

Modelling Properties of Services

Jens Hündling

Hasso-Plattner-Institute for IT-Systems-Engineering at the University of Potsdam
Prof.-Dr.-Helmert-Strasse 2-3, 14482 Potsdam, Germany
jens.huendling@hpi.uni-potsdam.de

Abstract. Information technology is the core enabler for global businesses and integrated, virtual enterprises that achieve common business goals. This integration can be supported or even automated and service-oriented computing offers a promising approach to achieve these goals. This paper argues that a key aspect is publishing standardized services descriptions in a formal, computer-readable way. Furthermore, the paper introduces a model for enriched service description allowing to characterize services modelling their properties. This model will be related to a general service model and to Service Level Agreements.

1 Introduction

Companies are coalescing, building virtual enterprises and are achieving common business goals together [4]. Simultaneously, companies try to stem dependencies and want to stay flexible and agile. Information technology is the core enabler for tightening integration of companies, and Service Oriented Computing (SOC) is the arising paradigm that is also able to handle flexibility [1, 9]. Even more, SOC increases automation based on publishing standardized service descriptions. SOC undoubtedly offers a promising approach to achieve this goal, but certain aspects are still in their infancies. One key aspect is addressed in this paper by introducing a model for properties of services, which enables enriched service description that exceed technical interface specifications.

The basic principles of SOC are manifested by the Service Oriented Architecture (SOA) [3, 9], defining the roles and interactions of *Service Providers*, *Service Repositories* and *Service Consumers*. Essential requirements are standardized description languages and standard communication protocols as well as a common understanding of the querying, the categories and other objectives of the service repository. De-facto standards for today's most popular implementation of SOC are existing in the form of Web services [1, 9]. Throughout the paper the terminology of the Web Services Description Language (WSDL) 2.0 [6] will be used. Additionally, for specific industrial sectors, standardized names for categories and services are evolving or are already existing. The semantic web [8] is an emerging technology for modelling the meaning of services with approaches like WSMO [7] and OWL-S [17].

The paper is organized as follows: Section 2 introduces the main terms related to service descriptions and presents an example. The services properties model is explained in Section 3 and Sections 4 gives a brief summary and outlook.

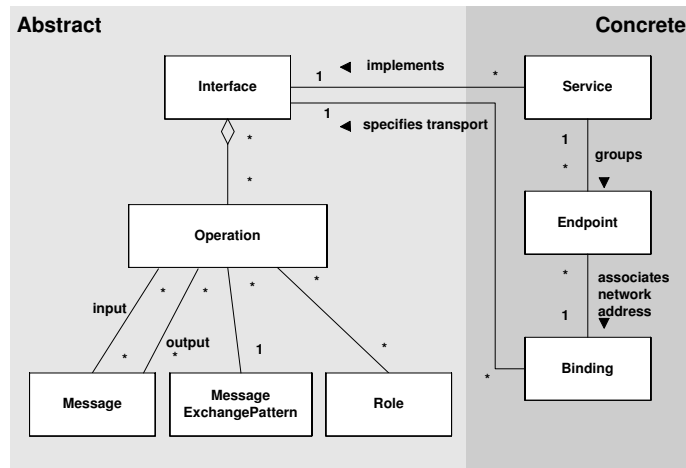


Fig. 1. Core Elements of a Service Description.

2 Service Description

A model for service description is given in as an UML 2.0 class diagram in Fig. 1. This model separates the description of the abstract interface from concrete details of a service description such as 'how' and 'where' it is offered. The abstract level describes a service in terms of the communication, i.e. the *messages* it sends and receives. One or more messages, their exchange pattern and the logical roles of the participants are aggregated in an *operation*, which in turn are grouped in an *interface*. The concrete level specifies transport details for interfaces in a *binding*, adds network address to form an *endpoint*, and groups endpoints implementing a common interface as a *service* [6]. These two levels of abstraction will also be relevant for the properties of services model introduced in this paper.

2.1 Example

For better understanding and to illustrate the requirements, a short example of transportation services will be introduced. These services are invoked by a message containing procurement information, destination address, etc. Additional information is available, e.g. temperature in climatic containers, cost, duration, security options. Table 1 shows the service implementations S_{11} to S_{32} with all properties. These diverse properties show the complex requirements for a service property model. All services are implementing a common transportation interface I_T , which is not described in more detail here.

2.2 Non-functional Properties vs. Quality of Services

Properties like those in the example are often subsumed either under *Quality of Services*(QoS) [5, 11] or *non-functional properties* [14, 18]. Since these terms are often

Service	Climatic		Cost	Duration	Security Option	State Monitor
	Temp.	Humidity				
S_{11}	–	–	100-150 €	12-48 hours	–	web portal
S_{12}	$\leq -8^{\circ}\text{C}$	20 – 60%	275 €	24 hours	–	web portal
S_{21}	–	–	£150	1 day	–	mail, phone
S_{22}	max. 20°F	–	£320	8 hours	128-bit key	mail, phone
S_{31}	$\leq +4^{\circ}\text{C}$	20 – 80%	400 €	24-36 hours	512-bit key	mail, web portal
S_{32}	$\leq -4^{\circ}\text{C}$	20 – 40%	450 €	48 hours	512-bit key	web Portal

Table 1. Sample Transportation Services and their Properties

used with controversial meanings, they will be discussed here shortly. *Functional properties* describe what the service does and *non-functional properties* are used to describe how the service does it. Thus, functional equivalence can be defined on two services, i.e. a service can easily be replaced by a functional equivalent service with more suitable non-functional properties, e.g. lower cost, faster execution, and higher security. This goes along with the basic SOC principle of loose coupling. Nevertheless, in a service-oriented environment it is often hard to decide whether a certain property is functional or non-functional. Consider the property duration (typically called non-functional) in a request like: *Transportation of goods within in 24 hours*. A consumer uses some technique out of the scope of this paper to find services (or interfaces respectively) with this capability [14]. Thus, the consumer gets a (set of) adequate interface(s); considering the example in Section 2.1, this would be the interface I_T . The request R_1 is specified more precisely by stating *within 24 hours*, which is also a *functional* requirement, because a functionally equivalent service has to match this mandatory property, as mentioned above. In the example, S_{31} will not match this criterion and for S_{11} it is not sure. Similar examples can be found for other properties like costs or security, which are often regarded as non-functional properties. To sum up, if a property is functional or depends not on the property itself, but on the request.

Quality of Services is also a widely used term. In [18] QoS constraints are defined as a synonym for non-functional constraints, whereas [13] states QoS as a subset of properties of Web services. Furthermore, in [2] properties are separated in functional, non-functional and quality aspects. There are also authors claiming that QoS implies aspects, which can be both, functional and non-functional [16]. The non-functional properties as defined in WSMO [7] include network-related QoS and adds meta-data about the service description like author, version etc. A model for properties of service is necessary and this paper introduces an approach for their classification. The term property will be used in this paper for a single criterion and Quality of Service specifies the set of all properties.

3 Properties of Services Model

As mentioned above, before using a service, it has to be found. The consumer has to decide, which specific service implementation to use. The service description in Fig. 1

concentrates on technicalities, but several properties dealing with different aspects are relevant for several usage scenarios like discovery, negotiation, composition, substitution and management. Each of these usage scenarios has its own requirements for properties of services; a detailed discussion on these scenarios can be found in [10].

3.1 Scales of Measurement

A first aspect to categorize the properties are the scales of measurement. As known from mathematics, Scales of measurement refer to ways in which variables or numbers are defined and categorized. The scales are *nominal* (e.g. state monitor), *ordinal* (e.g. security), *interval* (e.g. temperature) and *ratio* (e.g. cost). It has to be mentioned that each scale has its allowed arithmetics functionality. This has consequences for specifying the discovery query, e.g. for restrictions on the service's properties, like budget restrictions, temporal deadlines and necessary security protocols. Additionally, a provider has to take the scales into account when offering its service not with precise properties, but with ranges or at different levels; see the example in Table 1. If a consumer searches for a service, the restrictions and aspects like optimization criteria should be described in a declarative way. Methods well known from Description Logic could be applied for matchmaking, like described in [12]. Standardization efforts like the Web Ontology Language (OWL) [15] are suspected to build a semantic foundation for describing such restrictions.

This leads to the following model: The set of QoS for the service interface I is defined as a set of properties P_i , i.e. $Q_I = \{P_1, \dots, P_n\}$. Next to the above mentioned scale $S := scale(P_i)$, each P_i in has a domain $D := dom(P_i)$ which is a set of possible values like \mathbb{Q} for the property costs. Additionally, for each property a metric $M := metric(P_i)$ has to be defined, i.e. how, where, and when the property is measured and the numbers or labels assigned to the measured results. Thus, a property is a tuple containing domain, scale and metric: $P_i := (dom(P_i), scale(P_i), metric(P_i))$, $1 \leq i \leq n$, with $scale(P_i) \in \{nominal, ordinal, interval, ratio\}$. Let $dom(Q_I) = \bigcup_{P \in Q_I} dom(P)$, $scale(Q_I) = \bigcup_{P \in Q_I} scale(P)$ and $metric(Q_I) = \bigcup_{P \in Q_I} metric(P)$. Then, a concrete set of properties L_S of a service S_I implementing an interface I is a transformation $\lambda : Q_I \rightarrow (dom(Q_I), scale(Q_I), metric(Q_I))$, where $\forall P \in Q_I | \lambda(P) \in (dom(P), scale(P), metric(P))$.

Considering the example in Section 1, the set of QoS properties is {Temperature, Humidity, Cost, Duration, SecurityOption, StateMonitor }. The property cost for instance is defined as follows: $dom(cost) = \mathbb{Q}$, $scale(cost) = ratio$ and $metric(cost) = currency$. A concrete service implementation would then be the concrete values resulting from the transformation λ , i.e. the entries in Table 1.

Thus, a provider is able to define a service level of the applicable properties for each service implementation. This service level should be stored as part of the service description in the repository. This enables consumers to define queries with desired mandatory and optional properties. Thereby, for each property it is possible to check values and applicable comparisons like *at least*, *in between* or *max*. For example, a consumer can query for a service with the properties $P_{Duration} \leq 30$ hours, $P_{SecurityOption} \in \{(128\text{-bit key}), (512\text{-bit key})\}$ and so on.

3.2 Properties of Services Model

The model sketched in this paper can be mapped to the service entities as shown in Fig. 2. The *Quality of Services* is a set of properties. For each *Property*, the *Domain*, a *Scale* and a *Metric* has to be defined. This abstract set of QoS properties is bound to an *Interface*, specifying each interface's set of applicable properties. The concrete counterpart is a *ServiceLevel* which in turn is related to a service. When a *Provider* stores its service and the service level in a repository, ranges etc. can be defined. A *Provider* and a *Consumer* can negotiate on more precise properties and create a *ServiceLevelAgreement* as a contract.

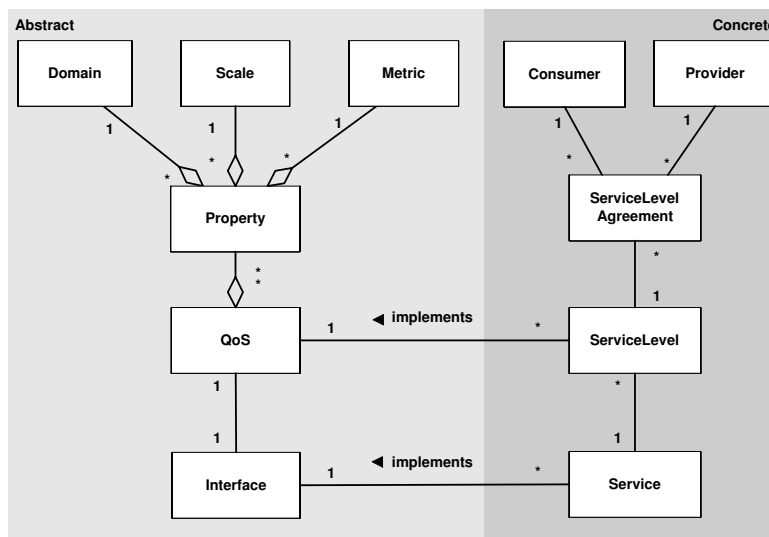


Fig. 2. Mapping Service Properties to Service Entities.

4 Conclusion

This paper sketches a services properties model. It is based on the principles of domains and scales of properties. These properties are grouped as a set that is related to an interface. For a concrete service implementation, the specific service level can be defined and enhanced requests for services can be generated. With service description including properties as presented in this approach, it is possible to enhance several use cases like service discovery or service composition. Service processes can be analysed and execution duration etc. can be estimated beforehand by using information in the enhanced service descriptions. Even more, the information can be used at run time to detect possible deadline violations in advance.

Nevertheless, many things that can be found in the example are not described in detail in this paper. Considering the sample service S_{11} , it is likely that the cost depends on the duration negotiated as well as the distance between starting and delivery location. This model is also not complete regarding the aspects of a property, e.g. service S_{31} : this service is offered by the provider with restrictions and ranges. Additionally, the metric has been left out of the discussion as well as the difference of the measurement units like £ and € or °C and °F. These aspects are future work, and defining properties in OWL [15] is currently under investigation. Additionally, using this approach for service composition and process (re-)planning is still under research.

References

1. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer Verlag, 2004.
2. S. Andreozzi, D. Montesi, and R. Moretti. Web services quality. In *Proc of Int. Conference on Computer, Communication and Control Technologies (CCCT03)*, 2003.
3. S. Burbeck. The Tao of e-business services. IBM Corporation, October 2000.
4. C. Bussler. *B2B Integration - Concepts and Architecture*. Springer, June 2003.
5. J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Elsevier Journal of Web Semantics*, 2004.
6. R. Chinnici, M. Gudgin, J.-J. Moreau, J. Schlimmer, and S. Weerawarana. Web Services Description Language (WSDL) version 2.0, August 2004.
7. J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Kifer, J. Kopecky, R. Lara, E. Oren, A. Polleres, and M. Stollberg. Web Service Modeling Ontology (WSMO). <http://www.wsmo.org/TR/d2/v1.1/>, Feb. 2005.
8. D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, Berlin, 2 edition, 2003.
9. J. Hündling and M. Weske. Web services: Foundation and composition. *EM - Electronic Markets Journal*, 13(2):108–119, June 2003.
10. J. Hündling and M. Weske. Modeling quality of services in service oriented environments. In *Proc. of the 1st Intern. Symposium on Leveraging Applications of Formal Methods (ISoLA 2004)*, October 2004.
11. M. C. Jaeger, G. Rojec-Goldmann, and G. Mühl. QoS aggregation for service composition using workflow patterns. In *Proc. EDOC 2004*, pages 149–159, September 2004.
12. L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proc. of WWW'2003*, pages 331–339. ACM, 2003.
13. A. Mani and A. Nagarajan. Understanding quality of service for web services. IBM Corporation, Jan. 2002.
14. J. O'Sullivan, D. Edmond, and A. Ter Hofstede. What's in a service? towards accurate description of non-functional service properties. *Distributed and Parallel Databases*, 12:117–133, 2002.
15. M. K. Smith, C. Welty, and D. L. McGuinness. OWL Web Ontology Language Guide. Available at <http://www.w3.org/2004/OWL/>, 2004.
16. R. Sumra and Arulazi D. Quality of service for web services - demystification, limitations, and best practices. <http://www.developer.com/>, March 2003.
17. The OWL Services Coalition. OWL-S: Semantic markup for web services. Available at <http://www.daml.org/services>, 2004.
18. V. Tasic, B. Pagurek, K. Patel, B. Esfandiari, and W. Ma. Management applications of the web service offerings language (WSOL). In *Proc. of CAiSE'03*, pages 468–484. Springer LNCS, 2003.