An Enterprise Ontology-Based Approach to Service Specification

Linda I. Terlouw and Antonia Albani

Abstract—In recent years, the Web Service Definition Language (WSDL) and Universal Description Discovery Integration (UDDI) standards arose as ad hoc standards for the definition of service interfaces and service registries. However, even together these standards do not provide enough basis for a service consumer to get a full understanding of the behavior of a service. In practice, this often leads to a serious mismatch between the provider's intent and the consumer's expectations concerning the functionality of the corresponding service. Though additional standards have been proposed, a holistic view of what aspects of a service need to be specified is still lacking. This paper proposes a service definition, a service classification, and service specification framework, all based on a founded theory, the Ψ-theory. The Ψ-theory originates from the scientific fields of Language Philosophy and Systemic Ontology. According to this theory, the operation of organizations is all about communication between and production by social actors. The service specification framework can be applied both for specifying human services, i.e., services executed by human beings, and IT services (i.e., services executed by IT systems).

Index Terms—Service specification, enterprise ontology, service-orientation, IT services, human services

1 Introduction

In today's economy, cooperations between organizations tend to be highly subject to change. Traditional static supply chains make way for dynamical organization networks. A lower price, a higher quality product, a larger product portfolio, or a faster delivery; all these factors act as reasons for changing the own organization and to adapt the relationships with business partners. The pace in which these changes need to occur, result in high demands on the supporting information systems of organizations. The concept of *service-orientation* helps organizations to deal with this required interoperability and flexibility. Available international standards enable organizations to quickly connect their information systems and the notion of *orchestration* makes the relation between business processes and the supporting IT services explicit.

In service-orientation, service providers interact with service consumers by offering them services. In this interaction, both parties have certain *expectations* of each other's *responsibilities*. Serious problems can occur if these expectations are not made explicit in a kind of contract, also known as *service specification*. Let us consider an example of a service called "consult legal expert." We distinguish between two different kinds of services, "human services" and "IT services." Human services are tasks executed by human beings, whereas IT services are tasks executed by IT systems. The example service "consult legal expert" is of the first type. The role of service consumer is fulfilled by an employee of a retail company, the role of provider by an legal expert working for a law firm. The employee of the retail company may expect the service to have the following behavior: The expert deals with all his problems for the planned duration of the visit, let's say 30 minutes. The legal expert, however, may regard the consultation as dealing with one specific problem with a maximum duration of 30 minutes. When the legal expert is requested to deal with two unrelated problems and the visit takes 29 minutes, the employee of the retail company may be surprised to receive an invoice for two consultations instead of one. Likewise, an ill-specified IT service can also lead to misunderstandings. Take for example an IT service for calculating a mortgage amount. If we only know the input and output of the service, then we have no idea how soon the results are returned (e.g., within 2 minutes or 3 days). Also, we do not know whether or not this calculation is legally binding when requesting a quotation for a mortgage based on this calculation.

Though the *Web Service Definition Language* (WSDL) [1] standard enables provider and consumer to have a common view on the *interface* of the service and the *Universal Description Discovery Integration* (UDDI) [2] can be used as a means for publishing some service information, they together do not provide enough basis to deal with questions regarding for instance the semantics of provided functionality, the semantics of the input and output parameters, the availability of the service, and the costs of calling the service. The original intent of the UDDI was to enable worldwide runtime service discovery, but it even falls short in enabling good design-time discovery.

The recognition of the importance of a comprehensive service specification becomes clear when looking at the efforts of numerous standardization organizations to develop service-related standards. Yet, the other side of the coin is that this development has led to a morbid growth of

• L.I. Terlouw is with Icris B.V., Martinus Nijhoffhove 2, 3437 ZR Nieuwegein, The Netherlands. E-mail: linda.terlouw@icris.nl.
• A. Albani is with the Institute of Information Management, University of St. Gallen, Müller-Friedberg-Strasse 8, 9000 St. Gallen, Switzerland. E-mail: antonia.albani@unisg.ch.

Manuscript received 26 Apr. 2010; revised 19 Sept. 2010; accepted 18 May 2011; published online 16 June 2011.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2010-04-0069.
Digital Object Identifier no. 10.1109/TSC.2011.38.

specification standards. In our eyes, it is time to take a step back and focus on *what* one should specify instead on *how* it should be specified. We take a different approach and base our work on a solid theory: the Ψ-*theory* [3].

The main contribution of this paper is, therefore, a method for service-orientation including a service definition, a generic service specification framework, and a distinction between different types of services. The method is based on the Ψ-theory. This theory regards organizations as social systems and sees IT systems as support for social actors in performing communication-related activities and production-related activities. Based on this theory, we see many similarities in the specification of human services and IT services. A definition of the notion of "service" is, therefore, given which can be applied to both types of services and that acts as a foundation for our framework. Our generic framework can be used by service providers for specifying both human services and IT services to enable service consumers 1) to find a certain service, 2) to determine whether the provided functionality corresponds to their needs, and 3) to know how to use a certain service.

The structure of this paper is as follows: We start by discussing related work that deals with the problem of service specification in Section 2. In Section 3, we give a brief overview of the Ψ-theory and explain the notion of "service" using this theory. We provide a service definition and present a classification in six types of services: ontological human services, infological human services, datalogical human services, ontological IT services, infological IT services, and datalogical IT services. Section 4 presents our Generic Service Specification Framework and its derivation based on the Ψ-theory. This framework shows which aspects of a service need to be specified in order for a potential service consumer to understand what a service does. An example case of an insurance company is used to show the application of the generic service specification framework in Section 5. In addition, an evaluation of the aspects of the service specification framework has been conducted at a global provider of financial solutions, presented in Section 6. We conclude the paper with a discussion of the results and recommendations for further research in Section 7.

2 Related Work

As mentioned in Section 1, the UDDI standard [2] is currently most popular in practice as a standard for service registries. This XML-based standard states both what to specify (to some extent) and how to specify it. The UDDI only prescribes a very small set of information that has to be specified. It has possibilities for describing the service function in the T-Models, but these T-Models are unstructured. Therefore, there is no consistency across specifications, which makes automated discovery and also manual discovery difficult. Also, in each individual case one again has to think about which aspects to describe in the T-models.

In the web service standards, community researchers and practitioners state that the service contract consists of an interface definition (WSDL), a message structure definition (XML Schema), and, if required, a *policy* definition. These policies specify rules and constraints that must be met by the consumer before it can access the web service. Policies are used to specify aspects of a service that cannot be specified in WSDL or XML schema. These aspects include among others technical limitations, choice of security protocol, privacy constraints, and type of reliable messaging used. These policies do not prescribe what one should specify about a service, but they provide a generic structure for specifying several aspects. WS-Policy [4] is the proposed XML-based standard that allows providers to specify their policies and that allows consumers to specify their policy requirements.

Also, two standards for specifying the *Service Level Agreement* (SLA) are evolving; Web Service Level Agreement (WSLA) [5] proposed by IBM and WS-agreement [6] proposed by the Open Grid Forum (OGF). These standards focus on specifying the agreements made by service consumers and providers and the way to evaluate and measure these agreements. In this sense, they have a broader scope than only specifying the service itself. However, they focus mainly on quality aspects like, for instance, performance.

More comprehensive frameworks are the business component specification framework [7] and the faceted specification approach of Walkerdine et al. [8]. Though the aspects mentioned in these frameworks seem plausible, there is no clear rationale why the frameworks are constructed as they are. For instance, Ackermann et al. [7] propose seven different levels in their framework. To us, it is unclear why we need these seven levels (why not three, five, or eight?) and if there is a hierarchical relationship between these levels like the name "level" seems to imply. Additionally, these standards are not based on a founded theory, therefore it is not clear why certain aspects need to be specified whereas others are not taken into consideration.

Researchers in the area of Artificial Intelligence proposed semantic web service standards like OWL-S [9], [10], WSMO [11], and WSDL-S [12] for extending the UDDI. The goal of semantic web services is thoroughly specifying every aspect of a service in order to enable automatic matching of supply of and demand for services. It takes a lot of effort (if feasible at all) for a large enterprise to specify everything into so much detail that automatic matching on runtime becomes possible. At this moment, none of the semantic web service approaches are popular in industry. Though industrial partners participate in research projects, we see little (if any) semantic service registries in real SOA environments. In practice, the matcher of supply and demand is still a human being and not a machine.

All in all, the work from Ackermann et al. [7] and Walkerdine et al. [8] is most closely related to our work. We take a different approach by building our approach on a sound theory enabling us to explain why we need to specify certain aspects. Also, our work has similarities with the work from the semantic web services world. But we do not try to realize automated discovery on a worldwide scale. We aim at providing a framework for practitioners that they can use as a reference for knowing which aspects of the service they need to specify (with a clarification why they need them). They can use different standards to actually specify these aspects. In summary, we focus on what aspects of a service need to be specified and not how they should be specified.

3 Defining Service Based on the Ψ-Theory

In this section, we give our definition of the notion of "service." As mentioned in the previous section, we base our ideas on the Ψ-theory, the theory that underlies the notion of *Enterprise Ontology* [13]. Enterprise Ontology is defined as the model of the organization that is fully independent of its implementation.

3.1 The Ψ-Theory

The Ψ-theory [3] finds its roots in the scientific fields of Language Philosophy, in particular the Language Action Perspective (LAP) [14], [15], and in Systemic Ontology [16]. It focuses on the use of language to achieve agreement and mutual understanding [17]. By applying the Ψ-theory one can disentangle the essential knowledge of the construction and the operation of the organization of an *enterprise*, by which we mean a commercial or nonprofit company as well as a network of enterprises. This essential enterprise model is called the Enterprise Ontology. The theory consists of several axioms and one theorem. In this section, we give a short summary of the Ψ-theory. We only discuss the parts of the theory that we need for developing a service specification framework, viz.: the operation axiom, the transaction axiom, the distinction axiom, and the organization theorem. A complete overview of the theory is available in the book [13] and the papers [18], [19], [20], [21].

3.1.1 The Operation Axiom

The first axiom, the operation axiom, focuses on the different types of acts that actors in organizations (people, also called subjects) perform and the results of these acts. It states the following [13]:

Axiom 1. *Actors perform two kinds of acts: production acts and coordination acts. These acts have definite results: production facts and coordination facts respectively. By performing production acts, actors contribute to bringing about the function of the organization. By performing coordination acts, actors enter into and comply with commitments regarding production acts. An actor is a subject fulfilling an actor role. Actor roles are elementary chunks of authority and responsibility.*

What are these so called production and coordination acts the axiom speaks about? And why do we need to distinguish between them? First, let us look at the production acts. Production acts are acts that deal with the delivery of material or immaterial goods by actors to their environment. Their results are production facts. Examples of production acts dealing with material goods are manufacturing and transporting. Their corresponding production facts are "Product P has been manufactured" and "Product P has been transported." For immaterial goods, examples are deciding and judging. Their corresponding production facts are "Decision D has been made" and "Judgement J has been made." Coordination acts serve a totally different purpose than production acts, though they are executed by the same actors. They do not directly contribute to the production of goods, but they *coordinate* the execution of production acts. An example of a coordination act and its corresponding fact is the request for manufacturing a product and "The production fact "Product P has been manufactured" has been requested." In the next paragraph, we will see that the different types of coordination acts form a limitative list.

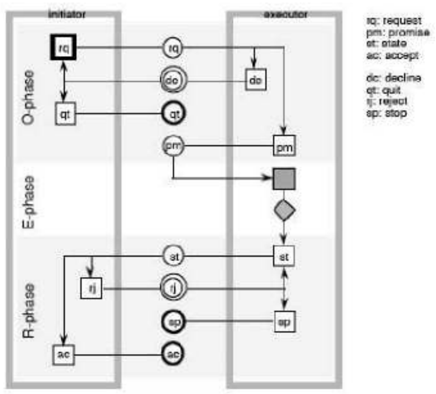


Fig. 1. Standard transaction pattern.

3.1.2 The Transaction Axiom

The second axiom, the transaction axiom, further looks into the coordination acts. It states the following [13]:

Axiom 2. *Coordination acts and production acts always occur in particular patterns. These patterns are paths through one universal pattern, called transaction. The result of carrying through a transaction is the creation of a production fact.*

A transaction evolves in three phases, the *order* phase (O-phase), the *execution* phase (E-phase) and the *result* phase (R-phase), see Fig. 1. Two actor roles are involved in such a transaction, the *initiator*, who starts and completes the transaction, and the *executor*, who performs the production act. In the order phase the initiator and the executor try to reach agreement about the intended result of the transaction, i.e., the production fact that the executor is going to create as well as the intended time of creation. In the execution phase, this product is created by the executor, and in the result phase both actors try to reach agreement about the fact that has been produced. The so-called *basic transaction pattern* consists of the request, promise, state, and accept coordination acts. An example of this basic pattern looks as follows:

1. Person A *requests* person B to *manufacture a car.*
2. Person B *promises* person A to *manufacture a car.*
3. <actual delivery of the manufactured car>.
4. Person B *states* to person A that he has *manufactured a car*.
5. Person A *accepts* from person B that he has *manufactured a car* (the car conforms to his expectations).

Transactions will conform to this basic transaction pattern in a *happy scenario*, i.e., everything goes as it should go. However, in reality the initiator and executor may dissent in two of the states: 1) the requested state and 2) the stated state. In the first case, the executor may (instead of promising) respond to a request by declining it (see Fig. 1).

In the second case, the initiator may (instead of accepting) respond to a statement by rejecting it. By allowing these acts, a transaction can end up in a *discussion state*. Dietz describes that in this situation the two actors must sit together, discuss the situation at hand, and negotiate about how to get out of it. When the basic pattern is expanded with these two dissent patterns, we get the *standard transaction pattern*. The standard transaction pattern is shown in Fig. 1. The *complete transaction pattern* is constituted by the standard pattern and four *cancellation patterns*. Cancellation patterns concern the revocation of a request act, promise act, state act, or accept act.

3.1.3 The Distinction Axiom

The third axiom, the distinction axiom, is concerned with the different abilities of a human being that are involved in the activities they perform. The axiom states the following [13]:

Axiom 3. *Three distinct human abilities play a role in the performance of coordination acts and production acts: the forma, informa, and performa abilities.*

How are these human abilities relevant for coordination acts on the one hand and production acts on the other hand? The forma ability deals with the form aspects of communication and information. Applying this to coordination acts, this means actors should have a way to utter and perceive information. Information should be expressed in a particular language or code scheme that both the initiator and the executor of a transaction understand. This is also known as syntactic (or significational) understanding. One might think, for instance, of information written in English. The informa ability concerns the content aspects of information and communication. In order to communicate, the initiator should formulate information in a way that the executor can interpret. In other words, the initiator and the executor should semantically be in agreement with each other and share the same thoughts. This is also called intellectual understanding. The performa ability states that new information and knowledge can be created through communication between the initiator and executor. Looking at coordination acts, this means that actors can expose and evoke commitments and it indicates social understanding between the initiator and executor. For the production acts, we see a similar distinction. The forma ability is concerned with the form aspects of information in terms of information transmission and storage. This type of production act is known as a *datalogical act*. A transaction that contains a datalogical act is called a *datalogical transaction* (D-transaction). The informa ability states that information can be reasoned, computed or deduced. This type of activity is known as an *infological act*. A transactions is called an *infological transaction* (I-transaction) if it includes this type of production act. The performa ability concerns making decisions, judgements, or creating material things such as products. This is what we call an *ontological act*. A transaction that includes an ontological act is known as an *ontological transaction* (B-transaction).

3.1.4 The Organization Theorem

We just presented three of the axioms of the Ψ-theory. Together with the composition axiom, which we did not discuss, they provide the basis for the organization theorem. This theorem provides a concise, comprehensive, coherent, and consistent notion of enterprise, such that the (white-box) model of an enterprise may rightly be called its ontological model [13]. It states the following [13]:

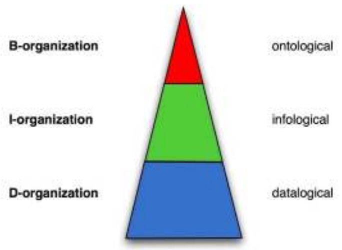


Fig. 2. The three aspect organizations.

Theorem 1. *The organization of an enterprise is the layered integration of three aspect organizations: the B-organization, the I-organization, and the D-organization.*

Fig. 2 shows the three aspect organizations. The *B-organization* concerns the essence of the enterprise. It consists of actors who directly contribute to the enterprise's goals and functions by performing ontological production acts. These actors are known as *B-actors* and are able to perform B-transactions, the ontological transactions we defined in the previous paragraph. B-actors are, for instance, consultants or sales persons. The *I-organization* embraces the content aspects of information and knowledge in the enterprise [13]. Actors in the I-organization, who are called *I-actors*, bring changes to information and knowledge by performing infological production acts. In other words, I-actors perform I-transactions. Business controllers are typical actors in the I-organization producing infological things. The *D-organization* deals with the documentation of information in the enterprise and only takes into account the form of information. To achieve this, actors in the D-organization perform datalogical production acts and thus D-transactions. These actors are known as *D-actors*, who are for instance archivists.

3.2 The Notion of "Service"

Before thinking about how we can specify services in a service specification framework, we need to define what a service is. Economists and business scientists have been debating about this *"service"* notion for more than two centuries [22]. Often, the definitions in business literature limit the service notion to the delivery of immaterial goods. The adoption of the notion of "service" by computer scientists and IT practitioners has been more recent. In both the business science field [23], [24], [25], [26] and the computer science field [27], [28], [29], [30] a service is regarded as an interaction between a requesting party (often called consumer or customer) and an offering party (often called provider or supplier). The offering party is able to produce a certain value that is requested by the other party. But even with this common notion a precise definition and mutual understanding of the term service is missing. Let us have a look at the definition given by the Open Group [29]. It says that a *service*

- is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit; provide weather data, consolidate drilling reports),
- is self-contained,
- may be composed of other services, and
- is a "black-box" to consumers of the service.

This definition is as vague as the other definitions mentioned above and one could discuss every single statement of the definition. For example, what is a business activity? The Open Group mentions "check customer credit" or "provide weather data" as business activities, but are these really business activities or are they only computational acts? What about a business activity concerning the "manufacturing of a car"? Such a business activity has a completely different granularity as the ones mentioned in the definition. What is self-contained? If a service is composed of other services is it still self-contained? What is precisely meant by a black-box when a service is also defined to be an activity? What about communication activities, e.g., to call the service or to accept/reject the requested result?

According to the Ψ-theory, the previous paragraphs have shown that the operation of organizations is all about communication between and production by social actors. Is not the main concern of service-orientation to support the operation of an organization and therefore also to support the communication between and production by social actors? Because the Ψ-theory describes the interaction between the requesting party and the offering party in a very formal way, it provides a basis for formalizing the notion of service. Based on this theory we have elaborated on the notion of service in [31] and we will summarize in the next paragraphs the main results which are of relevance for the specification of services.

The definition of service is based on the standard transaction pattern as introduced in Fig. 1. Though a service has many similarities with a transaction in the Ψ-theory, they are not equal. While the transaction includes all acts of the initiator and the executor, the service concept emphasizes more on the executor than the initiator side. We, therefore, define a service as a part of a transaction rather than a whole transaction.

Definition 1. *A service is a universal pattern of coordination and production acts, performed by the executor of a transaction for the benefit of its initiator, in the order as stated in the standard pattern of a transaction (see Fig. 1). When implemented it has the ability*

- *to get to know the coordination facts produced by the initiator and*
- *to make available to the initiator the coordination facts produced by itself.*

When looking at the standard transaction pattern, everything except the coordination acts of the initiator (request, quit, reject and accept) are part of the service. But in order to communicate with the executor of the service, the initiator needs to be aware of the standard transaction pattern.

This definition of a service just given is a very generic one, since it holds for two kinds of actors, *human actors* and *IT systems* and three kinds of production acts, namely *datalogical*, *infological*, and *ontological*.

Services executed by human actors or IT systems only differ in the way they are implemented; human services are implemented by human beings, whereas IT services are implemented by IT systems. IT systems assist human actors in their activities, therefore parts of a human service may also be executed by IT systems. For both human actors and IT systems, we can distinguish between communication acts and production acts on the datalogical, infological, and ontological level as described in the organization Theorem 3.1.4 (though at ontological level machines can only support responsible actors by mimicking decision making by applying decision rules, because machines can never reach true social understanding and cannot create really new, original things). Examples of datalogical production acts are storing, copying, transmitting of documents or data. Acts such as reasoning, computing, deriving, or reproducing knowledge are examples of infological production acts and the acts concerning the creation of original new things, such as creating material products or making judgments are examples of ontological production acts.

The basic concept of dealing with coordination and production aspects between an initiator and an executor party as defined in Ψ-theory and in the generic service definition given above allow us to distinguish between six different types of services:

- Ontological human service.
- Infological human service.
- Datalogical human service.
- Ontological IT service.
- Infological IT service.
- Datalogical IT service.

All service types conform to the definition of service given above, following the same service pattern and the described abilities. They only differ in the way they are implemented, either by human actors (eventually supported by IT systems) or by IT systems and in the different kinds of coordination and production acts, as described above. With the service definition and the different types of services in mind we can now introduce the service specification framework.

4 The Generic Service Specification Framework

Fig. 3 depicts our Generic Service Specification Framework. We base this framework on the generic service definition provided in Section 3.2. Our rationale for applying this generic approach is that in our eyes the same aspects need to be specified for services executed by IT systems as for services executed by human beings. Though the aspects themselves are equal for both types of services, it may be the case that the form in which these aspects appear is quite different. We will see some examples in Section 5.

Now, let us explain why the framework looks the way it does. When we recall the service definition, we see that for calling a service basically three things need to be known to the service consumer, namely information on 1) who provides the service (the executor), 2) which production

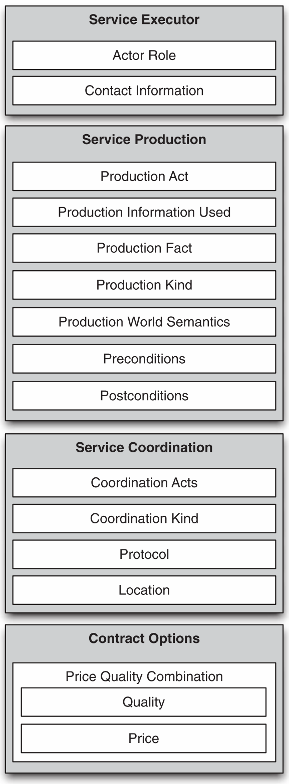


Fig. 3. The generic service specification framework.

act is to be performed by the provider, and 3) how to interact with the service executor by executing and dealing with coordination acts. We translate these information needs into three main *areas of concern*, i.e., *service executor*, *service production*, and *service coordination*, respectively. As transactions can have a commercial as well as a nonprofit character, we add *contract options* as an additional area of concern; the consumer needs to know what he gets for which price.

The *service executor* area of concern defines who is the provider of the service and contains two aspects, namely the *actor role* and *contact information*. The actor role aspect specifies the role of the actor that takes final responsibility for the service. In human service, this is the actor role of the human executing the production fact, whereas in IT service, this is the actor role of the human responsible for executing the production act, but who has delegated his responsibility to an IT system. This information can be gained from two types of diagrams provided by the Enterprise Ontology, namely the Actor Transaction Diagram or the Process Model. It would go far beyond the scope of this paper to introduce all the Enterprise Ontology models in detail. In the example case given later in this paper, we introduce only the most relevant ones. For further details, we refer to the Enterprise Ontology book (see [13]). Since the initiator could feel an urge to contact the service executor, contact information of the executor needs to be provided in the specification framework. We could consider, for instance, situations in which a protocol error arises after calling an IT service and the fault condition denotes to contact the service executor. Also, the initiator may still have some questions about the service after reading its specification.

The *service production* area of concern focuses on the production act to be performed by the executor. This is the actual value that the service executor offers to the service initiator. It should expose the service properties needed for choosing the right service by a potential service consumer. Unlike service coordination, service production does not concern the communication between the service initiator and the service executor. The aspects which need to be specified are *production act*, *production information used*, *production fact*, *pre- and postconditions*, *production kind*, *production world semantics*. The production act is gained from the Actor Transaction Diagram or the Process Model of the Enterprise Ontology. The information, which needs to be used in order to execute the actual production act, is described in the Information Use Table of the Enterprise Ontology. This table defines for every coordination act and production act which information is required and therefore specifies the required input parameters. The execution of a production act results in a production fact. This is the actual value requested by the initiator. By having specified the production fact, which is gained from the Transaction Result Table of the Enterprise Ontology, the result provided by the service has been defined. The Transaction Result Table defines the result type of each transaction and therefore defines the resulting production fact type for the service. Preconditions and postconditions state production facts that should always hold prior to, respectively after the execution of the service. Information about the pre- and postconditions are gained from the Action Model of the Enterprise Ontology. This model defines the operational business rules of an enterprise. Since we distinguish between different kinds of production acts—ontological, infological and datalogical—the production kind aspect defines the kind of service we are specifying. To have a common knowledge and understanding about the semantics of the service to be provided, the production world semantics need to be specified. This information is modeled in the State Model of the Enterprise Ontology and can, therefore, be used for specifying the production world semantic aspect.

The *service coordination* area of concern has as goal to give the consumer all information required for realizing successful communication with the provider. As we have seen in Section 3, the Ψ-theory states that communication between actors takes place by means of the coordination acts in a transaction. We, therefore, specify the required *coordination acts* for communicating with the executor. Next to this, for completely specifying the service coordination area of concern, we require three aspects that are implementation-dependent. Since the Enterprise Ontology models the essence of an enterprise in a completely implementation-independent way, we cannot gain this information from the Enterprise Ontology models. Though these aspects are also not explicitly mentioned in the Ψ-theory, they logically follow when one thinks about how to access a service. First, a service consumer needs to know whether the service is an IT service or a human service, because IT systems and humans communicate in a different way. We call the related aspect *coordination kind*. Second, the consumer has to apply a certain protocol for successful communication. Knowing the location of a service in itself is pretty useless for a service consumer without knowing how he has to offer the service input to and receive the service output from the service provider. Successful communication between the consuming service component and the providing service component is enabled using *protocols*. Protocols define the rules governing the syntax, semantics and synchronization of communication. Typical examples of protocols for IT services include Internet Protocol (IP), Transmission Control Protocol (TCP), HyperText Transfer Protocol (HTTP) and SOAP. Though often less explicitly defined, human services require protocols too. Take for example of service for ordering food in a three-star restaurant versus in a fast-food restaurant. Not only the quality of the product (the food) and the quality of the service itself (the delivery time, the atmosphere, etc.,) differ, but also the way in which the service consumer (the client) and provider (the waiter) interact. In the three-star restaurant etiquette play a far more prominent role than in the fast-food restaurant. Also, while in the restaurant it is protocol to sit down and wait for the waiter to come to you, you have to go to the counter to order in the fast-food restaurant. Finally, he needs to know the *location* of the service. This location can be either physical or logical. In IT services, logical locations are preferred, e.g., a URL or TCP/IP hostname and port number. By using location addresses, one can easily change the physical server at which the IT services are hosted. For human services, physical locations are more usual. The location of a human service might be, for instance, the second floor room 2.110. Though less common, the location of a human service can be specified as a logical location, e.g., a phone number or an email address. Especially for IT service, the form in which the location is specified is highly dependent on the communication protocol used [32]. Please note that these phone numbers and e-mail addresses play a conceptual different role from those specified in the service provider contact information, though they can have the same value. The location specifies where the service consumer can *access* the service, while the service provider contact information specifies where the service consumer can *get information about* the service.

When entering into commitments, e.g., by providing a service to a service requester, information concerning the quality and the pricing of the provided service need to be discussed between the providing and the requesting party. The results of such negotiations, or predefined pricing and quality aspects, need to be specified in the specification framework as part of the whole contract. In the specification framework such *price quality combination* aspects are specified in the area of concern called *contract options*. These aspects do not directly derive from Ψ-theory, but the negotiation about such aspects can be modeled in separate Enterprise Ontology models. The service contract option area of concern specifies one or several contract options from which service consumers can choose. The contract option aspect consists of a particular *quality* level and the *price* for using the service with this particular quality level. The service executor might define different quality levels in order to anticipate on the various needs and financial positions of different consumers. Example pricing mechanisms are memberships or paying per call. In our framework, we only mention these parts as they should be specified in a different way for human services and IT services. For IT services, one can, for instance, apply the Web Service Level Agreement standard; a standard for unambiguous and clear specification of Service Level Agreements that can be monitored by services consumer as well as provider.

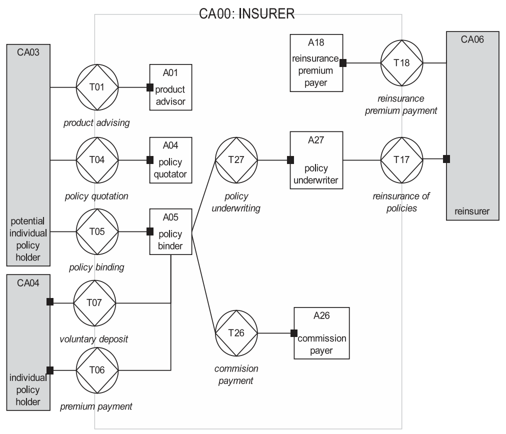


Fig. 4. Actor transaction diagram of protector.

5 Insurance Case

In the next paragraphs, we introduce a life insurance case example in order to demonstrate how to apply the specification framework.

The insurance company Protector [33] offers three types of *life insurance* products: term life insurance, pension insurance, and capital sum insurance. The term life insurance protects the beneficiaries of a policy from the financial damage they suffer in case the insured dies during the policy term. The pension insurance can be seen as an insurance that protects the insured from a large income loss if he reaches his pension age or that protects his life partner and young children from large income loss after the insured (would have) reached his pension age in case of the insured's death. The capital sum insurance is an insurance for building up capital. When the end date of the policy is reached, then the benefit will be paid in a single payment. This product type is, for instance, suitable for saving money to pay off a mortgage. Protector offers multiple products of each type.

These products can be sold by Protector either to a company, i.e., *collectively*, or to an individual person, i.e., *individually*. Some products may be sold both collectively and individually, some only collectively or individually. Most of the pension insurance products are collective insurances. An example of an individual insurance is a term life insurance related to a mortgage.

Fig. 4 shows a part of the Actor Transaction Diagram. For explaining how we can apply the service specification framework, we have selected the subset of the transactions of the life insurer that is of relevance for handling new individual policies. We distinguish between the following four composite actor roles: potential individual policy holder (CA03), individual policy holder (CA04), insurer (CA00), and reinsurer (CA06). In our case, Protector fulfills the role of insurer. The reinsurer insures persons that would cause a high risk for the insurer, for example, because the insured amount is high. This means that a part of the insured amount is insured by the reinsurer in order to spread the risk. A reinsurer can be another "regular" insurance company or an insurance company that is specialized in insuring insurance companies. Sometimes reinsurance is legally obligatory, and sometimes it is a choice made by the insurer itself.

The composite insurer actor role can be further decomposed into the following atomic actor roles: product advisor (A01), policy quotator (A04), policy binder (A05), reinsurance premium payer (A18), policy underwriter (A27), and commission payer (A26). This list is not complete as we only mention the roles that are related to handling new individual policies. The product advisor is responsible for providing a potential individual policy holder of advice of which products suit his needs. This is expressed in the transaction type T01 in Fig. 4. If the potential policy holder is interested in one or more products, he can request a quotation from the policy quotator (T04). After that, the potential policy holder can request policy binding (T05), which makes the policy legally binding. The policy binder request the policy underwriter to check whether or not the risk is acceptable and if an additional premium fee is required (T27). The risk may be so large that the policy underwriter requests reinsurance (T17). Also, the policy binder may request commission payment for an insurance agent (T26). The individual policy holder is responsible for premium payment (T06) and for some types of products (e.g., pension insurance) he may make voluntary deposits (T07). The reinsurance premium payer pays the reinsurance premium to the reinsurer (T18).

In the remainder of this section, we apply the generic service specification framework to specify the service "policy underwriting," which implements the T27 transaction as shown in Fig. 4, as an example.

5.1 Service Executor

As mentioned in Section 4, the service executor aspect specifies the role of the actor that takes final responsibility for the service. So whether it concerns an IT service or a human service, the same actor role remains responsible. The only difference is that in case of an IT service the actor fulfilling the actor role delegates his responsibility to an IT system. As shown in Fig. 4, the policy underwriter is the actor role executing the transaction T27 and, therefore, responsible for the "policy underwriting" service. The specification of the service executor area of concern for Protector looks as follows:

Actor Role
Policy Underwriter

Contact Information
University Street 1A
8291 BN Insurancetown
555-492022
underwriting@protectorinsurances.com

TABLE 1
Extract of the Information Use Table of Protector

Object class, fact type, or result type	Process steps
POLICY	T27/pm, T27/ex
INSURANT	T27/rq, T27/pm, T27/ex
party *par* is the insurant of policy *pol*	T27/pm, T27/ex
minimal_age	T27/rq
INSURED	T27/pm, T27/ex
person *per* is the insured of policy *pol*	T27/pm, T27/ex
INSURANCE PREMIUM	T27/ac
the payer of premium *pre* is the insurant of policy *pol*	T27/ac
INSURANCE BENEFIT	T27/ac
the beneficiary of insurance benefit *ben* is the beneficiary of policy *pol*	T27/ac

5.2 Service Production

In the service production area of concern, we specify the actual value that the policy underwriter actor role (in this case, the policy underwriter) offers to the service initiator (in this case the policy binder actor role). The *production act* in our example concerns policy underwriting, which is part of the transaction "T27 Policy underwriting." We define this act as follows:

Policy underwriting is the act of evaluating the risk and exposures of potential insurants. It involves making the decision whether or not the insurant can get coverage for the insured and what additional premium the insurant has to pay if the insured poses a larger than average risk.

The *information used* aspect is derived from the Information Use Table (IUT). The extract of the IUT concerning the transaction "T27 Policy underwriting" is shown in Table 1. This IUT is in its turn derived from the State Model and the Action Rules, and specifies for every object class, fact type, and result type from the State Model, in which steps of the Process Model its instances are used. The notation of the process steps in the right column are as follows: Transaction/Process step. For example, T27/ex denotes the execution of the production act in the transaction "policy underwriting," and T27/ac denotes the coordination act accept of the transaction "policy underwriting."

As shown, we need information about the insurant in the request process step. Also, we need to know the minimal age for someone to act as an insurant. During the promise step as well as during the production step, the executor needs to get information about the policy, the insured, and the insurant. These data are required to enable him to make the underwriting decision. The service initiator, namely the policy binder, requires information on the insurance premium and insurance benefit to allow him to decide whether or not to accept the production fact.

The *production fact* can be derived from the Transaction Result Table. As shown in Fig. 5 the result type of the transaction "policy underwriting" is "policy underwriting for policy *pol* has been done." The *production kind* of this transaction is "ontological," since it forms part of the ontological model of Protector. Examples of an infological respectively datalogical service are "calculate premium" and "store calculation results."

We model the *production world semantics* in the State Model, which uses an Object Role Modeling (ORM)-based notation technique called World Ontology Specification

Transaction	**Result type**
T01 Product advising	product advice *adv* is created
T04 Policy quotation	policy *pol* is quoted
T05 Policy binding	policy *pol* is bound
T06 Premium payment	premium is paid for policy *pol* for premium period *per*
T07 Voluntary deposit	voluntary deposit is made for policy *pol*
T17 Reinsurance of policies	policy collection *pco* is reinsured for period *per*
T18 Reinsurance premium payment	reinsurance premium is paid for policy collection *pco* for period *per*
T26 Commission payment	commission *com* is paid
T27 Policy underwriting	underwriting for policy *pol* has been done

Fig. 5. Transaction result table of protector.

Language (WOSL) [34]. Fig. 6 exhibits the complete State Model of Protector. In our service example, we only use parts of this model, so we only explain the most important concepts. We use the term *policy* for the individual policy as well as for a participation in a collective contract. An insurance policy has an insurant, one or more insured, and one or more beneficiaries. The insurant is an organization or person that is responsible for the payment of the premium of a policy. The insurant is the client of Protector. The insured is a person who is the "insured object." The beneficiary is a person who receives a payment if the insurant has a right to a benefit according to the product rules of a policy.

In Section 4, we stated that *pre- and postconditions* are gained from the Action Model (see Fig. 7). The action rules in this model are written down in a pseudoalgorithmic language. The following action rule, for example, specifies how the actor decides whether or not he promises to underwrite the policy.

As becomes clear from this example, the policy underwriter checks whether the insurant is an individual or a company. For companies, he always promises to underwrite the policy; for persons, only if the person is older than the minimal age (which in this case is 18). So one of the preconditions of the service states that if a person fulfills the

```
on requested T27(insurant)

    if < type(insurant)=person AND age(insurant)< minimal_age > →
        decline T27(insurant)
    ◇ not < type(insurant)=person AND age(insurant)< minimal_age > →
        promise T27(insurant)

no
```

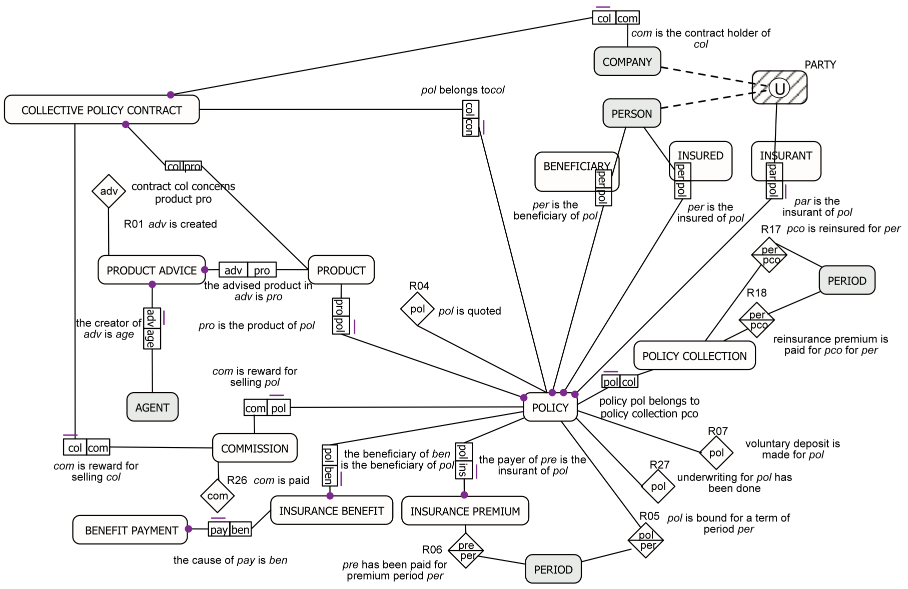
Fig. 7. Part of the action model of protector.

role of insurant, he should have a minimum age of 18. Likewise, an example of a postcondition is that the insurance premium always is larger than 0.

5.3 Service Coordination

With the *coordination acts* aspect, the steps of a transaction that deal with communication between the initiator and the executor need to be defined. In our example, if the initiator calls the service "policy underwriting," he wants to know whether the executor processes his request or if the executor may also decline such a request. If the initiator does not receive any notification, such as a promise or a decline, after having sent his request, he would be unsure if his request is being processed or not. For the specification of the coordination aspect, we therefore use a transaction pattern, which needs to be known and agreed upon by both parties, the initiator and the executor. In Section 3.1.2, three patterns have been introduced for describing the coordination between two parties, namely the basic transaction pattern, the standard transaction pattern and the complete transaction pattern. They differ in the way they handle dissents between the two parties. The most used pattern is the standard transaction pattern, which we introduced in Fig. 1, and which we also used for the specification of the coordination acts in the Protector case.

As explained in Section 4, the *coordination kind* states whether we are specifying a human or an IT service. Because



col com
com is the contract holder of col
COMPANY
PARTY
U
COLLECTIVE POLICY CONTRACT
pol belongs tocol
col con
PERSON
BENEFICIARY
INSURED
INSURANT
per pol
per pol
par pol
per is the beneficiary of pol
per is the insured of pol
par is the insurant of pol
col pro
adv
contract col concerns product pro
R01 adv is created
R17 pco is reinsured for per
per pco
PERIOD
PRODUCT ADVICE
adv pro
PRODUCT
the advised product in adv is pro
R04
pol
pol is quoted
R18
per pco
reinsurance premium is paid for pco for per
adv age
the creator of adv is age
pro is the product of pol
pro pol
POLICY COLLECTION
com is reward for selling pol
pol col
AGENT
com pol
POLICY
policy pol belongs to policy collection pco
R07
pol
voluntary deposit is made for pol
col com
COMMISSION
pol ben
the beneficiary of ben is the beneficiary of pol
pol ins
the payer of pre is the insurant of pol
com is reward for selling col
R26 com is paid
com
R27
pol
underwriting for pol has been done
INSURANCE BENEFIT
INSURANCE PREMIUM
R05 pol is bound for a term of period per
pol per
BENEFIT PAYMENT
pay ben
R06
pre per
the cause of pay is ben
pre has been paid for premium period per
PERIOD

Fig. 6. State model of protector.

protocols and location are implementation-dependent aspects, it makes a large difference for their specification which type of service we are dealing with.

Let us assume "policy underwriting" is a human service. The *protocol* should in that case prescribe how to interact with the human being fulfilling the role of service executor. We could imagine, for instance, the following procedure:

1. Send quotation to the policy underwriter by post.
2. Receive confirmation from policy underwriter by post.
3. Discuss additional questions by telephone with policy underwriter.
4. Receive final underwriting decision by post.

When we look at the protocol aspect, we see that we need to specify two types of *locations* for accessing the service: the postal address and the phone number of the policy underwriter. Keep in mind that though these data could have an overlap with the service executor data, they serve a different purpose. The location aspects specifies data required for the operational use of the service, while the contact details in the service executor area of concern specifies data required for the service initiator to contact the service executor *about* the service. Therefore, these data can very well differ from each other.

Now, let us think of "policy underwriting" as an IT service. Example *protocols* for getting data across a network include HTTP or a queuing protocol if guaranteed delivery is required. We can apply the WSDL standard for structuring the messages exchanged. The *location* can be specified as an URL, e.g., 165.34.2.113.

5.4 Service Contract Options

In the service contract options area of concern, we specify which quality of service the service initiator gets for which price. Let us assume Protector uses an internal costing calculation system as most large organizations do. Protector distinguishes between two types of quality levels: "regular" and "urgent." The regular service call has a price of EUR 8 and the urgent service call of EUR 12. For both IT services and human services we could specify the accompanying quality aspect as follows:

regular: the maximum response time in 90 percent of the calls is 5 hours

urgent: the maximum response time in 90 percent of the calls is 2 hours

Usually, IT services tend to have quicker response times than human service, though this does not have to be the case. Especially asynchronous services can also take multiple days, weeks, or even month to finish.

Besides response time, we could also take into account other quality aspects like security aspects, accuracy of data, availability, etc.

6 Validation of the Generic Service Specification Aspects

In Section 5, we showed the applicability of the service specification framework by means of a real case example. In order to validate the specification aspects proposed in this paper, we have conducted an additional case study. Cases or case studies are often not seen as an adequate research methodology for the verification of research results. However, Flyvbjerg [35] states that the problems with the conventional wisdom about case study research can be summarized in five misunderstandings or oversimplifications about the nature of such research:

- General, theoretical (context-independent) knowledge is more valuable than concrete, practical (context-dependent) knowledge.
- One cannot generalize on the basis of an individual case; therefore, the case study cannot contribute to scientific development.
- The case study is most useful for generating hypotheses; that is, in the first stage of a total research process, whereas other methods are more suitable for hypotheses testing and theory building.
- The case study contains a bias toward verification, that is, a tendency to confirm the researcher's preconceived notions.
- It is often difficult to summarize and develop general propositions and theories on the basis of specific case studies.

In his article "Five Misunderstandings About Case-Study Research," he corrects each of this statements and summarizes that a case study is a necessary and sufficient method for certain important research tasks. We agree among others with the following statement: 1) "One can often generalize on the basis of a single case, and the case study may be central to scientific development via generalization as supplement or alternative to other methods. But formal generalization is overvalued as a source of scientific development, whereas "the force of example" is underestimated." and 2) "The case study contains no greater bias toward verification of the researcher's preconceived notions than other methods of inquiry. On the contrary, experience indicates that the case study contains a greater bias toward falsification of preconceived notions than toward verification."

We, therefore, used the insurance case to demonstrate the applicability of our generic service specification framework, and the case study of a global provider (called GP in the paper) of leasing, business and consumer finance solutions, including vendor finance and factoring, to validate the specification aspects proposed in this paper. A short summary of the GP case study is given next.

The main goal of GP is to help its customers grow market share, enhance profitability and achieve strategic goals by offering them asset-based financing programs. GP is a fully owned subsidiary of a big bank. The company operates internationally and in 2008 GP reported a net profit of 235 million EUR. GP has been working on SOA for a couple of years and is currently further introducing SOA governance within the organization. At the moment of the case study, GP as they were planning to migrate their own service specifications, stored at the moment in various different Word templates, into a service registry/repository.

Among others, several interviews have been executed. Tables 2, 3, 4, and 5 depict the validation comments of the interviewees regarding the specification aspects.

The participants in this case study gave positive feedback about the service specification framework, since most of the

TABLE 2
Evaluation of the Service Executor Aspects

Service Executor Aspects	Comments of interviewees
Actor Role	GP does not specify the actor role, but only the actual owner. A distinction is made between a functional and technical owner. In practice, it is not always clear who the owner is or should be. The functional owner is among others responsible for getting functional requirements from users, setting the price and the funding of the service. The technical owner is, for instance, responsible for the middleware.
Contact Information	End users only contact the service desk for problems they encountered. The service desk routes problems to the right service owner. So only the service desk would need this information for immediate problems. For inquiry purposes this information would also be required by developers, architects etc. Contact information of every employee can be found in the business directory, so for internal owners this would not need to be specified separately in a service registry. For external owners the information is explicitly written down.

information they required was specified in the presented framework. Fig. 8 shows the average rating and standard deviation per service specification aspect. As can be seen there was little variation in the answers of the interviewees. Most of the specification aspects got a rating of 7.4 or higher within a scale from 1 to 10, where 1 is the lowest and 10 the best rating. The ones that scored lower were “production kind” (6.4), “coordination kind” (6.6), and “price quality combination” (6.6). An interesting finding emerged from the interviews and questionnaires. We presumed that software systems are mainly production-centric. However, GP did value the aspect of coordination acts, especially for services that reach beyond the boundaries of the organization itself, very high. Currently, they have a lot of manual work in order to correct the state of information systems after a cancellation or to ask for the status of a process because promise acts are not made explicit and therefore can be interpreted in different ways. Nevertheless, GP hardly specify or implement these coordination acts (e.g., promise or state messages) in services so far because they do not have a clear vision on how to do this. One thing that was mentioned as lacking in the framework was information about the life cycle (e.g., “in production” or “in test”) and versioning (e.g., “version 2.3”) of the service.

7 Conclusion

Service-oriented approaches are gaining more and more attention since they claim to provide new and flexible ways of supporting the activities in an organization. However, the current ways of implementing these approaches often lead to additional overhead and additional costs without delivering the expected advantages. Two major problems in the area of service-orientation can be identified. First, a complete and clear understanding of the notion of service is missing. As a consequence, it is unclear what functionality should be implemented as a service. Second, an appropriate framework for specifying services is missing. As a consequence, insufficient information is provided to the service consumer concerning various aspects, e.g., the functionality and the behavior of the respective service. In this paper, we primarily addressed the problem of service

TABLE 3
Evaluation of the Service Production Aspects

Service Production Aspects	Comments of interviewees
Production Act	This is sometimes, but not always specified. It would be useful to have this for having a quick overview of what the service does in search screens in the service repository. At GP, a consumer often needs to read the complete specification to get this overview.
Production Information Used	The input of a service is specified (including complex constraints) in functional designs. This is sufficient according to the interviewees.
Production Fact	The output of a service is specified (including complex constraints) in functional designs. This seems to be sufficient. However, some of the participants would like to see a more formal and consistent way of specifying this information.
Production Kind	This is not specified as the organization does not use the notion of Enterprise Ontology nor does the organization propose any other service taxonomy. The number of services is not large enough at the moment to need such a taxonomy. The participants do see a need for it (for searching purposes) when the number of services grows.
Production World Semantics	The business departments at GP use Excelsheets for specifying semantics. The interviewees would prefer to not only specify the definitions of objects, but also their relationship. However, the more important and immediate problem is that business people have difficulties in making precise and unambiguous definitions.
Preconditions	Preconditions are written down informally and in different locations of specification documents. This seems to be sufficient to most of the interviewees. However, some of the participants would like to see a more formal and consistent way of specifying this information.
Postconditions	Postconditions are written down informally and in different locations of specification documents. This seems to be sufficient to most of the interviewees. However, some of the participants would like to see a more formal and consistent way of specifying this information.

TABLE 4
Evaluation of Service Coordination Aspects

Service Coordination Aspects	Comments of interviewees
Coordination Acts	This is not specified, but according to the interviewees it would be useful to specify it.
Protocol	Protocols are usually not specified explicitly as they are almost always SOAP over HTTP for internal communication and SOAP over MQ for external communication.
Location	The location is specified on an intranet website. Here all servers in the service development phase can be found (development, test, acceptance, and production).

TABLE 5
Evaluation of Contract Option Aspects

Contract Options	Comments of interviewees
Price Quality Combination	For external services the price is usually documented. Internally it is not always a wish to specify prices for commercial and political reasons. Quality aspects are specified very basically. This is not always sufficient; sometimes consumer and providers get different expectations in this area due to lack of specification.

specification. However, as we have based our research on the rigorous Ψ-theory, we have also improved our understanding of services by relating them to transactions. The function of the specification of a service is to give all stakeholders the information about the service they need, e.g., for service discovering, selection, and usage. Solely by specifying the input and output aspects of a service, as is the current practice, the service consumer does not get sufficient information to determine whether the service fits his needs. These specification aspects only reflect part of the total externally visible behavior of a service. Though many standards exist for specifying certain aspects of a service, a holistic approach is still missing. The main contribution of this paper is the development of such a holistic framework, which we call the *Generic Service Specification Framework*. In order to do so, we provided a definition of the notion of service and introduced six different types of services, based on the Ψ-theory. The first distinction is between human services, i.e., services executed by human beings, and IT services, i.e., services executed by IT systems. The second distinction corresponds to the three aspect organizations, as proposed by the organization theorem: the B-organization, the I-organization, and the D-organization. By applying the Ψ-theory to an organization, one can extract the essence of the construction and the operation of (the organization of) an enterprise. This essence is contained in the so-called ontological model of the organization. This essential knowledge is needed for identifying the services that are needed in order to let the organization operate. Next, these services are specified according to the generic service specification framework. As an example for demonstrating the feasibility and the usefulness of the generic service specification framework, we analyzed the insurance case. For this case, we provided two ontological partial models and discussed how these models can be used to specify all aspects of a service, according to the service specification framework.

In future work, we have two main goals: 1) to validate our framework in more real-life case studies, especially at enterprises that have a large amount of services, and 2) to map existing standards to the different aspects of our framework. Currently, we have validated our framework in three case studies: the insurance and the bank case studies, discussed in this paper, and a large maritime organization. At the moment, we are further refining our framework. We have plans to also evaluate the framework additionally at an aviation company. After these steps, we intend to use our framework as a basis for selecting the most suitable standards for service specification. It may be the case that some aspects are not covered by existing standards and that we need new ones, but this is still to be investigated.

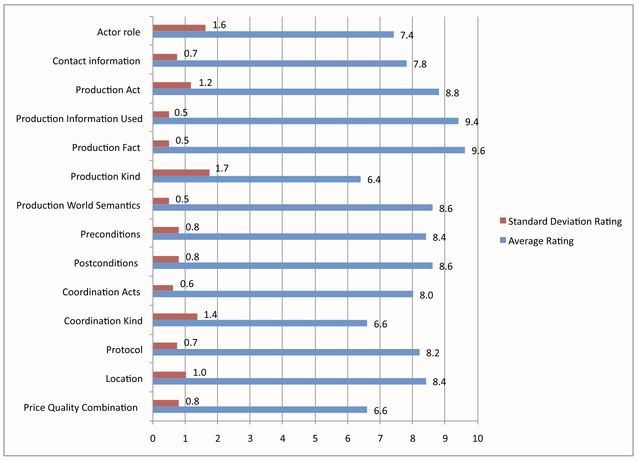


Fig. 8. Rating of service specification aspects.

Acknowledgments

First, the authors would like to thank Jan L.G. Dietz, the founder of Enterprise Ontology, for all the valuable discussions and comments about their work. Following his suggestions and applying his principle of modeling the essence of an enterprise, they were able to develop a generic but essential framework for service specification. Additionally, they would like to thank Gde Hardjosumarto, who did a lot of preparatory work in his master thesis, which could be used to derive their specification framework as presented in this paper. This project was supported by the Swiss National Science Foundation (SNSF) and Agentschap NL.

References

[1] W3C, "Web Services Description Language," http://www.w3.org/TR/wsdl12/, 2012.
[2] T. Bellwood et al., "UDDI Spec Technical Committee Draft," Oasis, technical report, http://uddi.org/pubs/uddi_v3.htm, 2004.
[3] J.L. Dietz, "Enterprise Ontology—Understanding the Essence of Organizational Operation," *Enterprise Information Systems*, vol. 7, pp. 19-30, 2006.
[4] W3C, "Web Services Policy 1.5—Framework," http://www.w3.org/TR/ws-policy/, 2007.
[5] A. Keller and H. Ludwig, "The WSLA Framework," *J. Network and Systems Management*, vol. 11, no. 1, pp. 57-81, 2003.
[6] Open Grid Forum, "Web Services Agreement Specification," http://www.ogf.org/documents/GFD.107.pdf/, 2007.
[7] J. Ackermann et al., *Standardized Specification of Business Components,* Memorandum of the Working Group 5.10.3, 2002.
[8] J. Walkerdine, J. Hutchinson, P. Sawyer, G. Dobson, and V. Onditi, "A Faceted Approach to Service Specification," *Proc. Second Int'l Conf. Internet and Web Applications and Services (ICIW),* p. 20, 2007.
[9] F. Colasuonno et al., "jUDDI+: A Semantic Web Services Registry Enabling Semantic Discovery and Composition," *Proc. IEEE Eighth Int'l Conf. E-Commerce Technology and IEEE Third Int'l Conf. Enterprise Computing, E-Commerce, and E-Services (CEC/EEE '06),* 2006.
[10] J. Luo et al., "Adding OWL-S Support to the Existing UDDI Infrastructure," *Proc. IEEE Int'l Conf. Web Services (ICWS '06),* pp. 153-162, 2006.
[11] WSMO, "D10 v0.1 WSMO Registry," http://www.wsmo.org/2004/d10/v0.1/, June 2007.
[12] P. Rajasekaran, J.A. Miller, K. Verma, and A.P. Sheth, "Enhancing Web Services Description and Discovery to Facilitate Composition," *Proc. First Int'l Conf. Semantic Web Services and Web Process Composition (SWSWPC),* pp. 55-68, 2004.
[13] J.L.G. Dietz, *Enterprise Ontology: Theory and Methodology.* Springer, 2006.

[14] F. Flores and J. Ludlow, "Doing and Speaking in the Office," *Decision Support Systems, Issues and Challenges,* pp. 95-118, 1980.
[15] G. Goldkuhl and K. Lyytinen, "A Language Action View of Information Systems," *Proc. Int'l Conf. Information Systems (ICIS '82),* pp. 13-29, 1982.
[16] M.A. Bunge, *Treatise on Basic Philosophy: A World of Systems,* vol. 4. D. Reidel, 1979.
[17] H. Weigand, "The Language/Action Perspective," *Data and Knowledge Eng.,* vol. 47, no. 3, pp. 299-300, 2003.
[18] J.L.G. Dietz and J.A.P. Hoogervorst, "Enterprise Ontology in Enterprise Engineering," *Proc. ACM Symp. Applied Computing (SAC),* pp. 572-579, 2008.
[19] J.L.G. Dietz and A. Albani, "Basic Notions Regarding Business Processes and Supporting Information Systems," *Requirements Eng.,* vol. 10, no. 3, pp. 175-183, 2005.
[20] J.L. Dietz, "The Deep Structure of Business Processes," *Comm. ACM,* vol. 49, no. 5, pp. 58-64, 2006.
[21] J.L.G. Dietz and J. Hoogervorst, "Enterprise Ontology and Enterprise Architecture, How to Let Them Evolve into Effective Complementary Notions," *GEAO J. Enterprise Architecture,* vol. 2, no. 1, 2007.
[22] J. Gadrey, "The Characterization of Goods and Services: An Alternative Approach," *Rev. Income and Wealth,* vol. 46, no. 3, pp. 369-387, Sept. 2000.
[23] V.A. Zeithaml, L.L. Berry, and A. Parasuraman, "The Nature and Determinants of Customer Expectations of Service," *J. Academy of Marketing Science,* vol. 21, no. 1, pp. 1-12, Jan. 1993.
[24] F. Gallouj and O. Weinstein, "Innovation in Services," *Research Policy,* vol. 26, no. 4/5, pp. 537-556, 1997.
[25] C.W. Hart, "The Power of Unconditional Service Guarantees," *Harvard Business Rev.,* vol. 66, no. 4, pp. 54-62, 1988.
[26] S.M. Goldstein, R. Johnston, J. Duffy, and J. Raod, "The Service Concept: The Missing Link in Service Design Research?" *J. Operations Management,* vol. 20, no. 2, pp. 121-134, 2002.
[27] OMG SOA SIG, http://soa.omg.org/, 2006.
[28] OASIS, http://www.oasis-open.org/, 2006.
[29] The Open Group, "SOA," http://www.opengroup.org/projects/soa/, 2006.
[30] W3C, http://www.w3c.org/, 2006.
[31] A. Albani, G. Hardjosumarto, L. Terlouw, and J.L.G. Dietz, "Service Definition Based on the ψ-Theory," technical report, Delft Univ. of Technology, 2009.
[32] M. Stal, "Using Architectural Patterns and Blueprints for Service-Oriented Architecture," *IEEE Software,* vol. 23, no. 2, pp. 54-61, Mar./Apr. 2006.
[33] L. Terlouw, "Towards a Business-Oriented Specification for Services," *Advances in Enterprise Engineering I,* Springer, 2008.
[34] J.L.G. Dietz, "A World Ontology Specification Language," *Proc. OTM Workshops,* pp. 688-699, 2005,
[35] B. Flyvbjerg, "Five Misunderstandings about Case-Study Research," *Qualitative Inquiry,* vol. 12, no. 2, pp. 219-245, Apr. 2006.

Linda I. Terlouw received the MSc degree in computer science and the MSc degree in business information technology from the University of Twente. She has a PhD degree in computer science from the Delft University of Technology and currently works as an enterprise architect with ICRIS B.V. (www.icris.nl). Before starting Icris, she worked for several large companies like IBM and Ordina. Her research is part of the CIAO! Program.

Antonia Albani received the master and PhD degrees in computer science. She is a senior researcher at the University of St. Gallen, Switzerland. Previously, she was an assistant professor at the Delft University of Technology, The Netherlands. She is a cofounder of the research network CIAO! (Cooperation & Interoperability—Architecture & Ontology, www.ciaonetwork.org) and member of the CIAO! executive board. She worked in IT consulting and was CEO and cofounder of an Internet start-up in the area of business process outsourcing. Her main research interest is on service innovation and service engineering.