Typical reliability requirements include the (un)expected downtime, response time and crash tolerance.

5.4.3 Performance

Performance requirements define how well the service register performs its tasks in speed and abilities. In terms of a service register requirements are formulated, for example defining how fast a certain type of request is handled, percentage of allowed wrong handling, capacity norms and how up-to-date the data is.

5.4.4 Supportability

Supportability requirements define how the service register is supported. For example, requirements may be formulated which define the consequences of regular maintenance in downtime or roll back procedures.

5.4.5 The '+' requirements

The '+' requirements of the FURPS+ method (Tarman, 2002) intend to describe requirements to support the system.

Functional management

The functional management procedures are steered at organizational level, not on a per-system basis. The service register has to fit the existing functional management structures, and therefore support the decisions which are made. The most important requirement on this area is the monitoring of the SLA which is agreed between the functional organization and the provider of the service register.

Application Management

Application management requirements are formulated to ensure that the service register itself is up and running. Important processes for the service register are incident- and availability management.

Exploitation Management

The exploitation requirements mainly focus on which hardware and software is used, and how the service register is aligned with existing systems. For instance, a requirement may be that an existing Active Directory of Radius server is used for authenticating a user. It may also be necessary to use a certain type of hardware if that ensures a faster problem management.

6 Service register design

This chapter designs a service register based on the requirements described in the previous chapter.
Section 6.1 briefly states when a service register is considered to be effective. Section 6.2 discusses the infrastructural needs for implementing the processes on organizational, information and technical means. Section 6.3 integrates the design in an Archimate model that shows an overall view on the service register. Section 6.4 briefly discusses the relationship between design and requirements.

6.1 Effective service register design

A design for a service register is effective if it satisfies the essential processes as mentioned in Section 3.7. This can be done in various ways and with various focuses. This research focuses on the disclosing of the service register by the end-user. We define the design to be effective when it satisfies the end-user in his information needs.

The end-user satisfaction is measured by assessing five factors, being *Content, Accuracy, Format, Ease of use* and *Timeliness* (Doll & Torkzadeh, 1988). These factors are therefore the main design focus.

6.2 Infrastructure layer

The third design layer deals with how the processes are supported. Three main means are distinguished which realize the processes: information, technical and organizational means. As the means are supposed to support the processes, the means can best be identified by determining the needed means per main process (deliver content, use content facilitation and transaction processing) which match the 3 design aspects of *content, use features*, and *stream*. Without the ambition to be complete, the following sections propose needs for each category of means.

6.2.1 List of Information means

The processes need information to perform tasks for the service register. The information means are the available and accessible data and means to be able to provide the data, i.e. the organization of this data and resources, and the management of the up to dateness (Wijnhoven & Kruijenbrink, 2008). The following table shows information means for the processes as mentioned before.

Table 6 Information means

Requirement Process	Information means
Deliver content	- list of available service specifications - requirements for new services - requirements for standards to use when creating a service specification - check ists for validity checking of service specification - request of consumer - authentication of consumer - list of pending specifications - information about informat on need of consumer - structured data storage anc retrieval - contracts with suppliers - list of suppliers - knowledge about consumer usage - data models for efficient storage and retrieval
Use content facilitation	- data about consumer's need - supportive information database - data models for retrieval. - mapping of client's needs with available data - ontology data and relations - relevance data - modification date of service specification - data about preferred user interfaces and poss bilities - links to supportive information (help functions) - direct links betweer service specifications and supportive data
Transaction processing	- contracts with consumers - contracts with creators - delivery data - receivement data - subscription data

6.2.2 Conceptual data model

This section provides a data model for the service register. The intention has not been to be exhaustive, but rather to provide a general overview of the data which is stored.

The information means in

Table 6 are divided into the various information types as developed in Section 3.2. These are discussed after Figure 19 that shows the data model.

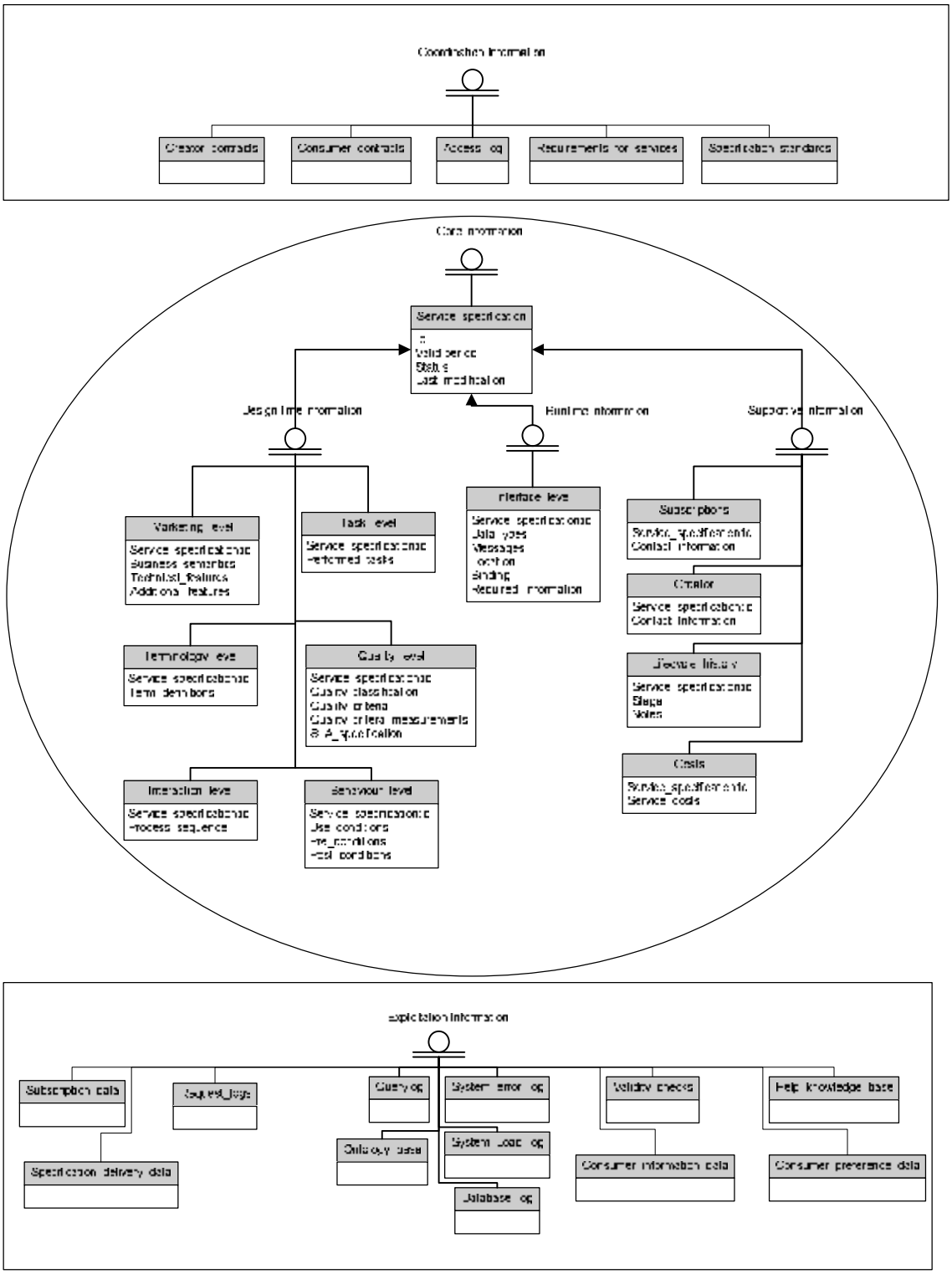


Figure 19 Conceptual data model

The data model is divided into the information types categories, which are discussed below. Not all tables are discussed; the focus is on the tables that are important in the delivering of the specification to the end-user.

Core Information

As previously described, the core information consists of the service specification (split up into design- and runtime information) and supportive information which is directly linked to the service specification.

The tables represent the different layers of Turowski et al (Turowski et al., 2002), and contain specification information of the service. The syntax of the specification elements (i.e. which standards are used for describing the elements) is left for implementation. Directions to do so are suggested by Turowski et al.

The first table identifies a certain service specification with an unique identifier. To support the service lifecycle, a period may be specified in which this service specification is valid. The end date can be optional, and can be used to fase-out a certain version of a service, and therewith a service specification.

Data in the *Interface_level* table precisely describes which data types are used, which messages are communicated by the service (including e.g. error messages), where the service is located, how to bind and which data is needed to invoke the service.

The *Behaviour_level* table specifies rules which should be true before or after the service is invoked. These pre- and post conditions constrain the values of the data submitted to the service or produced by the service.

In the *Interaction_level* table, process information is stored. It might state that the user should make sure that the service assumes that other services are called before or after the invocation of the service.

Whereas the previously described tables mainly contain functional information of the service, the *Quality_level* table describes how the service is adhering to quality factors. These factors are described; measurements are described how the factors are assessed and which values of these measurements the creator strives to offer. If the creator is offering a SLA, these values may be guaranteed in SLA information.

The table *Terminology_level* contains (formal) definitions of the concepts and other important words which are used in the other tables. The definitions can be used to create an ontology which can be used by the end-user to find services.

Both tables *Marketing_level* and *Task_level* are important to create an application domain wherein the service is designed to work for. Experiences may lead to supplementary application domains, which give the user a better idea of how the service can be useful.

Within the category of supportive information are tables which store additional information. The end-users may subscribe to the service specification to receive updates about the specification. The contact information is stored in the *Subscriptions* table and can be used by the creator to

notify the subscribers. In the other way around, subscribers may need the contact information of the creator, for instance to notify in case of failures, or to agree upon a SLA.

Apart from the validity period specified in the service specification, other service-lifecycle information can be stored such as minor changes in the table *Lifecycle_history*.

Whether the user has to pay for the service which is specified in the service register depends on the organization. However, it is most likely that the offering of service leads to "Software as a Service" (SaaS). The service register stores the payment method (subscription/pay-per-use) and the fee, and supports the billing process by providing this information.

Exploitation Information

The Exploitation category contains tables in which information is stored to facilitate other processes, which are not directly linked to a specific service specification. The system supports maintenance processes by storing information about the system itself. These are the *System_error_log* to log warnings and error items which occur, a *System_Load_log* to monitor the performance of the system and the *Database_log* in which database queries which are performed are stored, so that a rollback can be performed when needed.

Tables are introduced which contain information related to search queries. In the *Querylog* the queries of the end-users are stored. In addition, the results of the query can be stored with the data. Analysis of the contents of this table can be useful in two ways. First, the log can be analysed to examine how users are searching for their information. With these results, the system can be adjusted to better fit the information requirements of the end user. Second, when the results are examined with the query, the exactness of the results can be assessed and improved. Furthermore, using the results of the previous search action, new search actions can be improved by reasoning about earlier results.

The *Ontology_base* table contains links between ontological elements, and can be used to assist the human end-user with presenting alternatives. Using an ontology is also a means for determining the relevance of the result, and therewith the ranking of search results.

Coordination Information

Coordination information is primarily meant to give insight in the use of the service register system. This can be achieved by logging the access to the system by type of user in an *Access_log*.

6.2.3 Technological means

Technological means are hardware, data communication networks, software platforms, and tools to support the processes of the service register. The following table shows technological means for the main processes.

Table 7 Technological means

Requirement Process	Technological means
Deliver content	- tool for syntactical checking of service specification - transformation tool for Level of Representation - querying software - interfaces for receiving content - communication channel with creators and consumers - databases to store specifications, requirements, invoices
Use content facilitation	- interfaces - interactive search determination tool - search system - ontology searching mechanism - data communication network - presentation/customization tool - database with feature information - hard/software platform
Transaction processing	- billing system - subscription system - counting system

6.2.4 Organizational means

The organizational means consists of lists of responsibilities, assigning of tasks, coordination principles. Furthermore, the organization may be changed to create a responsible unit which is essentially part of the service register. The following table shows the organizational means.

Table 8 Organizational means

Requirement Process	Organizational means
Deliver content	- Service register system programmers - Service register system maintenance people - ICT services management to support client interactions - Network of suppliers. - End-user support - Coordinator to create/manage specification requirements - Procedures to add service specifications - Archiving procedures. - Librarian - Database administration.
Use content facilitation	- End-user support unit - Librarians who manages supportive information - Librarians analysing system's usage to improve matching - Provide search and results options to clients - Create and maintain ontological links - Client feedback collection procedure
Transaction processing	- Creators ('account management') - Consumers - Sales administration - Contract management

Among others, Koppenjan and Groenewegen (Koppenjan & Groenewegen, 2005) argue that the organizational (re)design is as important as the technical design when implementing complex technologies. When implementing the service register in a real case environment, the model of Koppenjan and Groenewegen identifies levels which are to be considered when (re)designing organizational structures for a service register.

- *Actors and games* – an actor analysis to determine the actors which are involved in the service register (i.e. who are the consumers, creators, system maintenance etc), and how these actors interact and influence each other.
- *Formal and informal organizational arrangements* - Agreements, contracts, mergers within an organization etc as formal arrangements, and rules, codes, norms within an organization as informal arrangements
- *Formal organizational environment* – Formal regulations, rules and laws which constrain the design or specify mandatory organizational means.
- *Informal organizational environment* - Norms, values and codes which are required by the environment

Organizational arrangements are made based upon these analyses, so the technological design is supported by the organization.

6.3 Integrative architectural design

In order to bring the insights of the previous sections together, we are looking for a modelling tool which is able to represent the different design layers. Most modelling tools are specifically designed for business or technical modelling; however a more general architecture tool is needed to model the individual layers, and the interconnections between the layers. Such a tool is Archimate. This section describes the Archimate tool and provides an architecture for a service register.

6.3.1 Archimate tool

Archimate is developed by Lankhorst and his project team (Lankhorst, 2005) and is a language to reflect the already existing modelling languages from the business process and computer sciences fields into a language that both fields can understand. By doing so, Archimate is able to create an overall architecture in which (at a desired level) the domains of a company can be modelled, and relationships between domains can be identified. In this way, the architecture can be analysed, and impact of changes can be determined.

Archimate is oriented on the concept 'service', as this provides support of developments such as web services. A common way of looking at a service oriented model is a layered view, in which each layer provides services for another layer. The Archimate language defines three layers, which all use services of the layers below it. Each layer can be split up in other layers, the external layer where the services are available, and an internal layer with processes that implement those services.

Within these layers, Archimate uses on an abstract level the same concepts and connections as in the other layers, though with a slightly different meaning in each layer. This way, meanings can be given to the concepts without losing the coherence between the main layers. Archimate is specifically useful to model organizations whose business processes rely on Information and (technological) infrastructures

6.3.2 Archimate model

The Archimate model in Figure 20 shows the service register as a high level architecture of an Information Intermediary. As this version is difficult to read, a larger version is present in Appendix D. Appendix C provides information on how the design elements previously discussed in this chapter are translated to concepts which are used in Archimate.

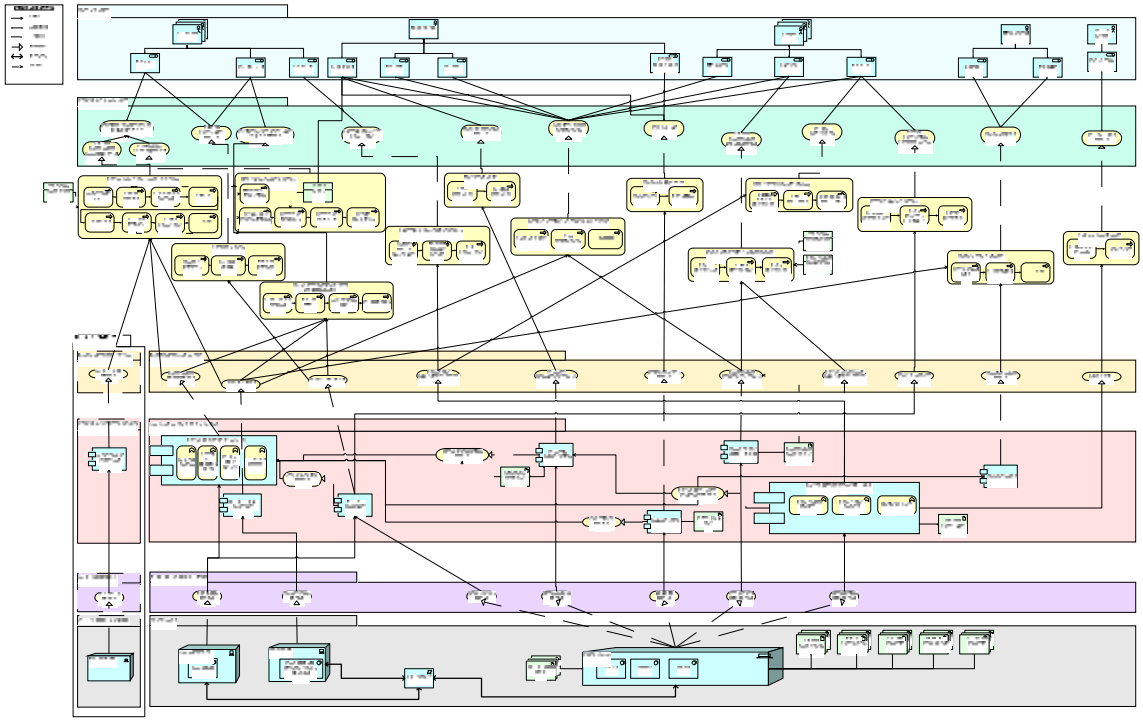


Figure 20 Integrative model of a service register as an Information Intermediary

6.4 Traceability of requirements

The use cases (i.e. the functional requirements) of Section 5.2.5 are reflected in the external business services in the Archimate design. The implementations of these services are supported with processes which in turn rely on lower level services.

As the design is not for a specific organizational setting, the traceability in the sense of showing which software/hardware is responsible for what functions. The same holds for the quality requirements. In a real design, these will be achieved by making design choices that affect the quality design. However, since these quality requirements are not known and the purpose was to create a general design, these can not be traced back to applications

7 Design and construction of end-user interaction (SISIS)

The validation of the service register architectural model in Chapter 6 would require a complete implementation in an organization, for which no resources are available within the scope of this research. Therefore, as mentioned before, we focus on the end-user satisfaction of the service register system. For future reference, we refer to this part of the service register as "SISIS" which stands for "Search In Specification Information System". This chapter further specifies the SISIS and describes the case study to test the prototype which is based upon the specification.

A part of the general service register design in chapter 6 is further designed in this chapter. Section 7.1 defines the functional requirements of this part by presenting the relevant use case for the design. The next Section, 7.2, briefly discusses some quality requirements. The design is presented in Section 7.3 by subsequent discussions of the activity diagram, dataflow diagram, data model and quality design. Section 7.4 briefly discusses the connections between requirements and design. Finally, Section 7.5 shows how the prototype is constructed.

7.1 Functional requirements

We focus on the functional requirements in the further design specification of the end-user. The quality requirements of the system are important for the system as a whole, however are less relevant when a small part of the system is designed for a prototype.

7.1.1 Goal of the SISIS

Before identifying the requirements, it is useful to state the goal of the SISIS. The SISIS has the main goal of providing service specifications to human end-users. To be able to provide the specification to the user, the system first needs to know what the user is looking for. From sessions with the assigner of this research becomes clear that the system should be able to search for specifications in different ways. For example, based upon the name of the service or based upon the function it provides. We identified before that the SISIS should assist the user with the formulation of its search question.

The SISIS searches available content to look for matching specifications and presents the user with the results of the search action. As became clear in previously mentioned sessions, the user wishes to further process the results, for example by exploring the meaning of definitions or retrieve extra information. The specification levels of a service, based upon (Turowski et al., 2002) as discussed in Section 3.3 are therefore presented to the user as additional information.

The interaction with the service register is web-based to ensure that users can connect to it with minimal efforts and requirements for the workplace.

7.1.2 Exclusions

Some choices have to be made when designing the SISIS. Not all aspects of the total service register design can be taken into account for this single component. Main exclusions of the requirements are depicted below.

- *Authorization* – Although the authorization of the user is essential for access to the service register system, we assume that the authorization is done on forehand, by other means.
- *Security* – The security of the system, dealing with firewalls, error logging and reporting is not included in the requirements
- *Accessibility* – The user is expected to have a certain level of knowledge and abilities to deal with the system, and the SISIS does not have to provide extensive accessibility options.
- *Supportive information processing* – The end-user can have information needs for supportive information, however this is not included in the SISIS.

7.1.3 Use case

The interaction by the user with the service register system is represented by the use case in Figure 21.

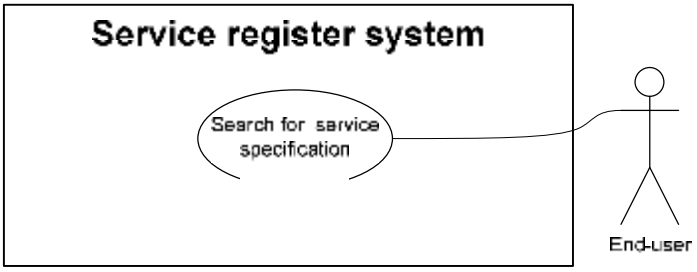


Figure 21 extract of the use case diagram

This use case is further specified textually. The success scenario is the default scenario for a user that does not use functionalities to improve the end-user satisfaction.

Use Case: Search for design service specification

Primary actor: End-user requesting specification information

Stakeholders and interests:

End user: Needs information about a service.

– Manager: Wants the end-user to find the information which is needed, to prevent redundancy and other deficiencies.

Creator: Wants the specification of his service to be found by the end-user

Preconditions: The user is working on a workstation with internet access. The system is online and able to receive the user.

Success end-condition: The user is satisfied in its information needs.

Success scenario:

1. The user opens a connection to the service register via an internet browser.
2. The system presents the user with an opportunity to input a search query.
3. The user enters his query and submits it to the system
4. The system validates the query and searches existing content for matches.
5. The system returns a set of results for the query based on relevance.
6. The user chooses a result.

7. The system shows the summary result, and provides opportunities to further inspect the results.
8. The user stops

Extensions

4a. The system detects an invalid query
   1. The system returns to the previous screen and displays an error message.
   2. The user adjusts the query and submits again.

5a. The system does not find a result
   1. The system returns to the previous screen and displays an error message, stating that no matching specification can be found
   2. The user adjusts the query and submits again or is satisfied and stops

5b. There is only one result
   1. The system immediately shows the summary result (step 7).

6a. The user does not recognize a useful result
   1. The user adjusts the search query
   2. The system proceeds to step 4

7a. The system has alternatives
   1. The system shows alternative results
   2. The user chooses an alternative result and proceeds to step 7

7b. The user chooses a further inspection
   1. The user submits his information need
   2. The system shows additional information

8b. The user enters a new query
   1. The user submits a new search query
   2. The system goes back to step 2

The SISIS should at least support the success scenario of this use case.

7.2 Quality requirements

This section describes some of the quality requirements that apply to the SISIS. Quality requirements for reliability, performance and supportability are not considered here, as they are set by the specific context.

In the design the SISIS, the usability requirements are important for a high user satisfaction. Therefore, the design has to:

1) Be easy to use
2) Provide help functions
3) Provide alternate content (when images are used, and print functions)
4) Have a clear interface

Other quality requirements can be set when implementing the SISIS.

7.3 Design of the SISIS

The SISIS is designed using an activity diagram to model the process, a dataflow diagram to show the use of data and a data model that models the tables which are used in creating the SISIS.

7.3.1 Activity diagram

A first step in the design is the analysis of the process that provides the functionality. This process implements the functionality as required by the use case "Search for design service specification". This process is shown in Figure 22.

The success scenario indicated in the use case is represented by following the straight line downwards, also indicated by the numbering (X.0 series). The success scenario can be extended at various moments in the process.

The process is initiated by the user who is then presented with search functionalities. The arrow on the left represents the user when he decides to stop at this point (e.g. when no results are returned). However, normally the user will enter a query and submits the query. When the query is determined to be invalid, the system returns to the previous state of presenting the search functionalities.

The number of returned results determines the next action of the system. If there are multiple results, the user is presented with a choice in the list of result options. If there is just one result, the user does not have an option and the system presents the summary result. If there are no results, the system returns to the first screen.

The search query can be adjusted when the user is not satisfied with the returned list of results, after which the query is again checked to be valid. However, when a result is picked, the system shows the summary result, further inspection options and alternatives simultaneously. When the user is satisfied, the process stops. However, the user might choose an inspection option, after which the result for that inspection is shown. This can be repeated, whilst the alternatives and inspection options remain displayed.

The user may also choose an alternative result. The system treats this choice as if it is picked in the list of results, and shows the summary result, further inspection options and alternatives simultaneously.

Explained: search for content

The search for content is assisted with ontological support. As is explained in Section 3.3, the definition level of (Turowski et al., 2002) is not seen as a specific specification level, but supports the other layers. The definitions which are specified for the service specification have the following usage.

- By combining the definitions of the service specifications, an ontology of relevant terms can be defined. When a search is entered, the ontology can be used to find a service, which uses a slightly different terminology, however is closely related to the search query of the user.

- In a similar way, the ontology can be used to determine which other service specifications are closely related to the search query. These can be suggested to the user.
- The distance of terms in the ontology may be used to indicate how relevant the specification to which the term belongs is for the user.

The definitions can also be used to define a domain language and glossary. When the user is confronted with such a term during his search, he is able to retrieve a definition of the term in the glossary.

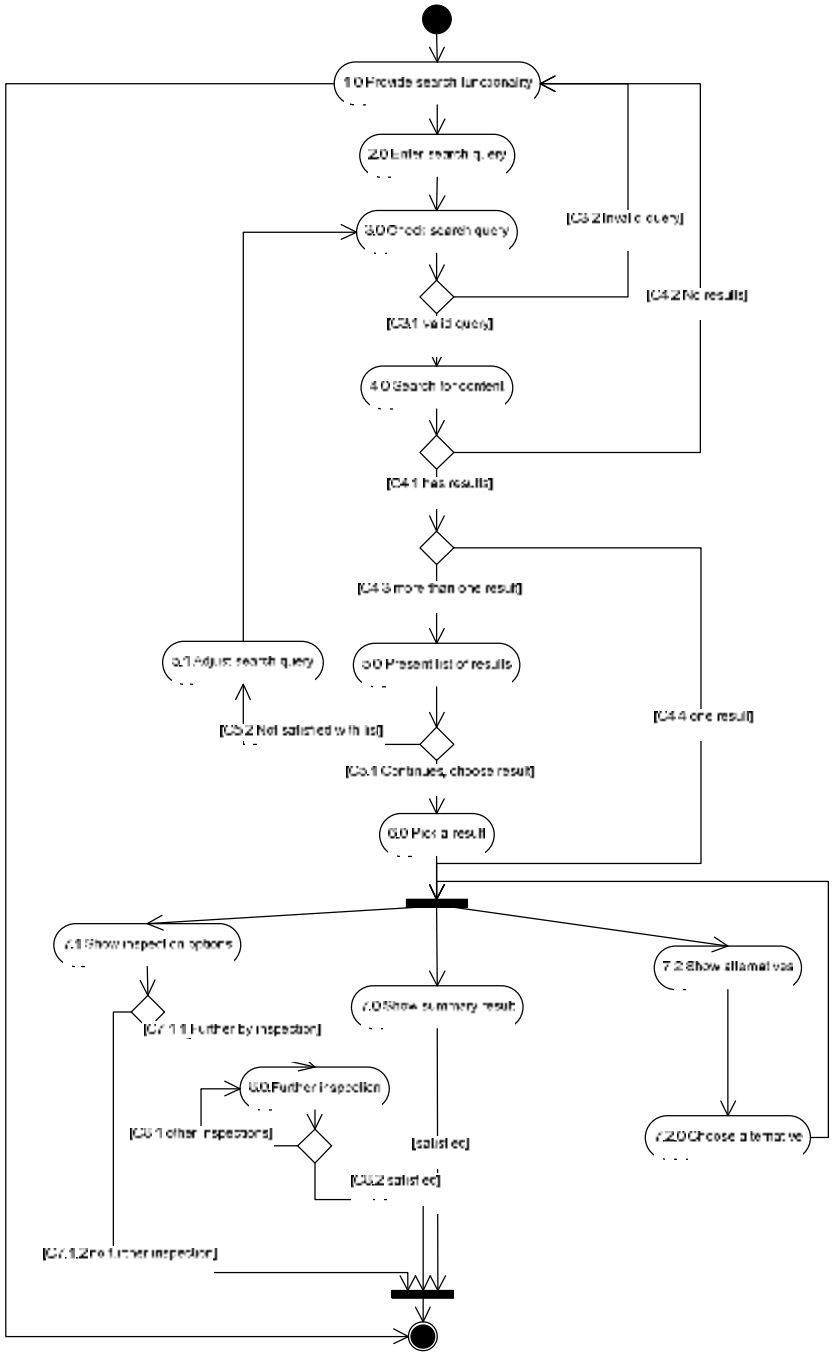


Figure 22 activity diagram 'search design service specification'

7.3.2 Dataflow diagram

Data is used to support the process that is modelled with the activity diagram in Section 7.3.1. A dataflow diagram shows the data stores that are used to produces the results. This is shown in Figure 23. Not all activities have to be supported with data, user's actions or for example the validity check may be done without data stores.

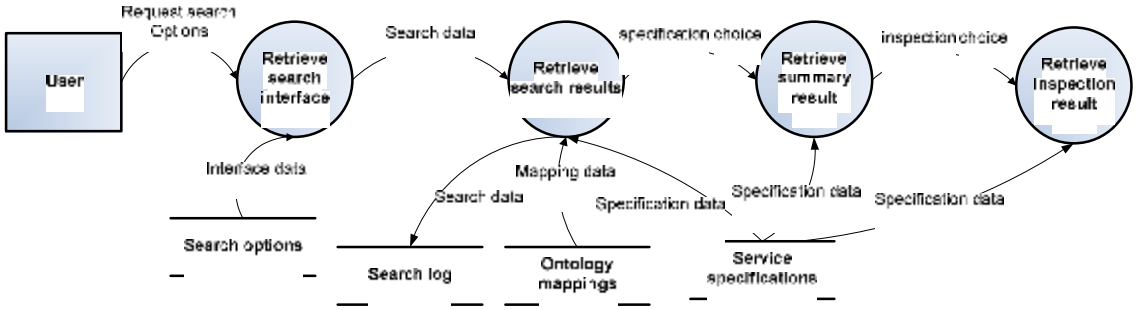


Figure 23 Data Flow Diagram process 'search design service specification'

When a request for a search action is retrieved by the web server, the user interface for the search form is created. What search options are presented to the user is derived from the appropriate data store. By configuring the data store, the service register can adjust the options which are displayed. The user submits the search data, and the process uses different data stores to generate the search results. First the search data is stored for future analysis. Matching service specifications are derived from the service specifications data store, and ontology mapping generates the relevance of the results and alternatives. When a choice is made from the result list, the summary result for that specification is generated from the service specifications. When the user chooses to inspect further (additional information), the relevant specification data is retrieved. Table 9 shows the direct relationships.

Table 9 specification of data usage

Data store	Activity in activity diagram
Search options	1.0 Provide search functionality
Search log	4.0 search for content
Ontology mappings	4.0 search for content 7.2 show alternatives
Service specifications	4.0 search for content 7.0 show summary result 7.1 show inspection options

7.3.3 Class diagram

The classes used in the SISIS are depicted in the class diagram in Figure 24. A few relations are briefly discussed here.

- The *User* receives a number of *search options* from the system. A single search option may be received by various users.

- The *User* generates *Search log items* while he is searching. Each log item belongs to a certain user, and may have a query linked to the log item.
- A *service specification* has multiple 'belongs to' associations with classes that further specify the specification. However, the multiplicity of the 'xxxx level specification' classes may vary from zero to any number of associated entities.
- The *ontology class* is linked to the *ontology role* twice. This is because the *ontology class* can be involved in a *ontology role* (relationship 'has link'), however the *ontology role* may refer to a 'single' entity or a 'complex' class. In the latter case, the relationships of that class are also part of the link.

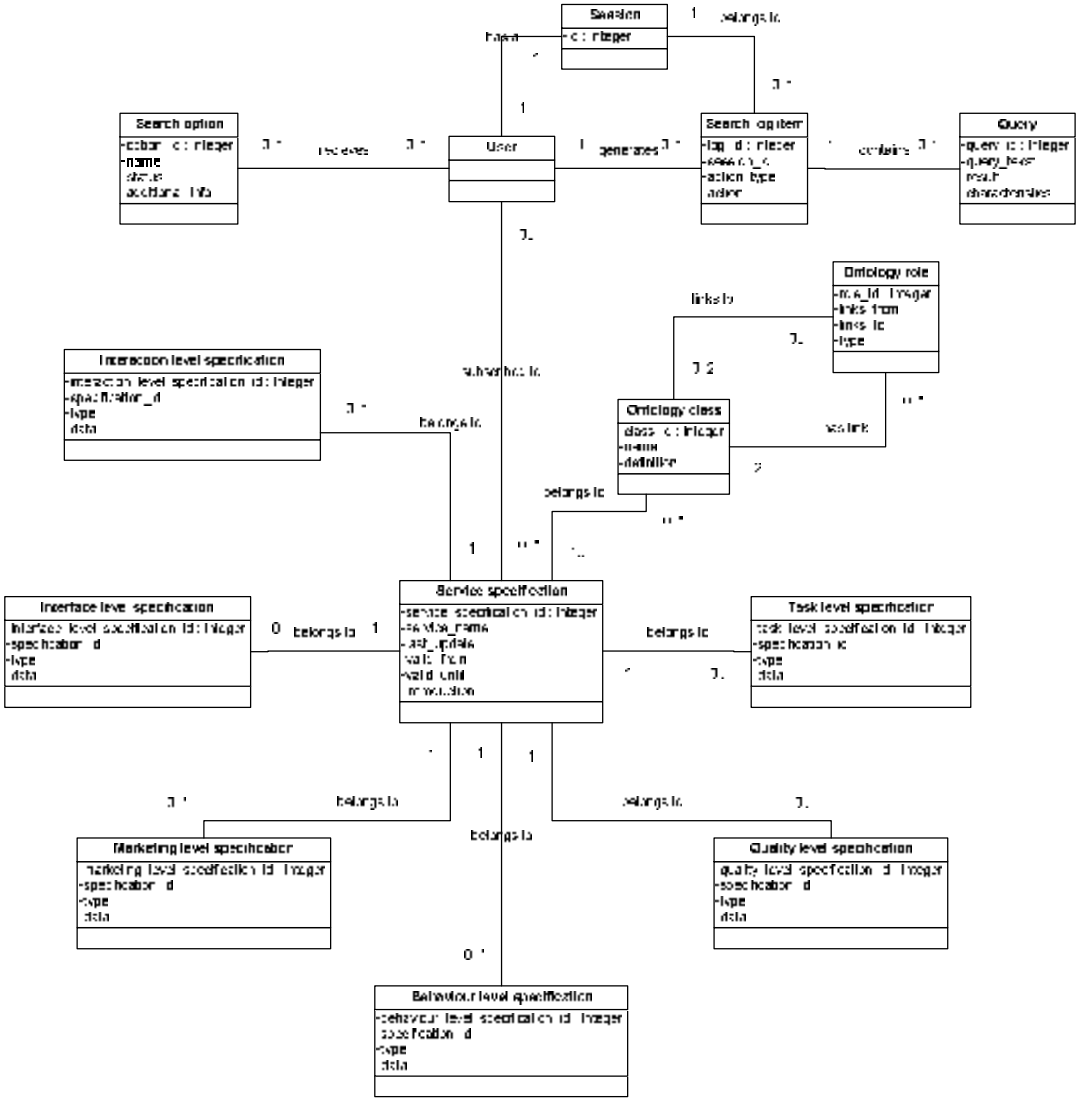


Figure 24 class diagram SISIS

7.3.4 Data model

The storage of information is important for the data flows in the processes. A division of the information in various data stores is already made in the Data Flow Diagram in Figure 23. Section 7.3.3 showed the relations of the classes. This section further specifies the data stores and classes into instance tables.

7.3.4.1 Search options

The data store *Search options* contains a table which can be used to configure and create a search interface for the user, and can be used to add use features. By adding or removing items, search options can be changed. The web server displays the search options which are enabled. The table which is needed has only a few attributes.

Table 10 Data table search options

Attribute	Type	Description
Option_id	Integer	Identifies the search option
Name	Character	Name of the search option
Status	Integer	Status of this option, whether it is enabled.
Additional_info	Character	Optional extra configuration information, needed to render the option

7.3.4.2 Search log

The Search log data store is used to store information about search queries of the user and can be used in the exploitation phase to analyse the search behaviour of the end-user. The coordinator or other assigned roles within the service register uses this information to improve results of the queries. The data store has at least a table in which the initial query is stored

Table 11 Data table query

Attribute	Type	Description
Query_id	Integer	Identifies the query
Query_tekst	Text	Contains the query itself
Result	Text	Contains information about the results after execution of the query
Characteristics	Text	Optional extra information, such as which features are used in constructing the query.

Each query is essentially part of a search action, and is therefore linked to a search log entry. A user generates multiple search log entries in a single session, some of which contain a query, however all actions of the user can be logged to create reports which show how the user is using the system

Table 12 Data table search log

Attribute	Type	Description
Log_id	Integer	Identifies the log item
Session_id	Character	Session identification number to link different log items
Action_type	Integer	Identification of the type of action the user performed
Action	Text	Data related to the action, e.g. the query

7.3.4.3 Ontology mappings

An important feature to improve the use satisfaction of the system is to add use features which improve the experience. The provision of information about related service specifications and the relevance of the search results relate to this function. The definitions which are used in the ontology are stored in the *service specifications* data store, however the mappings of the items which form the ontology are stored in this datastore. The service register can add mappings to create a common domain ontology. The creation of an ontology can not fully be automated.

As identified in Section 3.9.2, the Rule-Based Logic is the best way to represent knowledge. However, it is not implemented widely, and we therefore base the data model upon the Description Logics. With Description Logics, roles can be described between different classes and their instantiations. Reasoning about description logics is reasonably efficient.

The classes of the ontology can be defined and stored in a table. Instantiations of the classes can be found in the *service specifications* data stored and linked to the classes with the use of a separate table, containing the roles between the classes and instantiations.

Table 13 Data table Ontology classes

Attribute	Type	Description
Class_ic	integer	Identifies the class
Name	Character	Textual identifier of the class
Definition	Text	Short definition of the name

Classes and their instantiations are linked to each other with roles. These roles are stored in a separate data table.

Table 14 Data table Ontology roles

Attribute	Type	Description
role_id	Integer	Identifies the role
Links_from	integer	Identifies the class or instantiation involved in this role
Links_to	integer	Identifies the class or instantiation involved in this role
Type	Text	Type of the role

The *type* attribute defines the relationship of the two classes and/or instantiations which are involved in the link.

7.3.4.4 Service specifications

The data store *Service Specifications* is of high importance in the service register as it contains the service specifications (i.e. the *content* aspect of the design). Although we have identified that a service register must be able to store supportive information (Sections 3.3, 3.4 and 3.5) this is neglected for the SISIS as the end-user does not have this information need when searching for specifications.

As is shown in Section 5.2.3 and the conceptual data model in 0, the *Service Specifications* stores specification information at different levels. It is important to store the specifications using standards to ensure that all aspects are described and that it is described clearly. However, this requires defining standards that are used in the service specification. Defining these standards is out of the scope of this research, and we therefore model this information as if it is a Binary Large OBject (BLOB), which can contain various information types.

The data store consists of various tables, starting with a table that identifies a specific specification.

Table 15 Data table service specification

Attribute	Type	Description
specification_id	Integer	Identifies the specification
service_name	Character	Textual name identifier for the service that is specified
Last_update	Date	Contains the date of last modification
Valid_from	Date	Date from which the service can be used
Valid_until	Date	Date until which the service can be used

This general information describes specifications which are relevant for all linked information tables, specified below.

Each of the specification levels of a service (see levels of (Turowski et al., 2002), Section 3.3) has a table. The reason for this is that each service specification may have more than one entry in the specification layer tables below. For instance, the quality level may require an entry describing the measurements and values, and also other SLA information.

As the table structure will not vary much over the different specification levels for the SISTS (however when implemented with appropriate standards it differs greatly), the table structure is given once.

Table 16 Data tables specification levels

Attribute	Type	Description
id	Integer	Identifies the specification level entry
Specification_id	integer	Identification of the specification the entry belongs to
Value	BLOB	Content of the entry
Type	Varchar	Depicts the (MIME)type of the Value attribute

7.4 Traceability of requirements

As the SISTS consists of a single process, the mapping of requirements and the design is direct. The process steps in the activity diagram implement the various steps specified in the use case that determines the functional requirement. Table 17 shows the links between use case actions by the system (7.1.3) and the activities of the activity diagram (Figure 22).

Table 17 Traceability of requirements

Requirement	Activity diagram number
The system presents the user with an opportunity to input a search query	1.0
The system validates the query and searches existing content for matches	3.0 4.0
The system detects an invalid query -> The system returns to the previous screen and displays an error message	C3.2
The system returns a set of results for the query based on relevance	5.0
The system does not find a result -> The system returns to the previous screen and displays an error message, stating that no matching specification can be found	C4.2
There is only one result -> The system immediately shows the summary result	C4.4
The system shows the summary result, and provides opportunities to further inspect the results	7.0 7.1
The system shows alternative results	7.2
The system shows additional information	8.0
The user submits a new search query	5.1

Although little quality requirements are formulated, these are to be implemented in the design. This is done by following guidelines for an accessible web page (Overheid, 2007), as defined in the design.

7.5 Development SISIS prototype

The SISIS design of chapter 7 is developed into a prototype to get a view on the proposed system. This chapter describes the prototype development and the development of the LDDI implementation which is used to compare the user satisfaction with.

An Information Intermediary is generally meant to be accessible for a large public, and is therefore often developed using web-based techniques and tools. The service register prototype of the SISIS is developed using the following software:

- Apache http 2.0.63 webserver on a CentOS 4.1 operating system
- PHP 5.2.4 scripting language for dynamic scripting
- MySQL 4.1 database server for data storage
- Zend Framework 1.5 to create web applications

7.5.1 Functionality

The prototype implements the process as explained in the activity diagram in Figure 22. The activities performed by the system are explained in this section. As it is a prototype (a simplified example of the proposed system), not all functions from the design are implemented in real, but are simulated. For example, the use of ontology is not implemented.

Provide search functionality

The prototype provides the user with an initial screen as showed in Appendix E. The user can provide a keyword-based search screen. Furthermore, the user may indicate which level of information (i.e. specification level of (Turowski et al., 2002), except for the terminology level, see 3.3) the search should focus on, which is optional. Help is provided by clickable question marks, which –when followed- provide additional information about the various information levels.

Check search query

After submitting, the prototype filters the query and validates the query. The validation which is implemented consists of a check whether the submitted query is non-empty and consists of at least 3 characters to search for. If one of these checks fail, the prototype returns to the initial search screen and provides the user with an error message. This is shown in Appendix E. Additional, the keywords that are used are checked if they are 'commonly used'. This filters out keywords such as 'if', 'the', 'an' etc.

Search for content

If the submitted query is valid, the prototype searches for service specifications which match at least one of the provided keywords. This is not done via an ontology, but keyword based with added wildcards to keep the prototype simple. If a search query consists of multiple keywords, the returned results should contain at least one of the keywords provided. It assigns a score to the service specification, thereby taking into account the focus of the user if provided in the initial screen by doubling the score of that level. As seen in the activity diagram, there are three possible outcomes which are implemented.

1. If there is no result, the prototype returns to the initial screen with an explaining message for the user
2. If there is just one service specification which matches the submitted query, the prototype directly displays this specification, with an explaining message to the user.
3. If there are more than one results, the prototype continues to the next activity

Present list of results

The prototype now shows the service specifications which match the submitted query. For each specification, the prototype calculates a 'relevance' based upon the score of the specification, as determined while the query was processed.

The specifications are displayed in descending order of relevance. For each specification, the 'name', 'validity' and 'last update' fields are displayed, along with an overview of sentences in the specification which match the keywords. Where possible, each of the information levels is represented here

Additionally, the last submitted query is shown on the left side, so that the user is able to adjust the query and submit it again.

The user can either click one of the sentences to go to that specific information level, or choose to view the 'summary result' by clicking the name or follow the link.

Summary result

When the user choose the summary result, the same information about the service is provided as in the search list, complemented with a short description of the specification. The 'summary result' is one of the three views that are showed simultaneously, with 'show inspection options' and 'show alternatives'. The summary result displays a short description of the tasks of the service. The inspection options can be used to retrieve in-depth information.

Show inspection options

The inspection options which are implemented consist of a choice for one of the specification levels of the specification. These information levels represent the levels that are identified for service description by (Turowski et al., 2002). The user can use these levels to make a choice for the type of information that he needs.

The division of the entire specification into various levels is a way to give the user the option to choose his abstraction level and level of representation (see Section 4.4.1). Each of the specification level represents a different abstraction of the service that is specified. The marketing level does provide the user with general information. Each subsequent level has more detail, where the lowest level, interface level, provide the user with specific information such as wsdl information.

Help function about the meaning of each level is provided. The user can retrieve more information about what is stored in a certain level by clicking the question mark on the left of the screen, in the search query section.

Further inspection

If a level is chosen, the full result for this level is shown, and the matching keywords are shown in red and bold font. The other inspection options are now shown on the left side.

Appendix E shows an example of a further inspection of the 'interaction information level'. The full result of a specification level can consist of multiple parts, which are all displayed here, and can be of any data type. This makes it possible to display images and create download links, for example for a wsdl specification. The Appendix shows some text and two tables that are part of the information of this specific information level. Displayed in the block on the upper left side are links to the other inspection options.

Show alternatives

The prototype generates alternatives based upon the search result and displays these during summary result and inspection on the left side. When the user follows an alternative, the summary result for this alternative is shown. This is also shown in Appendix E.

7.5.2 Specification data

The specifications which are stored in the prototype are transformed service specifications of a large insurance company in the Netherlands. The prototype replaces the name of this company with the fictive name "Sure4U", however the specified services are actual descriptions of services used in the company

7.5.3 Access

The prototype is deployed on a public web-server. At the time of writing, the prototype is available via the access URL http://demo.service-register.nl however will not be maintained and will not remain available.

8 Exploitation and evaluation

The implementation of any software system in an organization does not end with the construction of the software and hardware. The construction is followed by a continuous phase in which the system is constantly evaluated and adjusted to optimize performance. This phase is the final layer of the design science and is called 'Exploitation and evaluation'. The exploitation of the service register requires attention to keep delivering high quality service specifications and have satisfied consumers.

Before various exploitation issues are discussed in Section 8.5, we describe the case study that is performed to measure the satisfaction of the end-users of the prototype as developed in Chapter 7. Letting users interact with the prototype is a way of 'exploitation' of the service register. The case further functions as a validation for the design methodology.

Following (Dubé & Paré, 2003), we describe the case research in the areas of research design, data collection and data analysis.

Section 8.1 discusses the goal of the case research with a prototype of the design in Chapter 7. Section 8.2 describes how the data for the case is collected; Section 8.3 explains the procedure that is followed by the participants. Section 8.4 discusses the results of the case research; Section 8.5 discusses further exploitation issues that are important for an Information Intermediary.

8.1 Case research design

We develop the prototype to test how the end-user satisfaction of our approach is valued by the participants

The main research question is:

Does the prototype of a service register, designed as an Information Intermediary, provide a higher user satisfaction than the average that can be expected?

We expect the prototype based upon the design as an Information Intermediary to have a higher user satisfaction than average. The average that is meant here is the score of 3 on a five point scale, which is used to measure the user satisfaction. We expect a higher user satisfaction as some use features are implemented in the prototype. A higher user satisfaction may lead to a better use of the service register system (Ives et al., 1983), and is a factor for the success of the service register.

The end user satisfaction measurement itself is best described by (Doll & Torkzadeh, 1988) who define 12 indicators to measure end-user satisfaction, which are still widely used. The indicators are grouped to measure *Content*, *Accuracy*, *Format*, *Ease of Use* and *Timeliness*.

The case study uses service descriptions of a large insurance company in the Netherlands. This company primarily sells insurances via various intermediaries. For various reasons is decided to create a SOA architecture to provide services that are accessed by the intermediaries. The descriptions of these services are transformed and stored in the prototype.

8.2 Data collection

The prototype is accessible via the internet. A group of professionals is asked to perform certain tasks with the implementation.

After performing the tasks, the participant is presented with a short questionnaire to measure the end-user satisfaction, using the questions in the model of (Doll & Torkzadeh, 1988) and a 5-point rating scale. These results are stored for analysis.

The procedure of the case is further explained in Section 8.3, the data analysis is found in Section 8.4.

8.3 Case Procedure

The case procedure is described by the tasks that are performed and the measurement of the end-user satisfaction hereafter. A group of 89 professionals is asked to go to the main web page of the research, http://www.service-register.nl. The participants read a short introduction to the assignment before proceeding to the prototype.

8.3.1 Tasks

Background

The SISIS is used for searching service information by employees in the organization. This situation is simulated in the service register implementations described above. Therefore the participants should perform tasks with the implementation that reflect information needs as encountered in an organization.

The content in the service specification consists of service specifications that are used in a large insurance company. The company currently has no service register, however uses separate documents to keep track of the services. The usefulness of a service register is understood, however they have not been able to find a service register that meets their (information)needs. The prototype uses the data in the documents to create a service register.

As the terminology is company specific, we give a brief introduction to the context here. The services that are specified provide functions that can be used by intermediaries that sell insurance products to customers for the insurance company. A customer is called a 'party', data about a party is stored as 'party characteristics'. Data about a sold product is stored in a 'Policy' that can be altered with an 'Endorsement' service. Communication between different domains passes through an 'external service' to validate the request.

Used in case

The participant is instructed to read an introduction to set the context of the service register. He is then asked to perform two simple tasks with the service register implementation he is about to see. These tasks are selected on the following considerations:

1. The task reflects a typical information need of an end user

2. The task results in at least one possible service specification[2].

Based on these considerations the following tasks are identified, from which two tasks are arbitrarily selected and presented to the participant. The tasks contain specific terminology that is used in the company.

- *Is there a service to 'change a party characteristic' ?*
- *What code is returned when the PartyChangeV1 service completes successful?*
- *What is sent as input to the validation method of the service 'ExternalService'*
- *Is "Cancellation of the entire Policy" one of the supported types of the Endorsement service?*
- *Can the service "Contract.GKet" be used to retrieve Cover details?*

Directly below the assigned tasks is a link to the assigned service register implementation. The participant is instructed to follow this link (which opens in a new window) and afterwards return to this screen to fill out the questionnaire.

8.3.2 End-user satisfaction measurement

After the participant has completed the tasks the participant is asked to fill out a short questionnaire. The first part of the questionnaire contains the questions of the instrument of (Doll & Torkzadeh, 1988) to measure the end-user satisfaction. These questions are:

- Does the system provide the precise information you need?
- Does the information content meet your needs?
- Does the system provide reports that seem to be just about exactly what you need?
- Does the system provide sufficient information?
- Is the system accurate?
- Are you satisfied with the accuracy of the system?
- Do you think the output is presented in a useful format?
- Is the information clear?
- Is the system user friendly?
- Is the system easy to use?
- Do you get the information you need in time?
- Does the system provide up-to-date information?

These questions are answered on a five point scale, where 1 = totally not and 5 = absolutely yes. Additionally, the participant is asked for his experience with SOA in general and the service

[2] Although a negative response on a search question can also be considered as a result (that service does not exist, so we need to build it), this type of questions is not included as it may confuse users.

register specific to get a background of the participant. Furthermore, the participant is asked to rate his overall satisfaction with the application on a five point scale where 1 = inconsistent; 2 = poor; 3 = fair; 4 = good; 5 =excellent.

The participant is able to provide feedback about anything with an optional 'remarks' text field.

8.4 Case results

8.4.1 Demographic analysis

From the 89 invitations sent to the target group, 21 persons have participated in the research. From the participants, 81% indicated that they have worked with a service register before.

The fast majority rated their familiarity with the concept of SOA high (mean of 3,95 on a 5-point scale), their familiarity with the concept of service register a little lower (mean of 3,29 on a 5-point scale).

8.4.2 General discussion

The number of participants is too low to draw significant conclusions. Therefore, the discussion is limited to terminology like 'suggests' and 'indicates'.

Table 18 shows the mean of the accumulated scores of the participants and the mean of the overall score indicated by the user. The "Score calculated over scores" is the average score of all 12 questions by the 21 participants (so the average of 252 scores). The "Overall score by participants" is the average of the score that the participant was asked to indicate about his 'overall satisfaction'.

The overall score as provided by the participants for the prototype is close to the calculated score. This indicates that the questions provide a good indication of the user satisfaction. It scores above the 3-point average on a 5-point scale which indicates that the participants are more than average satisfied.

Table 18 main scores

	Score calculated over scores	Overall score by participants
Prototype	3,51	3,38

8.4.3 Scores per aspect

The instrument of (Doll & Torkzadeh, 1988) measures the user satisfaction in four different areas, being *content*, *accuracy*, *format*, *ease of use*, and *timeliness*. The scores for the questions that indicate a specific area are accumulated and the mean is calculated, and the standard deviation is given. The following is extracted from these numbers.

- The relatively low standard deviation for the Prototype suggests that the participants mostly rated the *content* and *accuracy* above the 3 points
- The Prototype scores slightly above average for the *format* and *ease of use* aspects.

Table 19 scores for aspects

		content	accuracy	format	ease of use	timeliness
Prototype	mean scores	3,52	3,71	3,29	3,36	3,64
	standard deviation	0,92	0,50	0,96	0,84	0,81

8.4.4 Scores per question

The main datasheet containing all collected data is found in Appendix F. Each column indicates a question, each row is a participant. The 'mean' column is the average score of the participant on the questions that are related to the aspects which are measured.

8.4.5 Feedback

The participants had the opportunity to send feedback about their experience. A few participants thought the Prototype was giving "too much information" about an inspection level, others recognized the usefulness of the division into various information levels "[The Prototype] provides service information from various perspectives, which is a positive point". Some participants wondered how the Prototype would perform when the search questions are fuzzier.

Although the full specification information was split into the various specification layers, participants still experienced that the information provided in the full result of a level was an overload of information. This calls for additional reduction of the level of representation of the *content*, to provide the end-user with as few as possible amount of information.

The case results indicate that the answer to the research question for the research case is positively, in the sense that the prototype results in a higher user satisfaction than average. However the small research population and the results on the *format* and *ease of use* only permit a reserved statement here.

8.5 Other exploitation issues

The discussion of the maintenance view of an Information Intermediary in Section 3.6 already showed that the maintenance of the *content* in the service register is important. In fact, the maintenance of the content is the most important part of the exploitation of the service register.

Content

An Information Intermediary strives to deliver a valuable good to its consumers. To ensure that the consumers will use the service register and are satisfied with the information extracted from the service register, the main issue for the service register is to understand the consumers. Only by understanding the needs of the consumers can the service register deliver value for its consumers.

Understanding the needs of consumers can partly be derived by analysing logs which are kept by the service register. A first step is to determine which data is logged to analyse the use by the consumer. Secondly, analysis tools are used to see patterns in the use of the service register, or analyse the generated results with the initial query.

Additional to the log files, the service register should keep contact with its consumers to keep measuring the performance on user satisfaction. Questionnaires can be used to determine the needs and the satisfaction of the end user. Additionally the end-users can be interviewed

The needs of the consumers can change over time, forcing the service register to anticipate on these changing needs and keep developing the software system and procedures.

Use features

The use features that are used when delivering the content to the end-user also require constant attention during exploitation. For example, the provision of relevance information and alternatives for the search query requires the domain ontology to be kept updated, which is also needed for accurate search processing.

Which use features are needed and how they are brought to the end-user again requires an awareness of the needs and wishes of the end user.

Revenue

Revenue streams have a minor role in the service register. Depending of the choices that are made to arrange the revenue streams will these require little attention during exploitation of the service register. Access to the service register is authorized and billed or checked with subscriptions.

9 Final remarks

We developed a service register using an Information Intermediary design methodology in the previous chapters. This chapter provides some final remarks. The design guidelines of (Hevner et al., 2004) are used for evaluation and discussion purposes in Section 9.1. We explain for each of the guidelines of Hevner et al. (2004) how this was addressed in the designed service register. Finally, Section 9.2 presents our main conclusions.

9.1 Evaluation and Discussion

Design as an artifact

IT artifacts are instantiations of information systems and/or constructs, models and methods that are applied in the development of information systems, in this thesis called the service register.

We developed a design of a service register from an Information Intermediary viewpoint. The resulting design can be used as a reference model to check if the information needs of the various user views are met. As various design choices depend on the specific requirements of each specific situation, this model does not provide a final solution but rather provides a path to a design of a service register. Furthermore, we developed a concrete instantiation of the design view as a prototype, that shows how different levels of specification can be used to satisfy the needs of a stakeholder.

We used the design theory of (Wijnhoven & Kraaijenbrink, 2008) as a general design theory for designing Information Intermediaries. In this general design methodology for Information Intermediaries, we have to make some remarks for the appliance of the method to service registers. Firstly, the revenue aspect of the design theory might not be very relevant in service register design. The consideration of revenue for the service register is recommended for an efficient register; however, it does not require or use complex revenue collection mechanisms. Often a subsidizer will just subsidize the service register in an organization instead of a more complex per-use subsidizing. In a more public context a simple subscription method probably fulfils the need. Secondly, the design theory has the option to focus on a design aspect and/or layer. We think that the content aspect of the design methodology has this focus in service register design and that the layers should be followed subsequently as they are proposed in the design theory.

Problem relevance

The objective of design-science research is to develop technology-based solutions to important and relevant business problems.

Over the years, the SOA way of thinking has become mature. One of the important elements for a well-functioning SOA is the service register. It centrally stores information about the various services and their mutual interactions and the end-users of the services. This information and more semantic information are widely discussed by various authors, and standards are developed to support this. These are relevant problems that certainly should be addressed, however we find the problem addressed in this thesis to be more important for the near future.

There appears to be a need for a service register that satisfies the information needs of both business users and technical users. The current service registers are primarily focused on the integration with system architecture, and thus on technical information. However, the need for other service specification information is encountered by organizations. For example, the users expect that the service register provides answers to questions about the purpose or quality of a service. Therefore we noticed a gap between information needs of users and the information that is currently provided by service register. We addressed this problem that is encountered in organizations by distinguishing various information levels and types of information to be specified for a service.

Design evaluation

The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.

We were able to develop and test an important part of the design. However, the full design is not evaluated yet, as this would require a complete implementation project. Based on recommendations of (Dubé & Paré, 2003) we set up a case study. The well established measurement instrument of (Doll & Torkzadeh, 1988) were adopted to measure the user satisfaction. Results show the prototype of our design approach of a service register resulted in an end-user satisfaction that is higher than the 3-point average on a 5-point scale.

Research contribution

Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.

Current research mainly focuses on the automatic discovery of services by applications through semantic searches in the service register. Emphasis is on the semantic web, which promises automatic discovery of services by advanced semantic querying on the service register. Various authors propose extensions to the current UDDI standard to make this possible. However the information needs of human users seem to play a minor role in research.

We designed a service register that better addresses the information needs of the human users compared to other approaches. We hereby focus on the informational purpose of the service register for human users. We think that the design aspects (*content*, *service features* and *presentation*) should be addressed for a design of a service register to satisfy information needs of all involved stakeholders. The structured design approach requires the designer to evaluate different design aspects and make choices whether to include these in the design which results in a comprehensive design. Therefore we propose the use of the design theory for Information Intermediaries to create a design for the service register.

Research rigor

Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.

We have applied several existing theories in the construction and evaluation of the various artifacts. The design science of (Wijnhoven, 2008) is used as construction methodology. As a

guideline for what content is used, we used the service specification layers of (Iurowski et al., 2002). This resulted in a design artifact that was further developed into a prototype. To evaluate the prototype we followed case study guidelines from (Dubé & Paré, 2003) and evaluated using the measurement instrument of (Doll & Torkzadeh, 1988)

Design as a search process

The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.

Considering all these aspects, such as means, ends and laws, at the same time in a problem environment is often not possible. Hence, different researchers will come up with different solutions for the same problem. Therefore, multiple service register design approaches exist that all have stronger and weaker points.

We presented a design that can be considered as an attempt towards more functional service registers instead of research products. Whether the resulting service register indeed satisfies the needs can be concluded only after full implementation. Unlucky, we did not have sufficient time to actually fully implement our new design in practice. Hence, this thesis can be considered as a start point. To perform a sound comparison between different types of service registers we propose to perform research to following aspects:

Validation of design approach

The design methodology proposed in this thesis has not completely been validated. The methodology results in a design that takes multiple views of different actors into account and is complex to implement. Validation is only possible by implementation of a complete design, which we recommend. Based on such an implementation we are able to determine whether the design methodology meets the expectations, information needs and satisfaction of all actors.

Inclusion of specification standards

The separation of the service specification into various specification layers is a first step towards a structured way of storing the service specifications. It is important to specify exactly what specification information is to be stored. Using standards for storage of information results in less information overload and clear communication to user. Furthermore, the service register can use the structure of the standards to determine which information to present to the user (altering the level of representation). Finally, especially for the 'lower levels' of a specification, the use of standards is important to exchange process information.

Checklist for comparison

It is of interest to derive a checklist from which the result of the methodology indicates focus areas and/or functions. Such a checklist can be used to compare different current, commercial service registers to determine which register is suitable for use in which specific organization. Furthermore, this checklist can be used to check whether an organization has paid attention to all aspects of a service register.

Communication of research

Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

We believe that the research presented in this thesis is accessible to both audiences. The use of technological or managerial jargon is kept to a minimum or otherwise explained in this thesis.

The intention is to provide technical audiences with information on how to design the service register and provide the management audience with information of why this design approach is proposed, i.e., to solve a possible problem in their organization.

9.2 Conclusion

To conclude on the thesis, we recall the main research question as it was posed in Section 2.2.:

Are Information Intermediary design principles a possible structured method for service register design?

The answer to this question is that the Information Intermediary approach is a suitable way to service register design. We will further elucidate this below.

The design of a service register is a complex undertaking due to multiple choices that have to made about the functions that it should and that it should not perform. Properly scoping the service register identifies the information needs, leading to the requirements of the service register. The development of a service register that fulfils the information needs of various types of users is a prerequisites and can be done by using the information intermediary design principles.

We showed that the application of Information Intermediary design theory to the development of a design for a service register is viable. The design theory forces the designer to approach the service register from different views which are involved in the functionality of service registers. Essentially these are the views of *creators*, *management*, *maintenance personnel* and *consumers*. The design method identifies layers (business, process, infrastructure) and aspects (content, use features and revenues) that can be filled in for a service register. These layers and aspects should all be addressed to create a service register that fulfils the requirements set by the users.

An important issue and main focus point in the design of any service register is the quality of the service specifications (i.e., the *content*) stored in the register. Previous attempts to create a public service register show that when the information cannot be trusted to be of high quality, the register is not (or only marginally) used. Therefore, the organization that considers the use of a service register should put mechanisms in place to ensure the quality of the information and make the information available to end-users (i.e., *use features*). The service register is thus more than just a software system, and clearly requires organizational structures. A separate organizational unit (or at least, some roles assigned to employees) can be created to run the service register. The consumers of the service register will be willing to pay for information with high quality.

Our design methodology provides a structured design approach for the service register. A levelled specification of a service is proposed, both for a structured storage of service information and as a means of adding use features. These specification levels cover information needs from consumers of the service register, and end-users can choose their desired level of

specification. Both technical and business oriented users should be able to find their desired information. However what information is to be stored and –at least as important- what standards are to be used when specifying the service are subjects for further research.

The specification of a service in various specification levels may also result in a feature for the users of the service register. The user will then be able to find his desired type of information by selecting the appropriate level of specification information. Other use features are identified such as timeliness, browse options and alternatives.

Definitions of concepts from the service specifications may create domain ontology. This ontology in turn can be used to search for specifications, to determine the relevance of results and to suggest other specifications that might use slightly different names for the concepts that are closely related in the ontology.

Our prototype in the case study showed a user satisfaction consistently higher than the 3-point average on a 5-point scale. However, due to the small research population size these results should be interpreted with care. Therefore we recommend future research to empirically determine the appropriateness of a information intermediary approach in satisfying information needs of the end-users.

To conclude, the application of an Information Intermediary design methodology to the design a service register is viable and makes it possible to apply a structured design methodology. The part that has been tested (the end-user interaction) indicates that the design satisfies the end-user requirements. Whether the design result is a suitable service register design should be empirically demonstrated and is left for further research.

References

Aalst, W. M. P. v. d., & Kumar, A. (2003). Xml-based schema definition for support of interorganizational workflow. *Information Systems Research, 14*(1), 23–46.

Afshar, M. (2007, may 2007). Soa governance Framework and best practices. *Oracle Whitepapers*.

Alevizou, V., & Plexousakis, D. (2006). *Enhanced specification for web service composition.* Paper presented at the European Conference on Web Services (ECOWS'06).

An, L., Jeng, J.-J. J., & Gerede, C. E. (2006). On exploiting system dynamics modeling to identify service requirements, *IEEE International Conference on Services Computing (SCC'06)*.

Cabral, L., Domingue, J., Motta, E., Payne, T., & Hakimpour, F. (2004). *Approaches to semantic web services: An overview and comparisons.* Paper presented at the ESWS 2004: European semantic web symposium No1, Heraklion.

Chen, L.-d., Soliman, K. S., Mao, E., & Frolick, M. N. (2000). Measuring user satisfaction with data warehouses: An exploratory study. *Information & Management, 37*, 103-110.

Colasuonno, F., Coppi, S., Ragone, A., Scorcia, L. L., Noia, T. D., & Sciascio, E. D. (2006). *Jaldi+: A semantic web service registry enabling semantic discovery and composition.* Paper presented at the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06).

Doll, W. J., & Torkzadeh, G. (1988). The measurement of end-user computing satisfaction. *MIS Quarterly, 12*(2), 259-274.

Dubé, L., & Paré, G. (2003). Rigor in information systems positivist case research: Current practices, trends, and recommendations. *MIS Quarterly, 27*(4), 597-635.

Earl, M. (2001). Knowledge management strategies: Toward a taxonomy. *Journal of Management Information Systems, 18*(1), 215–233.

Fettke, P., & Loos, P. (2003). Specification of business components. *Lecture Notes in Computer Science, 2591*.

Friesen, A., & Namiri, K. (2006). Towards semantic service selection for b2b integration. *ICWE'06 Workshops July 10-11*.

Gartner. (2003). *Service-oriented architecture scenario*.

Gartner. (2007). Hype cycle for application development, 2007. *Gartner publications*.

Geurts, G., & Geelhoed, A. (2004). Business process decomposition and service identification using communication patterns. *Microsoft Architect Journal*.

Gordijn, J., & Akkermans, H. (2001). Designing and evaluating e-business models. *Intelligent E-Business, 16*(4), 11- 17.

Heckel, R., Lohmann, M., & Thone, S. (2003). Towards a uml profile for service-oriented architectures (pp. 6): Faculty of Computer Science, Electrical Engineering and Mathematics University of Paderborn, Germany.

Hepworth, M. (2004). A framework for understanding user requirements for an information service: Defining the needs of informal carers. *Journal of the american society for information science and technology, 55*(8), 695–708.

Hevner, A. R., March, S. T., & Park, J. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75-105.

Hobbers, J.-W., Tigthart, A., & Terlouw, T. (2007). Ten ways to identify services. *SOA Magazine*(XIII).

Ives, B., Olson, M. H., & Baroudi, J. J. (1983). The measurement of user information satisfaction. *Communications of the ACM, 26*(10), 785-793.

Jurisica, I., Mylopoulos, J., & Yu, E. (2004). Ontologies for knowledge management: An information systems perspective. *Knowledge and Information Systems*(2004 - 6), 380–401.

Koss, K., Knackstedt, R., & Beverungen, D. (2007). *Identification of services – a stakeholder-based approach to soa development and its application in the area of production planning.* Paper presented at the 15th European Conference on Information Systems, St. Gallen.

Koppenjan, J., & Groenewegen, J. (2005). Institutional design for complex technological systems. *International Journal of Technology, Policy and Management, 5*(3), 240-258.

Krafzig, D., Banke, K., & Slama, D. (2004). *Enterprise soa: Service-oriented architecture best practices.* NJ, USA: Prentice Hall.

Lankhorst, M. (2005). *Enterprise architecture at work - modelling, communication and analysis* (Vol. 18): Springer.

Larman, C. (2002). *Applying uml and patterns – an introduction to object-oriented analysis and design and the unified process* (2 ed.). Upper Saddle River: Prentice-Hall Inc.

Larsen, G., & Wilber, J. (2005, 15 Oct 2005). Asset lifecycle management for service-oriented architectures. Retrieved 23/01/2008, 2008

Lauesen, S., & Younessi, H. (1998). *Six styles for usability requirements.* Paper presented at the REFSQ.

Levi, K., & Arsanjani, A. (2002). A goal-driven approach to enterprise component identification and specification. *COMMUNICATIONS OF THE ACM, 45*(10), 45-52.

Lewis, G. A., Morris, E., Simanta, S., & Wrage, L. (2007). *Common misconceptions about service-oriented architecture.* Paper presented at the ICCBSS '07.

Ligthart, A., van Es, R., Gerwen, N., Graave, J., & van Rooij, R. (2006). *Flexibel omgaan met voortdurende veranderingen.* Den Haag: SDU uitgevers.

Luo, J., Montrose, B., Kim, A., Khashnobish, A., & Kang, M. (2006). Adding owl-s support to the existing uddi infrastructure, *IEEE International Conference on Web Services (ICWS'06).*

Mahajan, R. (2006). *Soa and the enterprise -- lessons from the sfr.* Paper presented at the ICWS '06.

Manes, A. T. (2005). The elephant has left the building. Retrieved 14/01/2008, 2008

Natis, Y. V., Pezzini, M., Schulte, R. W., & Iijima, K. (2006). *Predicts 2007: Soa advances:* Gartner.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Retrieved 11-12, 2007, from http://protege.stanford.edu/publications/ontology_development/ontology101.pdf

Nunamaker, J. F. J., & Chen, M. (1991). Systems development in information systems research. *Journal of Management Information Systems, 7*(3), 89 - 106.

OASIS (2006). Reference model for service oriented architecture 1.0. *Committee Specification*

Ong, C.-S., & Lai, J.-Y. (2004). *Developing an instrument for measuring user satisfaction with knowledge management systems.* Paper presented at the 37th Hawaii International Conference on System Sciences, Hawaii.

Overheid, W. v. d. (2007, november 2007). Overheid.NI webrichtlijnen. Retrieved 23-4-2008, 2008, from http://www.webrichtlijnen.nl

Pokraev, S., Wieringa, R. J., & Steen, M. W. A. (2004). *Towards semantic service specification and discovery.* Paper presented at the CAiSE'04 Workshops in connection with The 16th Conference on Advanced Information Systems Engineering, Riga Technical University, Riga, Latvia.

Quartel, D., Steen, M. W. A., Pokraev, S., & Sinderen, M. v. (2006). *A conceptual framework for service modeling.* Paper presented at the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06).

Reeves, C. A., & Bednar, D. A. (94). Defining quality: Alternatives and implications. *The Academy of Management Review, 19*(3), 419-445.

Schultze, U., & Leidner, D. E. (2002). Studying knowledge management in information systems research: Discourses and theoretical assumptions. *MIS Quarterly, 26*(3), 213-242.

Shishkov, B., Sinderen, M. v., & Quartel, D. (2006). Soa-driven business-software alignment, *IEEE International Conference on e-Business Engineering (ICEBE'06).*

Studer, R., Grimm, S., & Abecker, A. (2007). *Semantic web services - concepts, technologies and applications*. Berlin Heidelberg: Springer-Verlag.

Thong, J. Y. L., & Yap, C.-S. (1996). Information systems effectiveness: A user satisfaction approach. *Information Processing & Management, 32*(5), 601-610.

Turowski, K., Ackermann, J., Brinkop, L., Conrad, S., Fettke, P., Frick, A., et al. (2002). *Standardized specification of business components* (Memorandum): Gesellschaft für Informatik.

Vrijkorte, B. (2006). *Semantics in service oriented architectures*. University of Twente, Enschede.

Wijnhoven, F. (2008). Process design theory for digital information services. In A. Bernard & S. Tichkiewitch (Eds.), *Methods and tools for effective knowledge life-cycle management* (pp. 533-546). Heidelberg: Springer.

Wijnhoven, F., Belt, P. v. d., Ip, V., & Vet, P. v. d. (2003). Internal data market services: An ontology-based architecture and its evaluation. *Informing Science Journal, 6*.

Wijnhoven, F., & Kruijenbrink, J. (2008). Product-oriented design theory for digital information services: A literature review. *Internet Research, 18*(1), 93-120.

Womack, R. (2002). Information intermediaries and optimal information distribution. *Library & Information Science Research, 24*, 129–155.

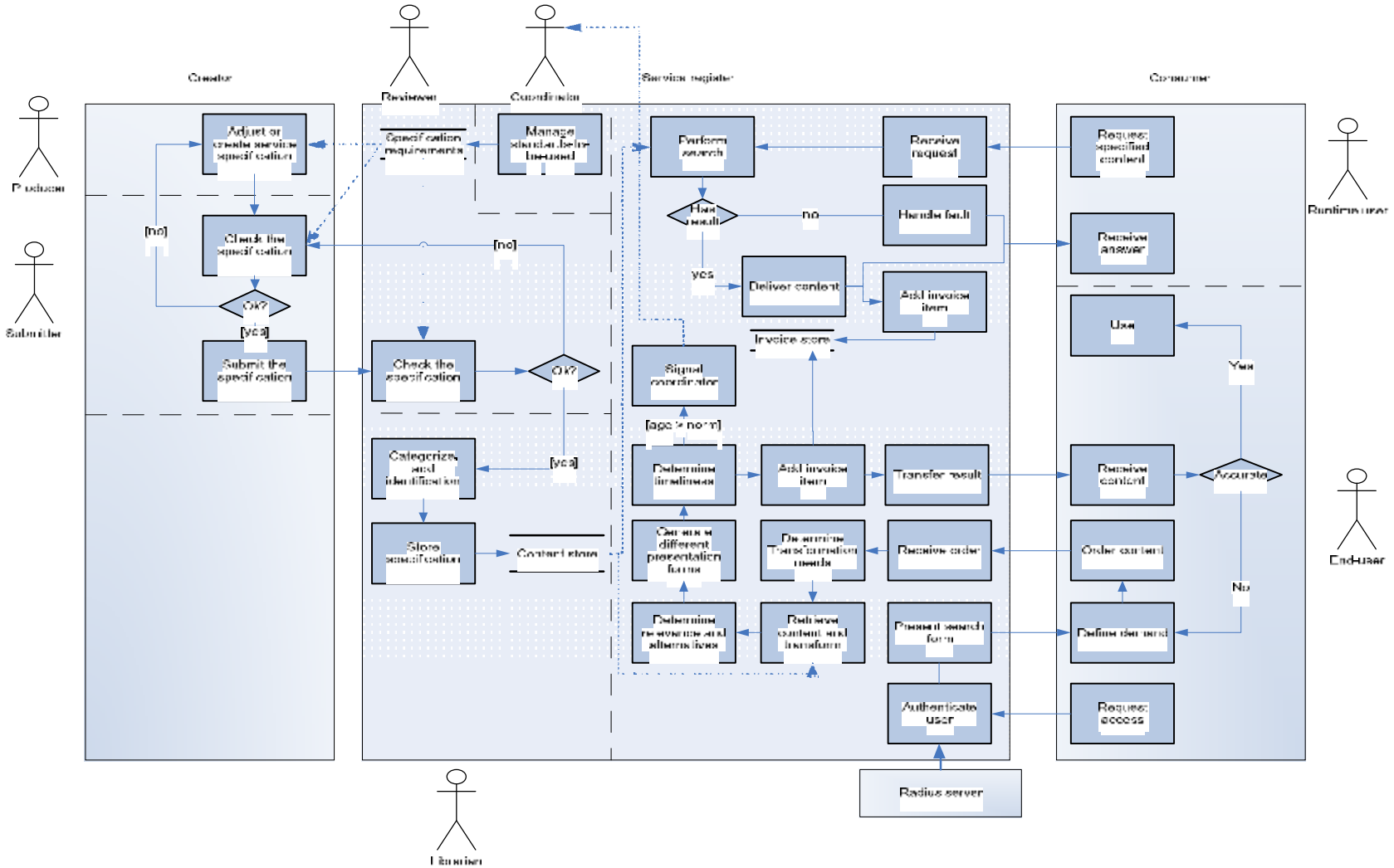
Zimmermann, B., Gnasa, M., & Harbusch, K. (2002). Modeling a corporate information system to improve knowledge management. *IsDBT 2002Workshops, LNCS 2490*, 435–449.

Zwaan, H., & Steenbakkers, W. (2007). *Puzzelen met portfolio's, ii governance* (Vol. 1). Den Haag: SDU.

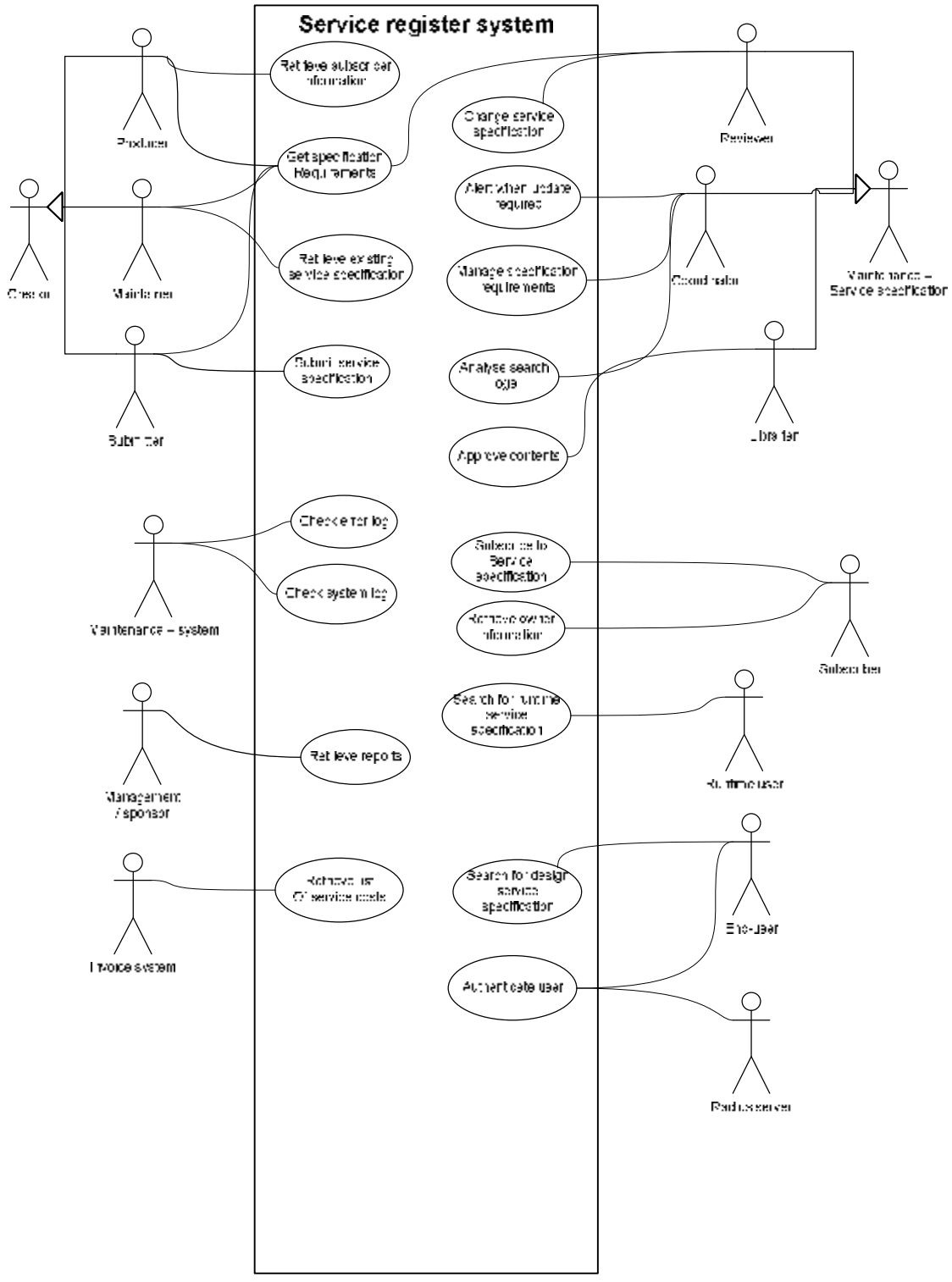
List of Figures

List of tables

Appendix A. Process activities



Appendix B. Use case diagram Sure 4 U



Appendix C. Architecture of a service register

A full architectural model is found in Appendix D, this section describes the transformation of the earlier mentioned design elements.

Environment

The actors which are involved in the exchange of a service specification are the first concepts of an Archimate architecture, and belong to the 'Environment' which is 'above' the business layer. As the actual actors are likely to be inconsistently present, they are assigned to a role, which is supposed to be consistent. These roles are mentioned before, and the Archimate representation is given by in Figure 25.

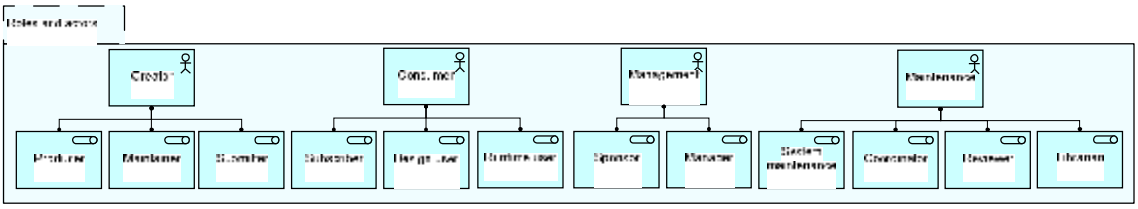


Figure 25 Archimate represenation of actors

Business layer

Archimate splits this layer to distinct between external (services meaningful for the environment) en internal (the realization processes for these services) behaviour. To make this possible, the business functions are to be transformed. By doing so, business processes are grouped into services and processes which implement these services. The naming of the concepts is from the environment point of view, so 'Design service specification acquiring' means that some actor in the environment can retrieve design part of the service specification using this service. The processes are high level here, the connections with actors and further decomposition is shown in the architecture in Appendix D.

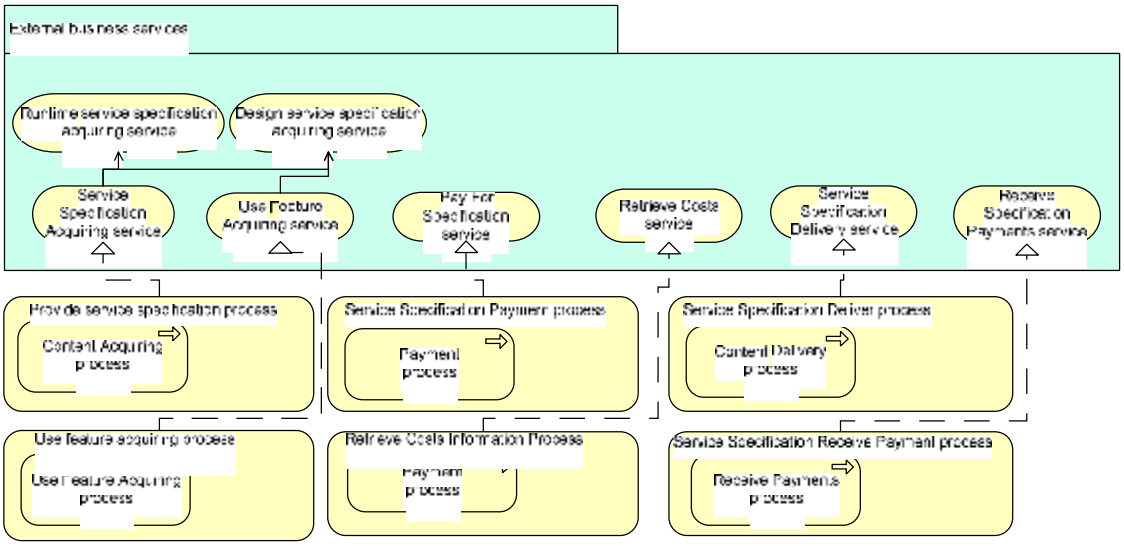


Figure 26 Business layer in Archimate

Process layer

As stated before, Archimate is based upon a service oriented view, in which processes implement the service a layer offers. This results in an other definition of processes and functions as the theory design. However, both layers are meant to show how the layer above is implemented in terms of application (functions). The identified processes in Section 5.2.3 implement the business layer and are translated into Archimate's language using the split layer design. Figure 27 shows the results.

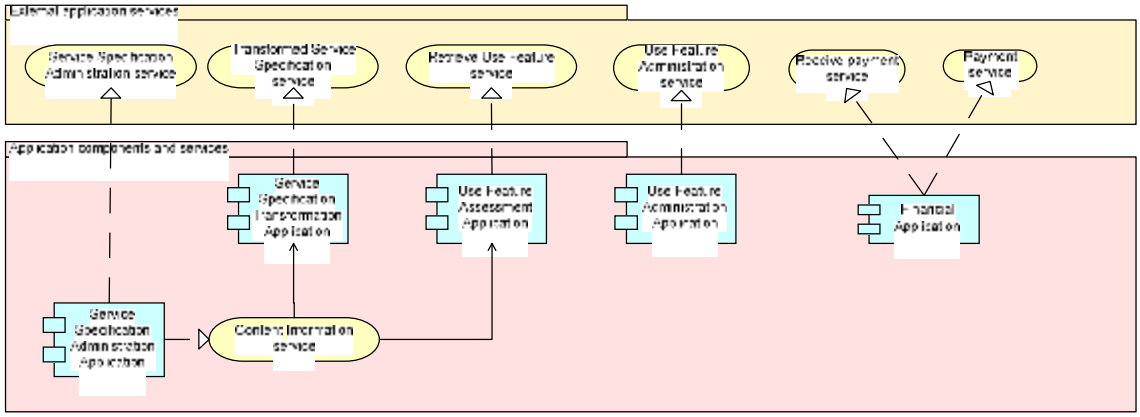


Figure 27 Process layer in Archimate

Infrastructure layer

The infrastructure layer as mentioned in the design theory identifies the means necessary to implement the processes. Archimate also has a third layer, named the 'Technology layer'. As the name already suggests, Archimate focuses on the technology as the means to implement the 'Application layer'. But this is just one of the three identified sorts of means, namely the 'Information Technology' means from the design theory. This section discusses the three sorts for their representations in Archimate.

Information means

The knowledge of contracts, creators, consumers etc. and lists of available service specifications is important as this information may be used to determine what can be offered. Accordingly, these information means should be taken into account when creating an architecture. Archimate deals with this by using the 'Business object' (business layer), 'Data object' (Application layer) or 'Artifact' (Technology layer) concept. This concept can be used to represent an Information object. As an example, a model is given which illustrates the use of an 'Artifact' in Figure 28. Here, an artifact containing a list of available service specifications is assigned to the database server, and is used by the 'Service specification administration Application' to determine if the content should be acquired. Other Information means can be modelled in the same way.

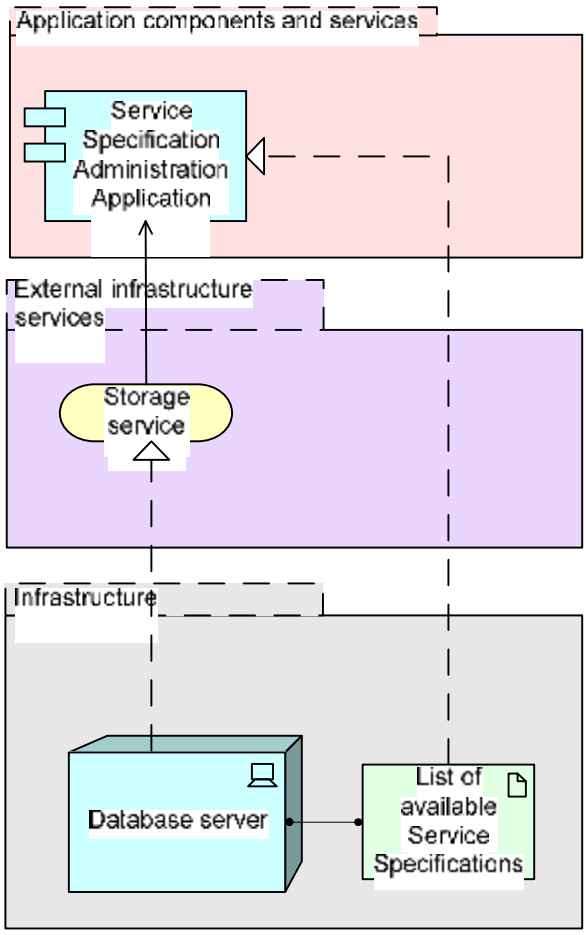


Figure 28 Example of use of artifact concept - Archimate

Organizational means

Three main categories of organizational means can be identified: Organizational units, procedures and rules, and network. The organizational units can be represented in Archimate with an actor concept, associated with a specific Process/Function/Collaboration. Procedures and rules can be represented by the 'Business Object' concept, and are linked to the Process/Function/Collaboration they apply to. Although having a network of (i.e.) consumers is well understood and should speak for itself, it can be represented by the use of multiple actor concepts. This is displayed in Figure 29 for a service register case.

The search options object represents the use of search options in that process. The assignment between the coordinator and the data object timeline report indicates that the coordinator is involved when a timeline report is generated. He is responsible for the handling of that report.

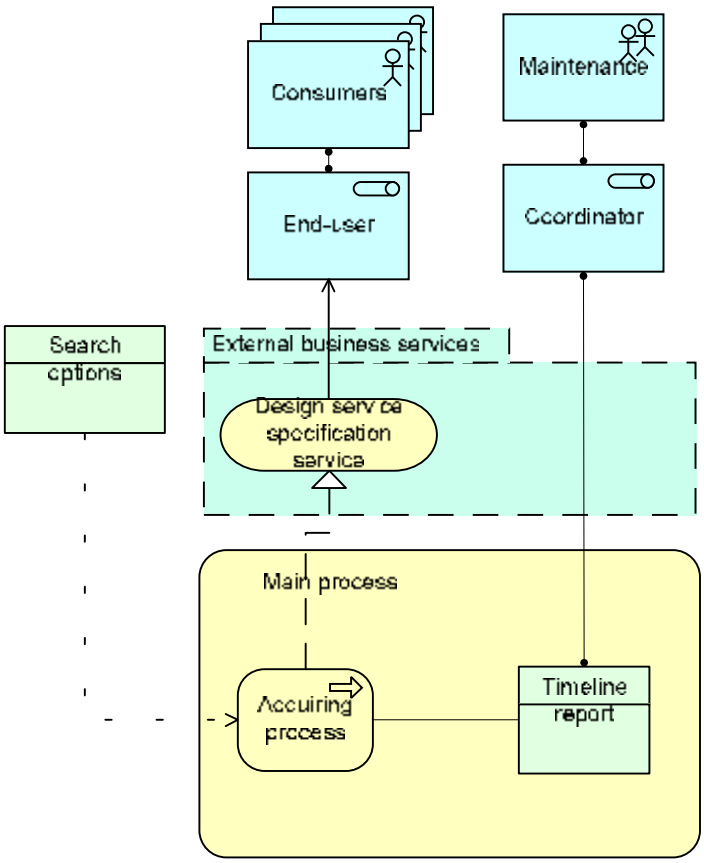


Figure 29 Examples of Procedures, Departments and Networks in Archimate

Information Technology means

The third category of means is the Information Technology means (IT). This is analogous to the third layer of Archimate, the Technology layer. Concepts of the theory can be classified into Tools, Servers and Devices, and connections. Tools are considered to be (parts of) applications in Archimate, while the others are placed in the Technology layer. Figure 30 shows the concepts for the transformation tools with related services and technology.

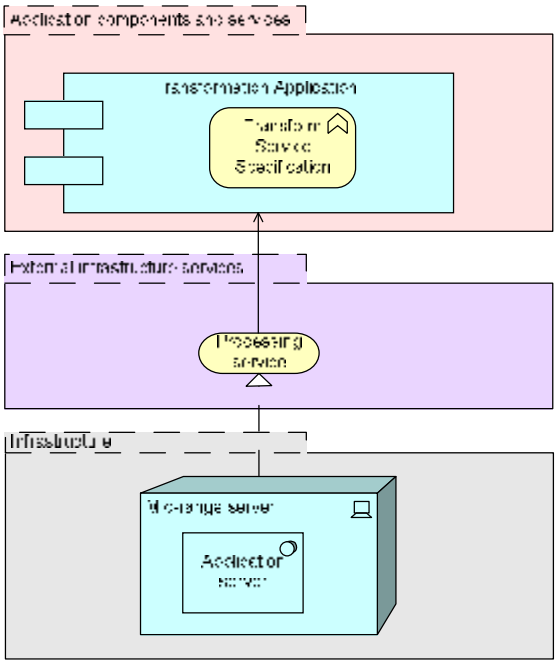
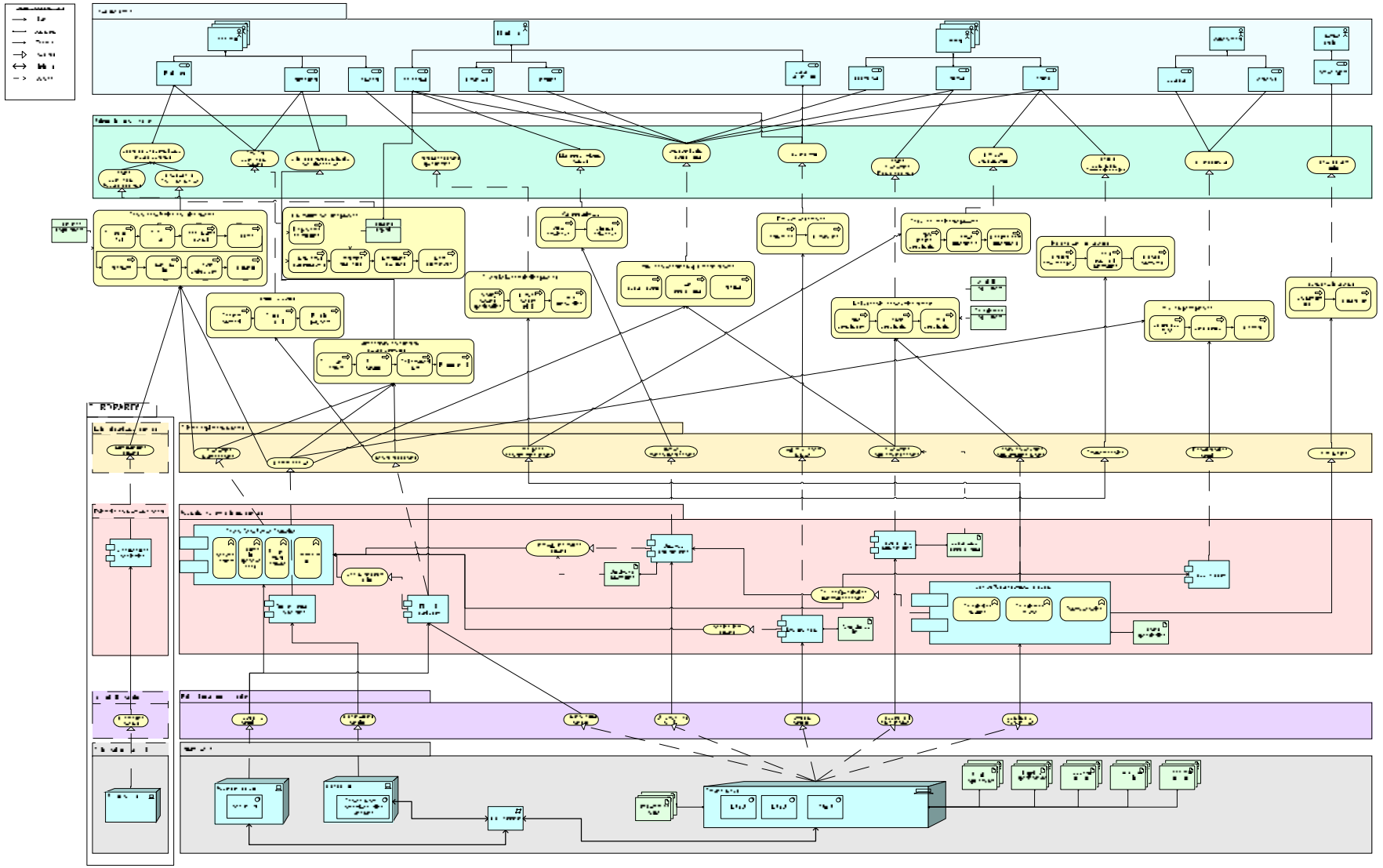


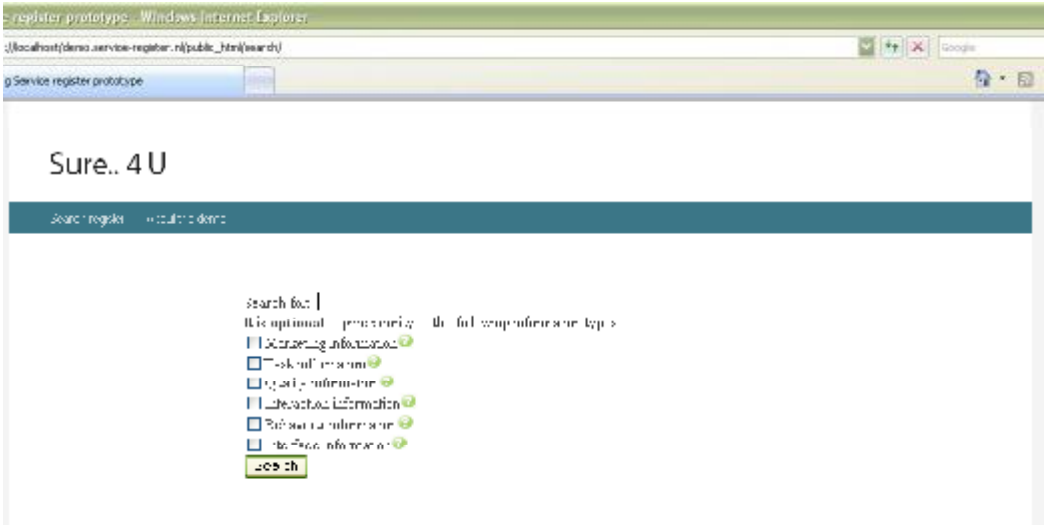
Figure 30 Transformation application with infrastructural means

Appendix D. Archimate service register architecture



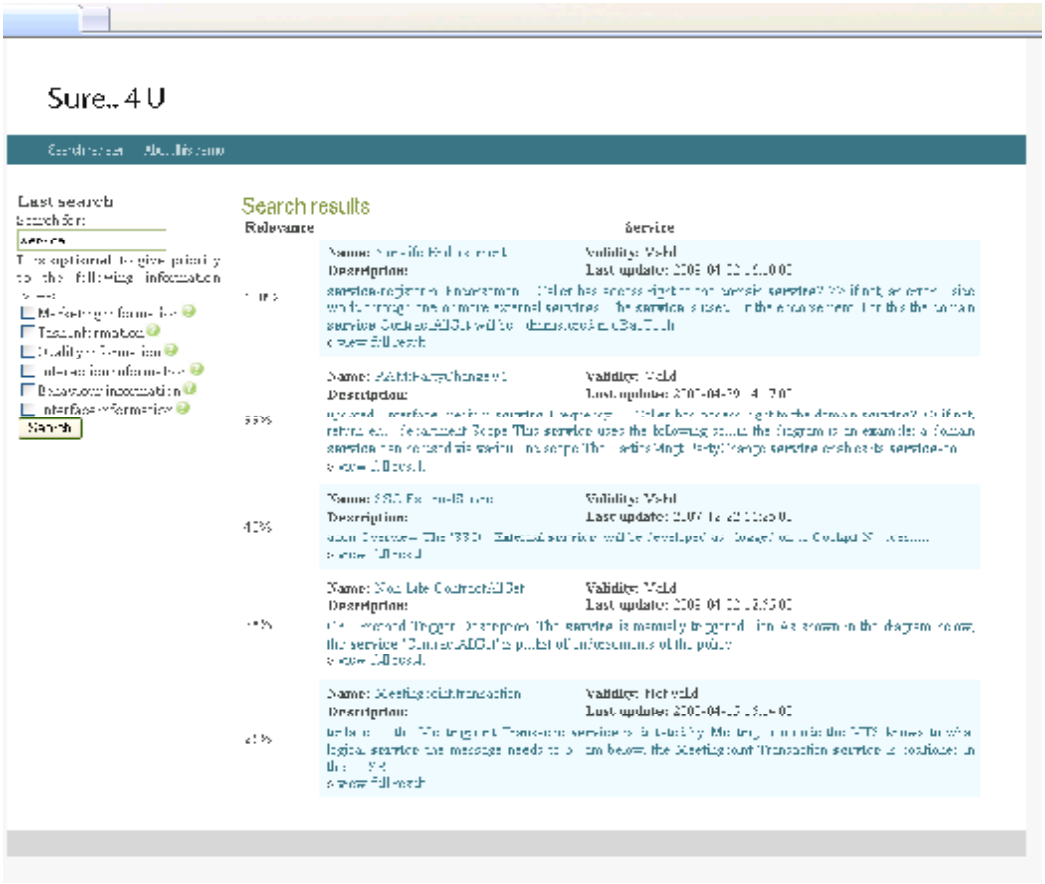
Appendix E. Prototype screenshots

Initial screen, providing search functionalities

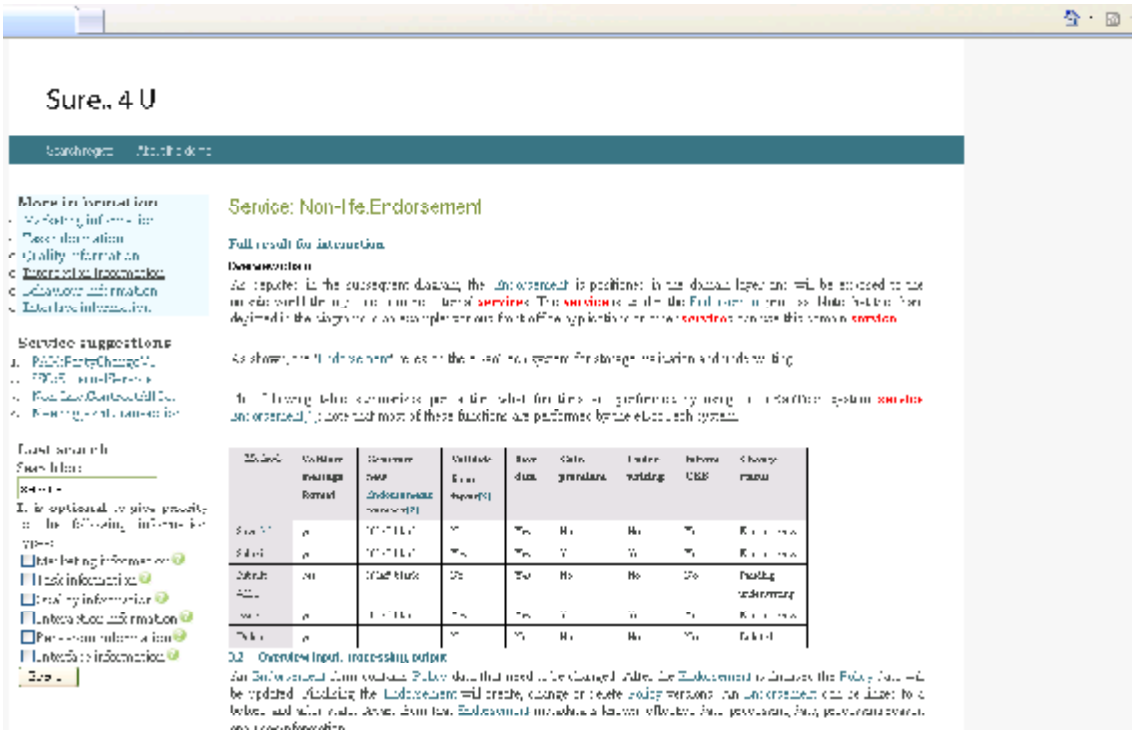


Error message when empty query is provided

List of results



Information level interaction for a service specification



Appendix F. Case results datasheet

PROTOTYPE	content c1	c2	c3	c4	accuracy a1	a2	format f1	f2	ease of use e1	e2	timeliness t1	t2	familiarity with SOA adrt1	Service Register adrt2	experienced 1=no, 2=yes adrt3	mean	overall	
	5	4	3	4	4	4	4	4	3	3	5	4	4	3	2	3.92	3.00	
	3	3	3	3	3	3	3	3	3	3	4	3	4	3	2	3.08	3.00	
	4	4	5	4	4	4	5	4	4	5	3	4	2	1	2	4.17	4.00	
	4	4	5	4	5	4	2	3	2	4	4	3	4	3	2	3.67	4.00	
	3	3	3	3	4	4	2	3	3	4	4	3	5	4	2	3.25	3.00	
	4	4	4	4	4	4	3	3	2	3	4	4	3	3	2	3.58	4.00	
	2	2	2	2	3	3	2	2	2	3	2	3	3	1	2	2.33	2.00	
	4	4	3	4	4	4	3	2	3	4	4	4	4	4	2	3.58	4.00	
	5	5	3	3	3	3	3	3	3	3	3	3	5	5	1	3.33	3.00	
	3	3	3	3	3	3	3	3	3	3	3	3	5	5	1	3.00	3.00	
	5	3	3	3	4	3	2	2	3	3	5	3	5	5	2	3.25	3.00	
	3	3	2	2	3	3	4	3	3	4	3	3	5	3	2	3.00	4.00	
	5	5	5	3	5	5	5	5	5	5	5	5	3	1	2	4.83	5.00	
	5	3	4	2	4	4	3	2	2	2	4	3	4	4	2	3.17	3.00	Mean calculated by accumulated scores
	4	4	3	3	3	4	4	4	3	4	5	3	5	4	2	3.67	4.00	3,51
	5	4	4	4	4	4	4	4	4	4	4	4	3	3	2	4.08	4.00	
	4	5	4	4	4	4	5	5	4	5	5	4	5	4	2	4.42	4.00	Mean based on overall score
	4	4	4	5	4	4	3	4	3	4	4	4	4	3	2	3.92	4.00	3,38
	4	4	3	3	4	3	4	3	4	4	4	4	4	4	2	3.67	3.00	
	4	3	3	2	3	3	2	2	3	2	3	2	2	2	2	2.67	2.00	
	2	2	2	2	4	4	4	4	3	4	2	4	4	4	0	3.08	2.00	
mean of question score	3.90	3.62	3.38	3.19	3.76	3.67	3.33	3.24	3.19	3.62	3.81	3.48	3.95	3.29	1.81			
standard deviation of question score	0.92	0.84	0.90	0.85	0.81	0.66	0.99	0.92	0.75	0.84	0.91	0.68	0.95	1.20	0.50			
aspect mean		3.52			3.71		3.29		3.36		3.64							
standard deviation of aspect score		0.92			0.59		0.96		0.84		0.81							