

OWL and OWL-S for Dependability-Explicit Service-Centric Computing

Glen Dobson

Computing Department, Lancaster University

g.dobson@comp.lancs.ac.uk

Abstract

In this position paper an attempt is made to relate dependability-explicit computing to the semantic web. The machine understandable nature of the semantic web suggests a way to reconcile the increasingly autonomous nature of service-based systems with the need to verify that, once deployed, systems conform to a dependability specification.

1. Introduction

A dependability-explicit development model is defined in [1] as centring around the notion that:

“...the means for dependability... should be explicitly incorporated in a development model focused at the production of dependable systems.”

The key aim of the model defined in [1] is to ensure that dependability is considered in a structured manner throughout a development process. Another key aspect of dependability-explicit computing is that a dependability specification can serve a purpose both at design-time and in dependability evaluation at runtime.

The runtime evaluation of dependability is a problem which becomes particularly important in the use of third-party, black-box software components. If the definition of dependability is taken to be the ability to deliver service that can justifiably be trusted [2] it can easily be seen why. The very nature of such components means that techniques applicable at design and development time no longer apply. In terms of the model presented in [1], even if dependability processes did take place during service development, the integrator does not necessarily know anything about them. Usually, the issue for the integrator of justifying their trust in a component's service therefore comes down to runtime evaluation. In this paper we will continue to use the term dependability evaluation as a catch-all for any technique used to establish trust in a service.

Web services are an example of black-box components with further problems particular to their

nature. The first of these is the use of the untrusted Internet channel of communication. The second is the possibility of dynamic binding in a service-based system. Dynamic binding means that the service instance to be used may not be decided until runtime. This means that the integrator will generally have no opportunity to assess service dependability. The evaluation will therefore have to be performed by the system itself and may indeed form part of the selection process for dynamic binding.

This paper discusses the application of dependability-explicit computing in the field of web services. The aim is to identify how explicit dependability information carried with a service can inform the provider, integrator and end-user as to how to make a decision on whether to trust a service.

A proposal, [3], covering much the same problem space, identified the development of an ontology as core to such an effort. We have already developed a dependability ontology as part of our Quality of Service Ontology (QoSOnt) [4], and are now working towards a unified QoS ontology with other researchers in the field [5]. We regard the use of semantic web technologies as the key to supporting dependability-explicit service-centric computing. Building upon these ideas, this paper also aims to discuss the effects that semantic web technology has upon the outputs of a dependability-explicit service development process. How the provider, integrator and end-user can use and add to these documents is also covered. The issue of what support for runtime dependability evaluation is provided by the use of semantic web technologies is also approached.

The remainder of this paper is structured as follows. OWL (the Web Ontology Language) is introduced in section 2. Section 3 then introduces the concept of specifying service dependability using OWL. Section 4 looks at potential problems with this approach, and the final section summarises the stated position and suggests directions for future work.

2. OWL and the Semantic Web

The semantic web is a movement which aims to make data machine understandable. By doing so, novel inferences can be made, in turn allowing machines to perform tasks that could previously only be performed by humans. A key part of the semantic web stack (Figure 1) is the ontology layer. An ontology is a machine understandable knowledge representation.

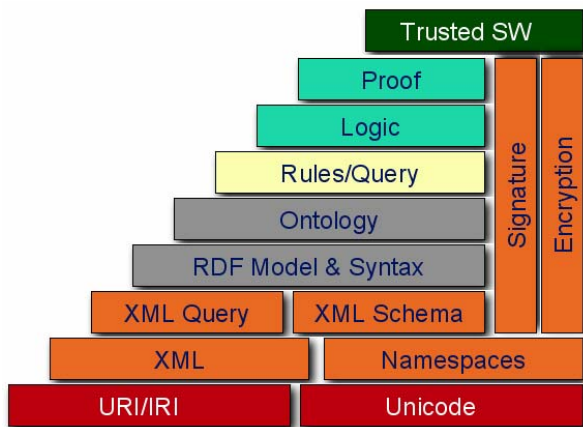


Figure 1. Semantic Web Stack

OWL (the Web Ontology Language) [6] has come to dominate this layer of the semantic web. OWL is an RDF (and therefore also XML) language. However, behind the RDF/XML syntax OWL is really a Description Logic (DL) [7]. Indeed the most useful “species” of OWL is named OWL-DL for this reason. A Description Logic is a logic that focuses on concept descriptions as a means of knowledge representation and has semantics which can be translated to first-order predicate logic. The nature of DLs means that classification, subsumption and satisfiability can be automatically computed by a reasoner.

An OWL ontology consists of Classes, which can be defined logically (chiefly by stating restrictions on their Properties). These Classes can be seen as sets of Individuals. New Class definitions can therefore be created by combining existing definitions using set operators (i.e. intersection, union, and complement). Class definitions can also be refined to create new Classes.

3. OWL for Dependability Specification

An OWL Class C states the exact conditions for Individuals to be classified, by a reasoner, as a C . A service specification in pure OWL would therefore consist of a Class S where everything which is an S is a

service meeting the specification. OWL-S [8] is an established OWL ontology which provides the relevant building blocks for describing (or in this case specifying) services. A functional (and to some extent behavioural) specification is possible using OWL-S alone. However, numerous supporting ontologies are required for complete service specification. One of the most notable absences is that of a Quality of Service (QoS) ontology encompassing dependability. With a QoS ontology aligned to OWL-S (e.g. [4]) it becomes possible to also specify non-functional characteristics of a service, including dependability. To demonstrate, a sketch is given below:

1. **SpecifiedValueRange** \equiv 0.0 – 0.01 inclusive
2. **SpecifiedPOFOD** \equiv POFOD $\sqcap \forall hasMeasurement. SpecifiedValueRange$
3. **SpecifiedServiceParameter** \equiv ServiceParameter $\sqcap \exists sParameter. SpecifiedPOFOD$
4. **SpecifiedServiceProfile** \equiv Profile $\sqcap \exists serviceParameter. SpecifiedServiceParameter$
5. **SpecifiedService** \equiv S $\sqcap \exists presents. SpecifiedServiceProfile$

The DL notation is used for succinctness. In OWL terms, all of the **bold** names above are Class names. The names in *italics* are Properties. \exists , and \forall are, respectively, the someValuesFrom and allValuesFrom property restrictions in OWL RDF/XML syntax; whilst \equiv and \sqcap are equivalentClass and intersectionOf respectively.

In short, all that is stated in the above specification is that as well as some functional specification S the service/s in question should also have POFOD (Probability of Failure on Demand). less than 0.01. Points 3, 4, and 5 are simply about relating the specification to the correct OWL-S constructs. POFOD is being used as an example reliability metric. In practice one would utilise multiple dependability metrics.

In point 5, it is assumed that an OWL Class representing a functional service specification S already exists. This would be formed by stating restrictions upon the OWL-S Class “Service”. 5 states that a service meeting the service specification is anything which meets the functional specification S and presents a ServiceProfile conforming to the SpecifiedServiceProfile. ServiceProfile is another

OWL-S Class, which provides hooks for advertising service QoS.

In 4, `SpecifiedServiceProfile` is defined as any Profile where at least one `serviceParameter` is a `SpecifiedServiceParameter`. A `ServiceParameter` in OWL-S may be any parameter one wishes to advertise about a service – but no parameters are actually defined in OWL-S itself. `QoSOnt` links to OWL-S by stating that a `ServiceQoSMetric` is both a `QoSOnt` `QoSMetric` and an OWL-S `ServiceParameter`.

In 3, it is stated that a `SpecifiedServiceParameter` is any `ServiceParameter` with at least one value which is a `SpecifiedPOFOD`. `POFOD` is assumed to be a `ServiceQoSMetric` from `QoSOnt`.

2 states that a `SpecifiedPOFOD` is any `POFOD` where all measurements are within the specified range. In practice the specification of this range in 1 is somewhat problematic in OWL (see Section 4).

3.1. The Advantages of Ontology-Based Dependability Specification

A great advantage of using an ontology for specification is that it includes the definition of a controlled vocabulary. With the potential of confusion between parties, and domains about the meaning of dependability metrics an ontology is therefore extremely valuable for disambiguation. Because OWL is basically a DL the machine rather than the human can perform the disambiguation.

A further advantage is that an OWL reasoner can also inform the service specifier if they have stated an unsatisfiable specification

On top of this OWL provides a model for refinement of specifications, e.g. by intersecting with further conditions as demonstrated in Section 3 or by explicit subclassing. Where a specification is implicitly a refinement of another specification a reasoner can make this explicit through inferring the subsumption relationship. The formal model of refinement is particularly important as not only the developer may have specifications to place on a service, but the provider may have further aspects to specify when the service is in situ, as might the service integrator. Eventually the service customer may also use a service specification as a means of service discovery or as the basis for a Service Level Agreement. At each stage the party in question may have new conditions to add or further constraints on existing conditions to add.

The final, and perhaps most important, advantage of using an OWL ontology for service specification is that an OWL reasoner can automatically check whether a given service meets the specification. Doing so simply consists of asking an OWL reasoner to perform

classification on the service/s description. If the description does not meet the specification it will simply not be classified under the specification Class. This, of course, assumes that there is a reliable source of the dependability characteristics of a given service expressed as a service description in OWL-S/QoSOnt. At this stage we are not proposing any specific measurement or monitoring technologies. The linking up of a particular monitoring technology to an ontology is a relatively trivial exercise.

4. Potential Problems with the Approach

As hinted at in the previous section there are some problems with specifying numerical ranges in OWL, which has an obvious knock on in specifying measurable QoS. These difficulties arise largely from imperfections in the original OWL specification meaning that it is not clear how to refer to XML schema datatypes from OWL (see [9] for more details). Thankfully the OWL 1.1 draft suggests a standard solution for this problem and so it is envisioned that consistent tool support should emerge in the near future for expressing and reasoning with numerical ranges in OWL.

Another potential problem with the approach is that inference over large OWL knowledge bases is notoriously slow. Thankfully, only a small number of services are ever likely to be checked against a specification, so for the application in question this might only be a minor problem.

The expressiveness of OWL is also in question, whilst the use of related, but more expressive rules languages means that inference may not terminate. These issues are therefore important for experimental investigation.

5. Conclusion

In this paper a position has been stated on the use of OWL for dependability specification. This has potential advantages not only in terms of disambiguation of specifications, but in machine support for runtime dependability evaluation. It also provides a model for refinement of specifications throughout the service lifecycle by providers, integrators and customers.

Future work should aim to create a prototype system for runtime dependability verification. Clearly, the approach proposed also applies to wider QoS characteristics – dependability is used only because it is an important, tractable sub-problem. The dependability prototype could be used to assess the performance and expressibility worries expressed in

Section 4. Meanwhile, a widespread agreement on a QoS ontology as is the aim of [5] is also an important pre-requisite.

6. Acknowledgements

This work is funded by EU Integrated Project 511680 – Service Centric System Engineering (SeCSE).

7. References

- [1] M. Kaaniche, J.-C. Laprie, and J. P. Blanquart. “A Framework for Dependability Engineering of Critical Computing Systems,” *Safety Science*. vol. 40, no. 9, pp. 731-752, 2002
- [2] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, Carl Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 01, no. 1, pp. 11-33, 2004
- [3] John Fitzgerald, Savas Parastatidis, Alexander Romanovsky and Paul Watson. “Dependability-explicit

Computing in Service-oriented Architectures.” Fast abstract at DSN, 2004

[4] Glen Dobson, Russell Lock and Ian Sommerville, “Quality of Service Requirements Specification using an Ontology”, SOCCER Workshop, 2005

[5] OWL-QoS collaboration, http://www.comp.lancs.ac.uk/owl_qos

[6] Deborah L. McGuinness, Frank van Harmelen (editors), “OWL Web Ontology Language Overview”, W3C Recommendation, <http://www.w3.org/TR/owl-features/>, 2004

[7] Franz Baader, Ian Horrocks, and Ulrike Sattler, “Description logics for the semantic web”, in *Künstliche Intelligenz*, Vol. 16, No. 4, pp. 57-59, 2002

[8] David Martin et al, “Bringing Semantics to Web Services: The OWL-S Approach”, *Lecture Notes in Computer Science*, Volume 3387, 2005, pp. 26 - 42

[9] Glen Dobson, “Numeric Ranges and Custom XML Datatypes in OWL”, <http://www.comp.lancs.ac.uk/computing/users/dobson/modules.php?name=News&file=article&sid=3>