8-5-2011

The State of the Art of Service Description Languages

Sebastian Schlauderer
University of Augsburg, sebastian.schlauderer@uni-bamberg.de

The State of the Art of Service Description Languages

Sebastian Schlauderer
University of Augsburg
sebastian.schlauderer@wiwi.uni-augsburg.de

ABSTRACT

The service-oriented computing paradigm has gained more and more attention in research and practice over the last couple of years. Today, the idea of composing services in a loosely and flexible way in order to support the rapid and low-cost development of applications has been widely accepted. Along with the increasing success of the service-oriented computing paradigm came a variety of languages, which aim at documenting service properties in order to support the selection and binding of services. Amongst them are approaches like the Web Service Description Language or the Web Ontology Language for services. However, until today no approach was able to establish itself as a sufficient standard for a complete description of all relevant service properties. In this paper, we aim at classifying current web service description languages to identify the different properties that a service description language can cover. We furthermore debate on the characteristics of the analyzed approaches and highlight areas where further research is required.

Keywords

Service-oriented computing, state of the art, web service description languages.

INTRODUCTION

The service-oriented computing (SOC) paradigm promises developers a whole range of advantages and has accordingly become a well-known and widespread application building approach in research and industry (Mertz, Eschinger, Eid, Swinehart, Pang and Pring, 2009; Weerawarana, Curbera, Leymann, Ferguson and Storey, 2005). On the one hand, SOC helps decreasing the complexity of an application development project by allowing developers to break down a task into several smaller services. In engineering disciplines, such divide-and-conquer strategies are well known to reduce the complexity and therefore to lower costs and to raise quality (Kanigel, 1997; Speed, Councill and Heineman, 2001). On the other hand, SOC envisions the reuse of services which results in improvements like enhancing the flexibility or reducing the development time (Barros and Dumas, 2006; Papazoglou, Traverso, Dustdar and Leymann, 2007).

A prerequisite for truly realizing these advantages is that developers are able to compose applications by "discovering and invoking network-available services" (Papazoglou et al., 2007). In order to be able to discover and invoke services, developers need to be able to assess their characteristics, like e.g. interface descriptions, quality properties or the business semantics a service implements. Only consequently, numerous languages which are thought to document such characteristics have evolved along with the increasing popularity of SOC. These languages range from technically oriented and standardized languages like the Web Service Description Language (WSDL) to research approaches that focus on business semantics like e.g. the one proposed by Overhage and Schlauderer (2010b). Besides the WSDL, no approach was able to establish itself in practice until now. The WSDL, however, only documents the programming interface of services. Other properties of services are either not documented at all in practice or there is no standardized use of a language for these characteristics.

In order to point out which languages exist to describe the characteristics of services, we seek to classify current approaches. We thereby not only list the characteristics that can be described using a certain approach, but further discuss if a language is able to sufficiently specify these characteristics. We thus propose a classification scheme which in a first step distinguishes between different abstraction levels of functionality like e.g. the quality or the business semantics of a service. We then refine the abstraction levels on the basis of the views of general systems theory (Bertalanffy, 1976) to further structure them. Additionally, we introduce further criteria like the representation of an approach or the degree to which a language is formally specified. Such a classification not only allows to get an overview about existing approaches with their individual characteristics, but also illustrates in which areas future research is necessary.

The remaining paper is organized as follows: in the next section, we briefly discuss related work to confirm the research gap. In section three, we derive the classification scheme and the parameters used to characterize languages as well as their possible values. The resulting classification and the discussion of the analyzed languages are presented in section four. We conclude by discussing implications of our work and highlighting possible future research directions and limitations.

RELATED WORK

Regarding the description of services, there is a large variety of literature. On the one side, approaches aiming at specifying services are broadly defined and it exists a huge amount of documentation for them. For example, the World Wide Web Consortium (W3C) currently provides not less than ten completed working papers for the WSDL. On the other side, there exist numerous reviews adjacent to the documentation of specific approaches. These reviews range from theoretical analyses (e.g. Klein, Konig-Ries and Mussig, 2005) to practically driven analyses (e.g. Costa, Sampaio and Alves, 2009).

Despite the large quantity of literature for each individual approach, there exists little literature that summarizes and compares service description languages. In particular, we only found three approaches which address the current state of the art of service description languages (Haddad, 2009; Toma, Steinmetz and Lorre, 2008; Viganò, 2008). However, all of them seem to be part of larger industry or research projects and are not published in scientific outlets. Furthermore, they do not focus on classifying service description languages or showing future research directions. The work of Toma et al. (2008) merely summarizes current service description and discovery techniques to provide an overview, without categorizing or discussing them. The report published by Haddad (2009) solely examines service interface description techniques and discusses them with regard to their potential for an overall project. By contrast, the report of Viganò (2008) has a very broad focus and discusses all kinds of specification languages for Service-oriented Architectures (SOA). It, however, names only very few approaches regarding the specification of services. Articles that focus on the state of the art of service description languages and categorize these approaches do – to the best of our knowledge – not exist.

THEORETICAL FOUNDATION

In a first step we categorize what a service actually does, i.e. what kind of functionality it provides. As related approaches, we aim at documenting "in precise terms the intended effect of a piece of software" (Gehani and McGettrick, 1986). Later on, we add other criteria like the representation format or the formality of service description languages into the classification scheme.

For the classification of the functionality, we distinguish between the aspects of functionality a service description language can focus on and the views of general systems theory. The aspects of functionality a service description language can focus on can be determined by the following three levels of abstraction (D'Souza and Wills, 1999; Olle, Hagelstein, MacDonald and Rolland, 1991; Scheer, 2000): the business level expresses which business semantics a service provides, e.g. the business tasks a service supports or the business context in which it operates. A typical business context could e.g. be "warehousing" and supported business tasks could be "static storage" or "first-expired-first-out commissioning". Describing a service in this way points out that the corresponding service supports warehousing with a fixed-bin (as opposed to chaotic) storage strategy. These facets are defined during the conceptual design phase of the development process and are described in business terms. The architectural level of a service describes its programming interface and gives information about the technical integration of a service into an application system. Such a description could for instance point out that an operation called "check_inventory" needs an integer as input. The architectural level is defined during the technical design phase and described using computer-oriented languages. The quality level of a service describes the degree to which a service meets non-functional requirements and comprises information about e.g. the reliability or the maintainability. Such information could for example be that the mean response time of a service is less than half a second. The quality results from its implementation and can be documented using metrics like the ones given in the ISO 9126 quality model (ISO/IEC, 2001).

Systems View \ Abstraction level	Business level (conceptual design)	Architectural level (technical design)	Quality level (implementation)
Static view (structure)	I Organizations, stakeholders, information items	IV Signatures (type and interface declarations)	VII Usability, maintainability, portability
Functional view (capabilities)	II Tasks, activities, events	V Assertions (pre- and post-conditions, invariants)	VIII Functionality (security, persistency, transactions)
Dynamic view (execution)	III Processes, work-flows	VI Timing constraints (interaction protocols)	IX Reliability, efficiency

Table 1. Classification scheme for the functionality (on the basis of (Overhage et al., 2010b))

These three abstraction levels can be further structured according to their perspective. Following the general systems theory (Bertalanffy, 1976) and the traditional distinction between modeling perspectives (Arbnor and Bjerke, 2009), we decided to

differentiate these perspectives into a static, functional, and a dynamic view. The *static view* describes the structure of a software artifact. Regarding the business level of a service, this includes properties like the information items that are being processed or the participating business units (Scheer, 2000). On the architectural level, possible properties are type and interface declarations (D'Souza et al., 1999) while on the quality level they are usability or maintainability (ISO/IEC, 2001). The *functional view* determines the capabilities of a piece of software. It thereby characterizes properties like the business tasks or activities that are supported on the business level (Scheer, 2000), the technical pre- and post-conditions or invariants on the architectural level (Beugnard, Jézéquel, Plouzeau and Watkins, 1999), and the provided persistency or security on the quality level (ISO/IEC, 2001). The *dynamic view* gives information about the way a software artifact executes at run-time. On the business level it specifies the processes or work-flows that are supported (Scheer, 2000). On the architectural level it comprises timing constraints and therewith the sequence of interactions between the interface operations (Beugnard et al., 1999). Finally, on the quality level the dynamic view covers run-time properties like the reliability or the efficiency of a software artifact. The above discussed levels of abstraction with the corresponding system views as well as their possible properties are summarized in Table 1.

Table 1 can not only be used to describe the intended effect of a piece of software, but can in this context also be used to classify the properties that a service description language should specify in order to facilitate the selection and binding of services. Section 4 therefore discusses which of the matrix fields (I to IX) are covered by a certain approach. In this way, it is possible to get an overview about the areas that current service description languages are able to cover and to highlight in which areas future research is necessary. However, to further elaborate on the individual aspects of an approach, a more detailed classification is necessary. In order to archive this, we include the following criteria into the classification scheme. These criteria partly result from a study of comparable research that has been conducted in the area of software architecture description languages (Clements, 1996; Medvidovic and Taylor, 2000). We complemented these criteria with others that were found to be necessary for the specification of software components (Davis, 1993; Pressman, 1997; Thayer and Dorfman, 1990). However, some of the criteria seemed to be very abstract and difficult to quantify. For these criteria, we introduced measures that are more specific and easier to quantify. For instance, Clements (1996) points out that a language should be consistent. While the consistency of a language is difficult to measure, we checked if a language is defined by a meta-model which ensures a consistent use of the language and noted if it is standardized by a particular organization which monitors the language. We furthermore introduce pre-defined values for each criterion. We therefore took possible values of a criterion into consideration and derived categories. Doing so allowed us to reduce the number of possible values and to increase the comparability of the approaches.

The criterion s*tandardization* expresses if an approach is standardized by a consortium like the Object Management Group (OMG). As possible values, we consider that an approach is currently in the standardization process, already standardized, or that no standard exists. *Formality* summarizes the way that the use of a description language is defined and specified. The most formal way is if a description language has a meta-model which prescribes the use of the language in detail and independent of a certain representation format. If the execution of a language is specified only for a certain representation format, we state this as an existing grammar for a language. Besides that, some approaches solely present a concept without explicitly prescribing how it can be used.

Criterion	Values
Standardization	None | In progress | Standard exists
Formality	None | Grammar | Meta model
Representation	None | Keywords | Graphical | Technical languages | Other
Validation	Empirical | Theoretical | None
Integration	Independent approach | Must be integrated
Paradigm	SOA | SaaS | SOC | Not specified | Other
Use in practice	None | Partially | De facto standard
Functionality	Matrix fields I-IX (Table 1)

Table 2. Criteria for the classification of service description languages

Another aspect which varies widely between the approaches is the *representation format*. In order to avoid a large variety of possible values, we differentiated between approaches which use technical languages, graphical languages, languages that consist of keywords, and other forms to represent the content. Moreover some of the approaches do not provide an explicit

representation format. *Validation* embodies in how far a description language has been validated so far. For this criterion, we checked if we found literature in which the description language was either empirically or theoretically evaluated. The *integration* of an approach expresses if the approach stands for itself. We distinguished between approaches that must be integrated into other approaches and the ones that can be used independently.

The criterion *paradigm* pronounces in which context a service description language was developed. We did thereby not want to discuss the differences between the design paradigms, but rather reflected which paradigm the specification of a language has pronounced. We discriminated between service description languages that focus on SOA, SOC, and software as a service (SaaS). We furthermore wanted to know if and how often a service description language is used in practice. Hence we examined current service marketplaces (i.e. StrikeIron, Salesforce's AppExchange, Google's Apps Marketplace, and the WebCentral ApplicationMarketplace) for the criterion *use in practice*. We then categorized the approaches according to how often they were used and distinguished between de facto standards, languages that are partially used, and languages that could not be found in practice. Finally, we included the *functionality* aspect as illustrated in Table 1. Table 2 summarizes the discussed criteria as well as their possible values.

CLASSIFICATION

In this chapter, the service description languages are discussed with regard to the criteria in Table 2. Knowing that services could also be described with general approaches like the Systems Modeling Language (SysML) or other approaches that have a broad scope, we explicitly decided to only consider languages that focus on the description of services. The choice which service description languages were analyzed was then made as follows: first of all we distinguished between research and industry-driven approaches. To identify industry-driven approaches, we examined current service marketplaces as well as standardization consortia like the W3C or the OMG. To identify research-driven approaches, we followed the recommendations of Webster and Watson (2002). In the first step, we analyzed leading journals and conferences for relevant approaches. We then reviewed the citations for the identified articles in order to find prior approaches. As we could not discuss all of the identified approaches due to reasons of brevity, we had to narrow them down. We therefore related the approaches to one of the abstraction levels of Table 1. If there had been too many approaches covering an abstraction level, we followed step three of Webster's recommendation (2002) and used the citation index to determine which approach should be included. This results in the following service description languages.

The WSDL is a standardized specification language of the W3C and besides its comprehensive documentation, it also exists considerable scientific literature about the WSDL. Furthermore it was the only service description language that was used on one of the examined service marketplaces (StrikeIron). However, we could not identify another marketplace providing WSDL files, which is why we categorized its use in practice as partially. The WSDL allows users to specify signature lists, e.g. interfaces, element declarations, or protocol bindings (Chinnici, Moreau, Ryman and Weerawarana, 2007) and hence the static view of the architecture of a service. Regarding the functionality, the WSDL covers the matrix field IV in Table 1. For the description of these properties it provides a grammar which prescribes certain rules how it has to be used. The WSDL moreover uses a representation format (XML) which we categorized as technical language. Concerning the criterion paradigm, we could not find a specific reference in the documentation of the WSDL. The WSDL is exclusively developed for the specification of services and can be used independently from other approaches. It was already empirically evaluated (e.g. Overhage et al., 2010b) as well as theoretically analyzed (e.g. Wiley, Aihua and Jianwen, 2008).

There also exist approaches which aim at extending the WSDL with semantic information. The Web Service Semantics (WSDL-S) approach therefore uses the WSDL as conceptual base, but allows users to add more information. Such information can include the definition of preconditions, inputs and outputs, and the effects of operations (Akkiraju, Farrell, Miller, Nagarajan, Schmidt, Sheth and Verma, 2005). Using this approach hence makes it possible to specify the matrix field V of Table 1, in addition to the field IV which is already covered by the basic WSDL. As the WSDL-S approach mainly builds upon the WSDL, its formality, representation format, and paradigm are identical. It furthermore cannot be regarded as an independent approach, but has to be integrated into the WSDL. In contrast to the WSDL, it is only published by the W3C as submission and not as recommendation or draft. While we could identify an article which discussed and evaluated the approach (Li, Verma, Mulye, Rabbani, Miller and Sheth, 2006), we were not able to identify any marketplace which provided WSDL-S files to customers. Next to WSDL-S, there exist other approaches which augment WSDL files with semantics. One example is the Semantic Annotations for WSDL and XML Schema (SAWSDL). The advantages of this approach have been discussed by Ting-Xin, Pei-Jun, Yao-He and Bi-Qing (2008). It is also based on WSDL and can therefore describe the same properties, besides that it provides a mechanism which allows users to annotate WSDL files with semantic models (Farrell and Lausen, 2007). These models can be used to further specify the characteristics of a service, yet SAWSDL does not specify these semantic models. As SAWSDL moreover does not prescribe that the approach has to be annotated with semantic models and as miscellaneous models can be used, we could not consider this factor in our classification.

Another approach trying to add semantics to the description of Web services is the Web Ontology Language for Services (OWL-S). In contrast to the aforementioned approaches, OWL-S is not based on WSDL and can be used independently. Providing the three sub-ontologies ServiceProfile, ServiceGrounding, and ServiceModel gives users the opportunity to specify signatures, assertions, and timing constraints (Martin, Burstein, Hobb, Lassila, McDermott, McIlraith, Narayanan, Paolucci, Parsia, Payne, Sirin, Srinivasan and Sycara, 2004). In this way OWL-S covers the fields IV, V, and VI of Table 1 and can be used to unequivocally describe the architectural properties of a service. Although it is possible to use taxonomies in order to relate a service to categories and therewith to express the application domain of a service, we did not consider OWL-S as an approach to describe the business semantics of a service from a business-oriented point of view. OWL-S is a member submission and therefore no W3C standard. It furthermore uses XML for the representation and has a formal grammar that prescribes the use of the language. It was for instance already theoretically analyzed by Martin, Burstein, Mcdermott, Mcilraith, Paolucci, Sycara, Mcguinness, Sirin and Srinivasan (2007) or Balzer, Liebig and Wagner (2004).

The Service oriented architecture Modeling Language (SoaML) is a specification language which provides a Unified Modeling Language (UML) profile and a meta-model for the description of service-oriented architectures in general. The description of services is only one aspect of this approach. The SoaML focuses on the architectural properties of services and can be used to specify signatures, assertions and timing constraints. It hence covers the fields IV to VI of Table 1. It also gives users the opportunity to create a link to Business Motivation Models (BMM), which then depict how a service fits into the business plan (OMG, 2009). As this link is only optional and BMM are not specifically designed for services, we did not consider SoaML as a language to describe the business semantics. Until now, the approach is still in the standardization progress as the OMG only published a second beta version. Since it is based on an UML profile, it has a graphical representation format and can be used independently of other approaches. We could not identify marketplaces that provided any kind of SoaML documents, but we could find an approach which theoretically discussed SoaML (Gebhart, Baumgartner, Oehlert, Blersch and Abeck, 2010).

The Web Service Modeling Language (WSML) provides a concrete syntax to describe the elements specified in the Web Service Modeling Ontology (Bruijn, Fensel, Keller, Kifer, Lausen, Krummenacher, Polleres and Predoiu, 2005). This language can be used to describe interfaces, pre-and post-conditions, and choreographies. It hence covers the fields IV to VI of Table 1. The Web Service Modeling Ontology serves as meta-model for this language and ensures that the language can be used independently. The syntax is based on XML and logical expressions, which is why we categorized it as technical language. It was for instance theoretically discussed in the doctoral thesis of O'Sullivan (2006) and its specification does not pronounce a specific paradigm. Of the examined service marketplaces, we could not find one that depicts the provided information about services using the WSDML.

Description Language	Standardiza-tion	Formality	Representa-tion	Validation	Integration	Paradigm	Use in practice	Functionality
WSDL	Standard exists (W3C)	Grammar	Technical language	Empirical & theoretical	Independent approach	Not specified	Partially	Array IV (Table 1)
WSDL-S	None	Grammar	Technical language	Empirical & theoretical	Must be integrated	Not specified	None found	Array IV-V (Table 1)
SAWSDL	Standard exists (W3C)	Grammar	Technical language	Theoretical	Must be integrated	Not specified	None found	Array IV (Table 1)
OWL-S	None	Grammar	Technical language	Theoretical	Independent approach	Not specified	None found	Array IV- VI (Table 1))
SoaML	In progress (OMG)	Meta model	Graphical language	Theoretical	Independent approach	SOA	None Found	Array IV-VI (Table 1)
WSML	None	Meta model	Technical language	Theoretical	Independent approach	Not specified	None Found	Array IV-VI (Table 1)
WS-Functionality	None	Meta model	Other	Empirical	Independent approach	SOC	None Found	Array I-III (Table 1)
WS Policy	Standard exists (W3C)	Grammar	Technical language	Theoretical	Must be integrated	Not specified	None Found	Array VII-IX (Table 1)
WS-Agreement	Standard exists (OGF)	Grammar	Technical language	Theoretical	Must be integrated	Not specified	None Found	Array VII-IX (Table 1)

Table 3. Classification of service description languages

A research approach that solely focuses on the description of the business semantics of services is the WS-Functionality language proposed by Overhage et al. (2010b). Providing a meta-model, which distinguishes between information items, functions, and processes, enables this language to completely specify the business abstraction level. Hence this approach covers the fields I to III of Table 1. As it provides a meta-model, its representation format is not fixed to one form, yet the approach was presented using a representation format which regulates the natural language by providing sentence patterns. The meta-model also ensures that the approach can be used independently and does not need to be integrated into any other

approach. The authors empirically validated their approach against WSDL files. Additionally, the approach is presented in a SOC context and we could not detect any marketplace which provides this approach to customers.

The WS-Policy approach provides constructs that can be used to specify the requirements of services (Vedamuthu, Orchard, Hirsch, Hondo, Yendluri, Boubez and Yalçinalp, 2007). In particular, it allows users to define policies that are thought to describe the general quality characteristics or requirements of a service. Thus, this approach is able to define the usability, maintainability, security, and reliability of services, and covers the fields VII to IX of Table 1. The WS-Policy approach is a W3C recommendation and provides a formal grammar on how to use the language. It is usually integrated into WSDL files, but could also be integrated into the Process Execution Language for example. As it is based on XML we classified it as technical language. We moreover were not able to analyze WS-Policy files on the examined service marketplaces. However, it was already theoretically evaluated by e.g. Weerawarana et al (2005).

Another language that can be used for the specification of quality properties is the Web Services Agreement Specification (WS-Agreement). This approach uses XML and allows providers and consumers to make agreements. While the specification gives examples what these agreements might look like (e.g. response time or service availability), it does not explicitly specify them (Andrieux, Czajkowski, Dan, Keahey, Ludwig, Nakata, Pruyne, Rofrano, Tuecke and Xu, 2007). By adding agreement terms like the ones used in the ISO/IEC standard 9126 (2001), this approach is able to specify the fields VII to IX of Table 1. The specification is recommended by the Open Grid Forum (OGF) and includes no reference to a certain paradigm. It was already theoretically analyzed by Aiello, Frankova and Malfatti (2005), who furthermore extended the approach by formally defining agreements. Yet we were not able to find this approach in practice. Next to the WS-Policy and the WS-Agreement approach, there moreover exists a whole set of so called WS-* languages which mainly focus on technical or quality properties (e.g. WS-Addressing or WS-Security). However, due to reasons of brevity we could not discuss all of them. Table 3 summarizes the classification of the examined service description languages and gives a brief overview about the characteristics of these approaches.

	Business semantics (conceptual design)	**Software architecture (technical design)**	**Quality attributes (implementation)**
Types (static view)	I WS-Functionality	IV WSDL, WSDL-S, SAWSDL, OWL-S, SoaML, WSML	VII WS-Policy, WS-Agreement
Functions (operational view)	II WS-Functionality	V WSDL-S, OWL-S, SoaML, WSML	VIII WS-Policy, WS-Agreement
Flows (dynamic view)	III WS-Functionality	VI OWL-S, SoaML, WSML	IX WS-Policy, WS-Agreement

Table 4. Abstraction levels covered by service description languages

Table 4 outlines which abstraction levels are covered by which service description languages. Illustrating the functionality covered by an approach in this way helps pointing out which abstraction levels can be described by current description languages and in which areas future research might be relevant. It first of all highlights that we were able to identify many approaches which cover the description of the software architecture or of quality attributes. Regarding these two abstraction levels, we also excluded some of the identified approaches (e.g. some of the WS-* approaches). In contrast, we could only identify one research approach that aims at documenting the business semantics of services from a business-oriented point of view. However, in contrast to standardized languages like the WSDL, this approach is still in a very early state of its development. On the examined service marketplaces, the business semantics of services was only described by a few sentences expressed in natural language, demo-videos, or screenshots. Of course it is possible to get an overview about the provided functionality by analyzing such information. As well as it is possible to do so by evaluating technical description languages like the WSDL and examining the specified operations. Gaining detailed information about the business semantics of a service to e.g. find out if a service supports a chaotic or a fixed-bin warehousing strategy is a difficult task based on such information, however. It is furthermore questionable if a selection process which is based on such information is efficient enough. We could only identify one study by Overhage et al. (2010b) in which users had to choose a service which offers the appropriate business semantics based on the information provided by either a technical description (WSDL) or by a language that explicitly concentrates on the description of business semantics (WS-Functionality). In this study, the technical language was significantly outperformed in terms of efficiency and effectiveness. This leads to the conclusion that the explicit description of the business semantics a service contains is noteworthy and that future research will have to analyze how the business semantics of a service should be described.

CONCLUSIONS

In this paper, we examined current service description languages with respect to the abstraction levels they are able to cover. One of the limitations of this research is that we were not able to give a complete overview about all existing service description languages due to reasons of brevity. However we tried to depict at least two important service description languages for each abstraction level and discussed them. As we limited our scope to languages that are explicitly designed for the description of services, we could discuss most of the identified languages. We moreover introduced a classification scheme which helped us further discussing the characteristics of service description languages. The results serve as a first verification of the classification scheme as we were able to assign all of the identified service description languages to the fields of Table 1. Such a classification scheme can in future research not only be used to assess which aspects a service description language is able to cover, but it can furthermore be used to discuss how complementary approaches could be combined. Such insights could be particularly interesting for the formation of universal approaches which aim at completely describing all characteristics of services.

The results furthermore have implications for both academia and practice. For academia it in particular highlights in which areas future research is necessary. As already discussed, it for example shows that no standardized language for the description of the business semantics of a service exists. It moreover illustrates that no language exists which is able to cover all three abstraction levels and which can hence be used to completely specify the characteristics of a service. A cooperation of industry and academia currently addresses this issue by proposing the Unified Service Description Language (USDL). However, this approach was not part of the analysis as they themselves state that some of the USDL modules do not yet have a sufficient degree of maturity. In how far this approach is able to adequately document all of the characteristics depicted in Table 1 will have to be analyzed when the approach has reached a stable version.

Another area where future research might be interesting is the empirical evaluation of approaches. It would thereby be particularly interesting to see comparisons of different service description languages with respect to the selection process. On the basis of such evaluations, it would be possible to consider subjective measures like for instance the effectiveness, the efficiency, or the usability of a service description language. Such subjective criteria could then complement a theoretically driven classification scheme. Based on this information one might be able to determine which service description languages are superior over others regarding the selection or binding process of services. As we hardly found any research in which service description languages are empirically evaluated against each other, we were not able to include these criteria into our classification scheme, though.

For practice, the results indicate that current marketplaces are far away from describing services homogenously. Except of StrikeIron, none of the examined service marketplaces provided any kind of description language to depict information about services. And even StrikeIron solely provided WSDL files. An efficient searching process based on such information seems to be a difficult task, however. Consequently a study in which 14 practitioners were interviewed showed that they determined the lack of information on current service marketplaces to be a major market immaturity. They furthermore stated that the provisioning of comprehensive service specifications is a critical factor for the success of service marketplaces (Overhage and Schlauderer, 2010a). It hence seems to be of major relevance for practice as well as for academia to establish standardized languages for the description of services. What these service description languages should ideally look like could be identified by further empirical comparisons of service description languages. Giving users the opportunity to choose services on the basis of such evaluated service description languages would then support an efficient searching process with homogenously described services.

REFERENCES

1. Aiello, M., Frankova, G., and Malfatti, D. (2005) What's in an Agreement? An Analysis and an Extension of WS-Agreement, in B. Benatallah, F. Casati and P. Traverso (Eds.) *Service-Oriented Computing - ICSOC 2005*, Springer Berlin / Heidelberg, 424-436.
2. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., and Verma, K. (2005) Web Service Semantics - WSDL-S, Version 1.0, W3C Member Submission, World Wide Web Consortium.
3. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2007) Web Services Agreement Specification, Recommendation, GFD.107, Open Grid Forum.
4. Arbnor, I., and Bjerke, B. (2009) Methodology for Creating Business Knowledge, 3 ed., London.
5. Balzer, S., Liebig, T., and Wagner, M. (2004) Pitfalls of OWL-S: a practical semantic web use case *Proceedings of the 2nd international conference on Service oriented computing*, New York, NY, USA, ACM, 289-298.
6. Barros, A.P., and Dumas, M. (2006) The Rise of Web Service Ecosystems, *IT Professional*, 8, 5, 31-37.

7. Bertalanffy, L.v. (1976) General System Theory, George Braziller, New York, NY.
8. Beugnard, A., Jézéquel, J.-M., Plouzeau, N., and Watkins, D. (1999) Making Components Contract Aware, *IEEE Computer*, 32, 7, 38-45.
9. Bruijn, J.d., Fensel, D., Keller, U., Kifer, M., Lausen, H., Krummenacher, R., Polleres, A., and Predoiu, L. (2005) Web Service Modeling Language (WSML), Member Submission, W3C.
10. Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007) Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, World Wide Web Consortium.
11. Clements, P.C. (1996) A Survey of Architecture Description Languages *Proceedings of the 8th International Workshop on Software Specification and Design*, IEEE Computer Society, 16-25.
12. Costa, T., Sampaio, A., and Alves, G. (2009) Using SysML in Systems Design *Information Management, Innovation Management and Industrial Engineering, 2009 International Conference on*, 615-618.
13. D'Souza, D.F., and Wills, A.C. (1999) Objects, Components, and Frameworks with UML: The Catalysis Approach, Addison-Wesley, Upper Saddle River, NJ.
14. Davis, A.M. (1993) Software Requirements: Objects, Functions, and States, Prentice Hall, Englewood Cliffs, NJ.
15. Farrell, J., and Lausen, H. (2007) Semantic Annotations for WSDL and XML Schema, W3C Recommendation World Wide Web Consortium.
16. Gebhart, M., Baumgartner, M., Oehlert, S., Blersch, M., and Abeck, S. (2010) Evaluation of Service Designs Based on SoaML *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, 7-13.
17. Gehani, N., and McGettrick, A.T. (1986) Software Specification Techniques, Addison-Wesley, Wokingham.
18. Haddad, J.E. (2009) State of the Art : Languages for Services Interface Description and for Services Composition, R 1.a ANR JC07_186508.
19. ISO/IEC (2001) Software Engineering - Product Quality - Part 1: Quality Model, ISO/IEC Standard 9126-1, International Organization for Standardization.
20. Kanigel, R. (1997) The One Best Way: Frederick Taylor and the Enigma of Efficiency, Viking Penguin, New York, NY.
21. Klein, M., Konig-Ries, B., and Mussig, M. (2005) What is needed for Semantic Service Descriptions? A Proposal for Suitable Language Constructs, *International Journal of Web and Grid Services*, 1, 3/4, 328-364.
22. Li, K., Verma, K., Mulye, R., Rabbani, R., Miller, J., and Sheth, A. (2006) Designing Semantic Web Processes: The Wsdl-S Approach, in J. Cardoso and A.P. Sheth (Eds.) *Semantic Web Services, Processes and Applications*, Springer US, 161-193.
23. Martin, D., Burstein, M., Hobb, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004) OWL-S: Semantic Markup for Web Services W3C Member Submission, World Wide Web Consortium.
24. Martin, D., Burstein, M., Mcdermott, D., McIlraith, S., Paolucci, M., Sycara, K., Mcguinness, D.L., Sirin, E., and Srinivasan, N. (2007) Bringing Semantics to Web Services with OWL-S, *World Wide Web*, 10, 3, 243-277.
25. Medvidovic, N., and Taylor, R.N. (2000) A Classification and Comparison Framework for Software Architecture Description Languages, *Software Engineering, IEEE Transactions on*, 26, 1, 70-93.
26. Mertz, S.A., Eschinger, C., Eid, T., Swinehart, H.H., Pang, C., and Pring, B. (2009) Market Trends: Software as a Service, Worldwide, 2008-2013 Gartner Inc., Stamford.
27. O'Sullivan, J. "Towards a Precise Understanding of Service Properties," in: *Faculty of Information Technology*, Queensland University of Technology, 2006.
28. Olle, T.W., Hagelstein, J., MacDonald, I.G., and Rolland, C. (1991) Information Systems Methodologies: A Framework for Understanding, Addison-Wesley, Wokingham.
29. OMG (2009) Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS), Beta 2, ptc/2009-12-09, Object Management Group.
30. Overhage, S., and Schlauderer, S. (2010a) The Market for Services: Economic Criteria, Immaturities, and Critical Success Factors, in IEEE (Ed.) *Proceedings of the 43rd Hawaii International Conference on System Sciences*, Koloa, Kauai, HI, IEEE Computer Society Press, Los Alamitos, CA, 1-10.

31. Overhage, S., and Schlauderer, S. (2010b) What's in a Service? Specifying the Business Semantics of Software Services *Proceedings of the ICIS*, Saint Louis, MO, USA
32. Papazoglou, M.P., Traverso, P., Dustdar, S., and Leymann, F. (2007) Service-Oriented Computing: State of the Art and Research Challenges, *IEEE Computer*, 40, 11, 38-45.
33. Pressman, R.S. (1997) Software Engineering: A Practitioner's Approach, McGraw-Hill, New York, NY.
34. Scheer, A.W. (2000) ARIS - Business Process Frameworks, 3. ed., Springer, Berlin, Heidelberg.
35. Speed, J., Councill, W.T., and Heineman, G.T. (2001) Component-Based Software Engineering as a Unique Engineering Discipline, in W.T. Councill and G.T. Heineman (Eds.) *Component-Based Software Engineering: Putting the Pieces Together*, Addison-Wesley, Upper Saddle River, NJ, 673-691.
36. Thayer, R.H., and Dorfman, M. (1990) System and Software Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA.
37. Ting-Xin, S., Pei-Jun, T., Yao-He, L., and Bi-Qing, H. (2008) Web Services' Semantic Annotation and Auto-Matching Based on SAWSDL *Information Science and Engineering, 2008. ISISE '08. International Symposium on*, 577-580.
38. Toma, I., Steinmetz, N., and Lorre, J.P. (2008) State of the Art Report On Service Description and Existing Discovery Techniques.
39. Vedamuthu, A.S., Orchard, D., Hirsch, F., Hondo, M., Yendluri, P., Boubez, T., and Yalçinalp, Ü. (2007) Web Services Policy 1.5 - Framework, W3C Recommendation, World Wide Web Consortium.
40. Viganò, L. (2008) State-of-the-Art on Specification Languages for Service-Oriented Architectures, D6.2.1.
41. Webster, J., and Watson, R.T. (2002) Analyzing the Past to Prepare for the Future: Writing a Literature Review, *MIS Quarterly*, 26, 2, xiii-xxiii.
42. Weerawarana, S., Curbera, F., Leymann, F., Ferguson, D.F., and Storey, T. (2005) Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More, Prentice Hall, Upper Saddle River, NJ.
43. Wiley, M., Aihua, W., and Jianwen, S. (2008) WSDL-D: A Flexible Web Service Invocation Mechanism for Large Datasets *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, 157-164.