Faceted Service Specification

Pete Sawyer, John Hutchinson, James Walkerdine, Ian Sommerville
Computing Department, Lancaster University, Lancaster, LA1 4YR, UK
{sawyer, hutchinj, walkerdi, is}@comp.lancs.ac.uk

Abstract

Orthodox requirements engineering in a traditional software engineering setting is geared towards expressing and then satisfying a baseline set of requirements. That baseline represents a balance between priorities in users requirements and the available resources, with the assumption that a change to the baseline will result in a change in the system being developed. Creating systems from services, especially off-the-shelf or 3rd party services presents huge challenges to the way requirements are captured and expressed because each service, fixed and specified by its developer or provider, will have a different match to the required functionality and cost. The result is that service specification is of primary importance to the requirements engineer. We present a faceted approach to service specification, which provides an extensible structure for managing the different and competing formalisms for describing services for potential users.

Keywords: Services, specification, facets.

1. Introduction

Orthodox requirements engineering practice, developed over many years to support custom software development, is geared towards satisfying a baseline set of requirements. This baseline is fixed by analysis of the users' requirements, balancing the requirements' relative priorities against the resources available for their satisfaction. COTS software has caused this practice to be re-evaluated. There are strong economic and quality motivations for using shrink-wrapped software components. However, the ability of the available components to satisfy the requirements introduces an additional critical factor in establishing the functionality to be delivered.

The advent of service-based systems looks likely to further challenge RE practice. The relative ease of integration offered by SOAP and other standards in web services is expected to stimulate a market in services. This will mean that system designers are likely to have available a range of services, each with a different match to the required cost and functionality. In addition, this trade-off may not be fixed at design time. Run-time binding mechanisms offer the possibility for applications, such as those running on mobile devices, to invoke different services according to context. In many cases, these will be selected from a set of verified services established at design time. However, it is also possible to envisage circumstances where services are discovered at run-time too.

A service provider wishing to promote their service needs to help the potential service consumer evaluate the service against the application requirements by publishing a service specification. A service consumer, in this context, is a requirements engineer performing trade-off analysis of requirements against service capabilities in order to identify a single service to statically bind to the application, or a set of services to allow the application to select from at run-time. A service consumer may also be an application performing service discovery in the most dynamic of run-time binding scenarios.

The crucial point here is that specification is one area where service-based systems differ markedly from custom or component-based systems. Service developers face all the usual problems regarding how to develop software that satisfies a specification of the software requirements and how those requirements are identified in the first case. The added dimension for service engineering is how a deployed service is described to users and application developers that want to use or integrate that service, how these users can match what is described to what they require and how they can verify that what is described is what is delivered.

Like third-party components used in CBSE, services are opaque to users or application developers. Unlike components, however, service providers own the platforms on which services execute, controlling service invocation through SOAP messages. Users of a component-based application at least own instances of component binaries and normally have control over their execution environment. Once purchased, component users can control component change by, for example, choosing whether to install upgrades.

With service-based systems, users of third-party services have no ownership of the software components that deliver the services. Service providers may evolve services and while the service they are contracted to deliver may be

subject to (for example) service level agreements, the user may have no other guarantee that the service provided will be the same between two invocations. A component user can verify a component by subjecting it to testing because they own the binaries and the execution environment. Depending on business model, this is likely to be impractical for many service users who will have to rely upon published service specifications and adopt alternative verification strategies.

Because of these issues, service specification requires substantial work in order for service providers to deliver services usable by users and applications. In particular, existing standards and protocols have serious limitations that inhibit service selection in a world in which there are thousands of service providers offering potentially useful services across a range of application domains with widely differing requirements. Similarly, while service developers can employ existing testing practices, traditional testing techniques are simply not available to service users. Unless addressed, this is likely to lead to severe problems such as feature interaction, poor quality of service and compromised dependability.

In this paper, we describe the approach taken by the *Service Centric System Engineering* (SeCSE) project (IST 511680) for service specification that is based on the notion of facets. Section 2 introduces facets, section 3 illustrates these with an example facet designed for early service discovery and section 4 describes the structure of faceted specification and our facet management tool.

2. Faceted Specification

The service specification mechanism proposed addresses a number of important challenges. These relate to:

- Providing sufficient information for the added-value techniques proposed by SeCSE to be feasible;
- Navigating through the plethora of competing standards and proposals for standards;
- Avoiding an overly prescriptive approach to how service developers/providers specify the services that they offer and being robust to the choices that they make;
- Maintaining compatibility with standard approaches (as far as possible) and being flexible enough to be able to respond to changes that occur in the near future.

The result is a specification model that is primarily based on the notion of facets. In this model a service is specified using

- A standard UDDI specification. This is needed to permit the most fundamental level of registry-based discovery using the established UDDI protocol.
- (Optionally) a set of SeCSE facets. Facets represent SeCSE added-value for service consumers and assume the ability to query and retrieve them from a SeCSE (or SeCSE-compliant) registry.

The requirement that a service have a UDDI specification ensures that SeCSE services maintain a degree of compatibility with other approaches whilst the inclusion of the facet structure provides a mechanism for a much richer form of description/specification in order to exploit the SeCSE techniques being developed. Facets:

- Are projections over one or more service properties (e.g. binding, operation signature);
- Specify those properties in one or more languages (e.g. English, WSDL, OCL);
- Should be cohesive. They should address a particular SeCSE registry query type which itself supports some purposeful task of a service consumer (e.g. early service discovery).

Although facets bear a passing resemblance to viewpoints, they serve a very different purpose. Whereas facets are a projection over service functionality, used to achieve specification for a purpose, viewpoints are a projection over requirements. The key advantages of facets as a mechanism for modelling service specifications are:

- The set of facets available to describe a service are represented explicitly. Hence a service consumer can tell whether the information required to evaluate the service is available.
- The languages used to specify each facet property is represented explicitly. Hence, the ability of the human or tool performing the discovery to interpret the specification can be determined.
- New facets can be added to a service specification dynamically and new facet specifications can be added as they become available. Thus a facet may be represented using a number of different specifications if doing so results in clear benefits to service providers and service users.
- The set of facet types is not fixed. New facet types can be defined and facets can accommodate any new notation that is encodable using XML.

3. Service discovery

The demands of service discovery illustrate the motivation for a faceted approach to service specification, especially when considered in all of its different flavours. Early service discovery, or requirements driven service discovery is an activity that takes place in parallel with requirements definition, with service availability informing

Table 1. Specification type and associated standards.

Specification Type	Technology	Comments
Service Description	Text	
	UDDI [1]	UDDI allows structured textual descriptions of services to be provided. A mature standard.
	OWL-S [2]	OWL-S uses structured textual descriptions and ontologies. Most mature initiative in the field of service ontologies.
	WSMO [3]	Alternative to OWL-S, but less mature
	UML [4]	Can provide component based descriptions of services
Service Signature	Text	
	WSDL [5]	The standard technology for specifying a services interface. A mature standard.
	UML	UML allows interfaces and ports to be specified
	WSDL-S [6]	Proposed extension to WSDL
Operation Semantics	Text	
	OWL-S	Allows the specification of pre-/post- conditions and effects
	UML/OCL [7]	UML can incorporate OCL that can be used to specify pre-/post- conditions
	WSMO	Allows the specification of pre-/post- conditions and effects
	WSDL-S	WSDL extension allows pre- and post- conditions to be applied to operations
Behavioural specification	Text	
	UML	UML supports the representation of states and the modelling of state transitions
	OWL-S	OWL-S allows states to be modelled as well as process modelling
	OpenModel [8]	Comprehensive behavioural modelling. Not widely used
Quality of Service (currently there exists no standard for specifying the QoS attributes of a web service)	Text	
	UX [9]	}
	UDDIe [10]	} - QoS extensions for UDDI
	SWSQL [11]	}
	WSOL [12]	}
	E QoS [13]	}
	Extensible QoS Model [14]	} - QoS extensions for WSDL
	WS-QoS [15]	}
	WSLA [16]	Developed by IBM allows specification and monitoring of QoS with the use of electronic SLA's
	WSML [17]	A language developed by HP for expressing SLA's

the requirements process. Architecture time service discovery, by contrast is a process more constrained by the architectural demands of the system being developed, while runtime service discovery is a process of identifying services matching the runtime requirements of a system already in operation. Each of these processes place their own requirements on the nature of a service's specification. Other processes also create demands on service specification; for example, successful composition is only possible with adequate binding information.

Table 1 lists some aspects of services, or specification types, we are interested in modeling along with technologies currently used or proposed for describing them. Each proposed technology addresses one of more of the needs of each type of service specification. However, it is clear that it is impossible to make use of all of the proposed techniques. The faceted approach we propose is a mechanism that allows service developers/providers to specify their services in the ways they deem will be most useful for their potential customers, in a manner that makes any type of specification amenable to inclusion, but always optionally.

Requirements driven service discovery is the point of interaction between requirements and service specification, and provides a good illustration of the efficacy of specification facets. In an ideal world, a potential service user would express their requirements in an appropriate form, and upon submitting them to some form of service repository, or service broker, would discover a service that met those requirements. Clearly, this is unrealistic in the real world. From a software engineering perspective, there are issues of prioritising requirements according to available resources and these are fundamentally different from those that occur in traditional software engineering. The form of services and the costing models associated with service use are governed by the service provider, not the service user.

However, more profound is the very notion that requirements can be expressed in a form that will always lead to successful service discovery. In the absence of some ideal form of common language that can be used to provide "total" service specification (ignoring the theoretical issue of whether this could ever even be possible) and appropriate expression of requirements, we are left with the notion of both service specification and requirements expression being constrained by the specific context and purpose. An important element of this context is human intelligibility: whilst service discovery will involve machine interaction with a service registry (we envisage querying a UDDI registry along with some form of extended registry designed to support the use of specification facets), we also assume that the process will involve a human developer analyzing the service descriptions. For this reason, we consider a natural language element to be a vital part of a service description geared to requirements driven service discovery.

To maximize the consistency and flexibility of faceted specifications, we propose that higher-level facets may comprise more than one specification. Thus, if developments in OWL-S make such specifications standard in the future, a service description facet currently containing a natural language specification may optionally also contain an OWL-S specification.

The sort of natural language specification we currently propose contains structured elements such as, for example:

- Name
- Version
- Description of operations
- Business assumptions
- Technical assumptions
- Pre and post-conditions
- Etc.

The rationale for having a structured natural language description is that it increases the ability to carry out automatic analysis.

4. Service specification structure and Facet Management Tool

Service specification is a key element of the SeCSE project, whose aim is to create methods, tools and techniques for system integrators and service providers and to support the cost-effective development and use of dependable services and service-centric applications. An early activity in SeCSE has been to define a conceptual model to show how services and their surrounding context. The part of that conceptual model that deals with service specification is presented below.

4.1. Specification conceptual model

Figure 1 illustrates the conceptual model for our service

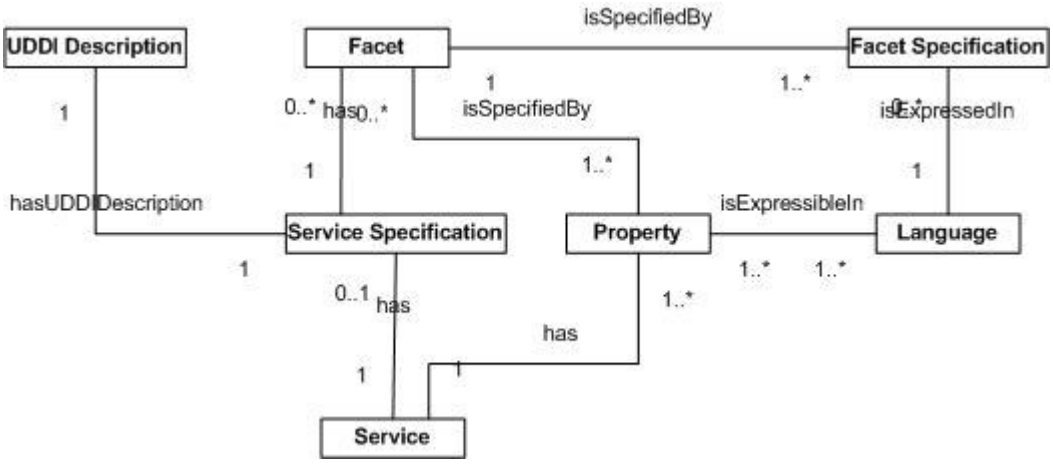

Figure 1. Specification conceptual model

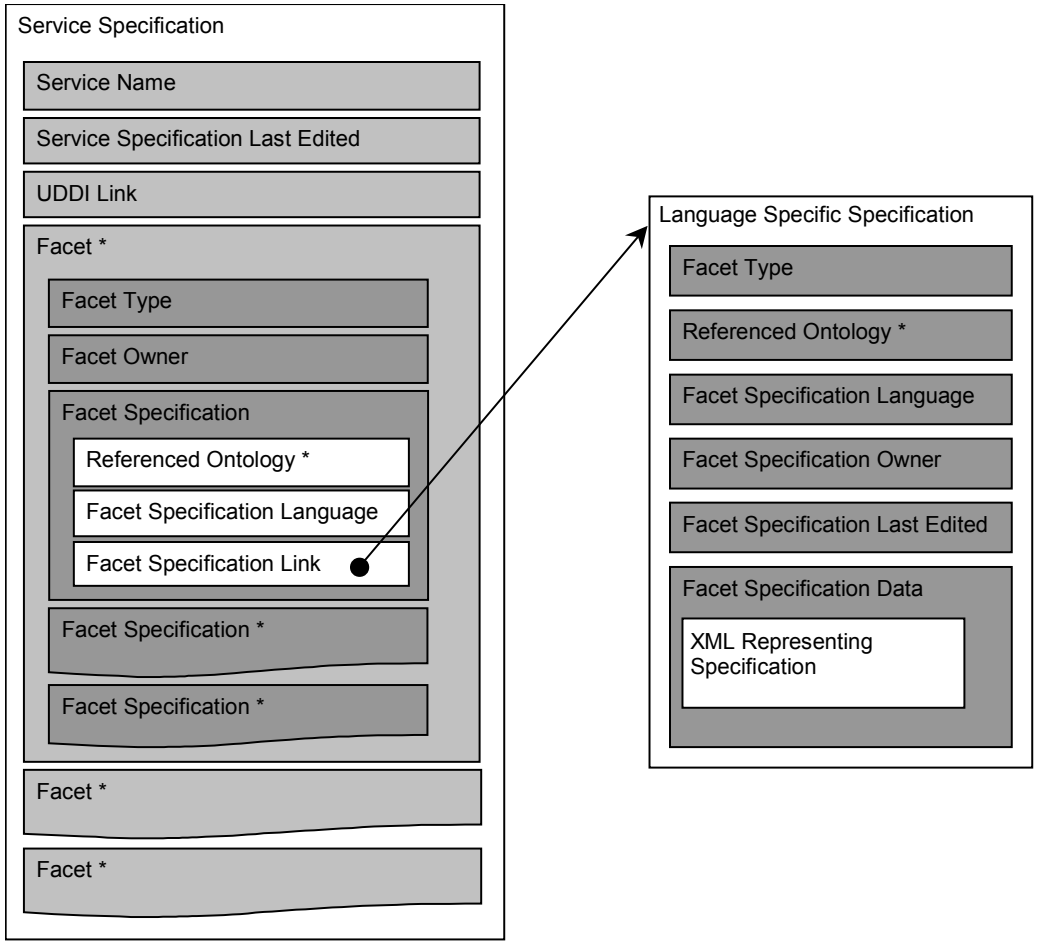

Figure 2. Proposed two-tier specification file structure (* denotes an optional element)

specification approach. The important elements in the model are:

- *Facet* - A Facet is a projection over one or more Service Properties that provides a partial description of a service.
- *Facet Specification* - A Facet Specification is a description of the Service Properties described by the Facet in a given Language.
- *Language* - A Language is a textual medium for communicating Service information. In most cases a Language is expected to be either a natural language (such as English, Italian or Spanish) or XML-based (such as WSDL).
- *Property* - A Property is some characteristic of a Service. A Property may be described by a Service Specification. A Property may be behavioural or structural. Behaviour properties include functional or non-functional characteristics of a Service. Structural properties represent those characteristics of a service concerning the Service architecture and the data manipulated or communicated by the service.
- *Service* - Software entity that performs one or more Operations. It is developed by a Service Developer and has at least one Service Description.
- *Service Specification* - A Service Specification represents the portion of a Service Description provided by a Service Interface Developer or a Service Provider.

Figure 2 illustrates how these elements are combined into a two-tier file structure.

4.2. Supported facets

Although a key element of the facet-based specification model is that it is extendible, within the SeCSE project we have identified different types of specification that are needed to support processes being developed to facilitate system development. We propose, therefore, to support the following facet types:

- **Signature**: this provides the same information as a WSDL specification. It is expected that WSDL will be the language used to specify this facet.
- **Service Description**: intended primarily to support early service discovery. This requires the definition of

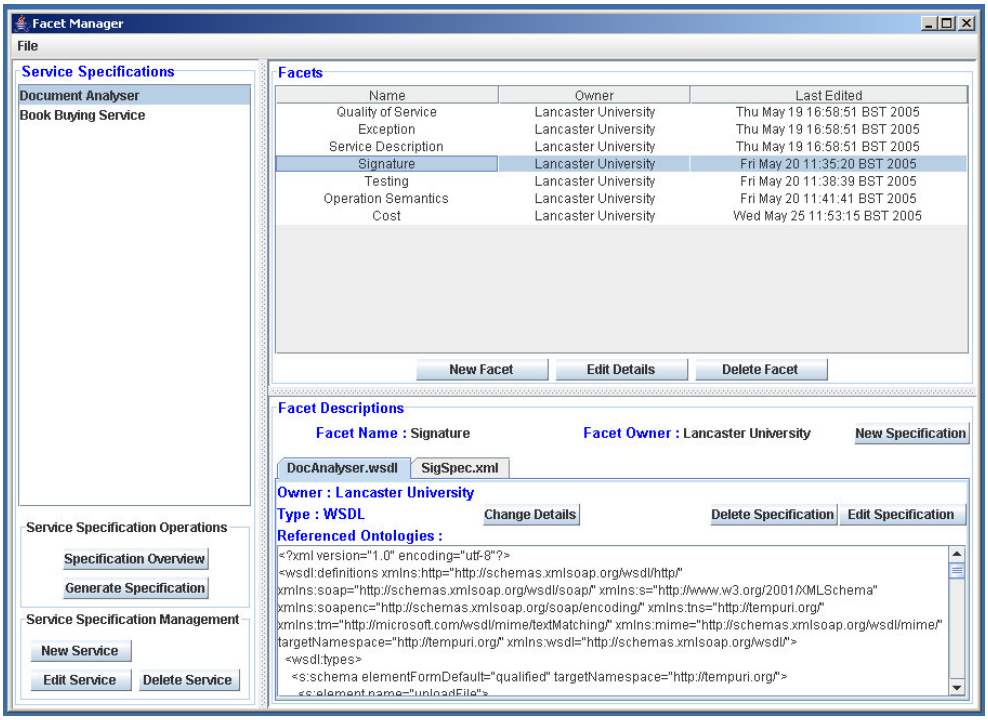

Figure 3. The Facet Management Tool.

a structured natural language description designed to support shallow semantic matching.

- **Operational Semantics**: intended to augment the signature specification of service operations that forms the current state-of-the-practice with information permitting the semantics of operations to be understood and evaluated by service consumers.
- **Exception**: intended to permit service failure behaviour to be described.
- **QoS**: intended to permit service quality of service to be described. QoS denotes a range of non-functional properties. QoS is currently poorly understood in the service engineering context, however, and it is unclear how we will specify QoS properties, what QoS properties are important, what metrics are most appropriate and the role of service level agreements (SLAs).
- **Cost**: intended to make cost information available for service consumers. A text based specification or an ontology-based specification may be used for this task.
- **Testing**: intended to make test data available to service consumers.

4.3. Facet Management Tool

It was always envisioned that the proposed faceted specification approach would be complemented by tool support. Consequently we have developed a Facet Management Tool that can be used by service providers to help create, specify and manage the facets within a service specification.

The current version of the Java-based tool supports:

- *Facet Management* - the tool allows facets to be created and edited. It is also possible to customise the tool so that a default set of facets are used whenever a new service specification is created
- *Specification Management* - the tool allows the user to import specifications into facets. The tool is able to handle natural language specifications and those that are a derivative of XML.
- *Editor Management* - the tool allows the user to assign 3rd party editors to different specification types, which can then be launched directly from the tool. For example if the user has a preferred WSDL editor, then they can link this into the tool.
- *Facet Templates* - although the Facet Management Tool is flexible enough to work with different types of specifications and editors, it is also able to support specification templates for individual facets. These templates represent a 'set specification structure' for the facet and in particular allow for better integration with other SeCSE tools.
- *Facet File Generation* - once the user has built up there faceted specification they can use the tool to

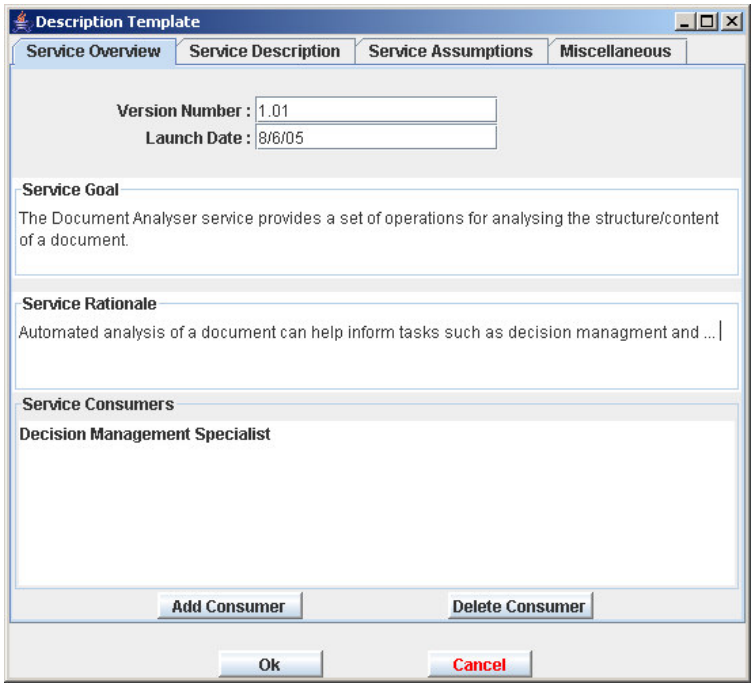

Figure 4. Using a template to capture service description information.

automatically generate XML based facet specification files. These files can then be incorporated within a SeCSE (or SeCSE-compliant) registry.

Figure 3 shows the Facet Management Tool in use and illustrates a faceted specification that has been created for a Document Analyser service. In the top right a table displays the facets that exist within the specification, below this is a preview pane that can be used to view the specifications within a selected facet. Third party editors can be launched from within the tool to allow for the editing of individual specifications, and specifications can also be imported by simply dragging a file into the window. Figure 4 shows a Facet Template being used for the Service Description facet.

The Facet Management Tool is an ongoing development and will evolve alongside the facet specification approach. It is intended that future versions will include support for:

- *UDDI generation* - the tool will provide a mechanism that will allow the user to create a UDDI specification for the service. This UDDI specification will include links to the faceted specification, if relevant.
- *Version management* - the tool will provide simple support for versioning the specifications that it manages.
- *Improved integration* - various integration aspects of the tool still need to be improved. In particular, ensuring that imported specifications are syntactically and semantically correct (where possible), and improving integration with the registries where the specifications will be stored.

5. Open issues

The most important contribution that the proposed scheme makes is that is provides a mechanism for dealing with uncertainty about the decisions made by service providers and service users and the plethora of specifications types and standards. There are, however, a number of open issues. In implementing service specification based on this scheme, the process and technological infrastructures must address variability in the following:

- *The set of facet types*. It must be possible to accommodate new facet types as service consumers' requirements evolve and greater maturity is achieved in different areas. The existing scheme was selected from a number suggested by SeCSE project partners – required to support the processes they propose to support service centric system development. Furthermore, they were chosen to be inclusive of potentially many different types of specification. However, we can already foresee the possibility of including a facet to describe "service context" information, for example.
- *The set of recognized service specification languages*. Since new service specification languages are still

emerging and existing ones still evolving, the set of languages available to a specification author will not be stable. However, service consumers wishing to evaluate a service specification need to be able to establish whether their tool (if any) is capable of interpreting the specification. Hence, an indicator of language used is needed but the mechanism used to interpret it needs to accommodate potentially arbitrary choices of language and their versions. For example, WSDL is recognised a relatively mature standard and is a key inclusion in the specification types for service's signature. If a new scheme is proposed and becomes the preferred standard, it must not only be supported, but assumptions about the availability of a WSDL specification should probably be relaxed.

Other open issues arise from the relative immaturity of the whole service-oriented computing field. A key aim of the SeCSE project is to provide support for dynamic reconfiguration at runtime when failures occur in running systems. How the requirements of a system can be mapped on to the capabilities of available service is something that can only be speculated about at the present time.

6. Conclusions

This paper has presented a faceted approach to service specification, which seeks to provide an extensible structure for managing the different formalisms that can be used to describe services. A facet represents a projection over one or more service properties which, in turn, can be specified using one or more languages.

The faceted approach is an ongoing development within the SeCSE project and will be developed further and refined as it is utilised by the user partners. Currently a number of facets have been identified that represent a set of 'standard' service properties (e.g. signature facet). For a selection of these, default specification structures have also been proposed (e.g. WSDL) – although there is no requirement for service developers/providers to follow them. It is also intended that the faceted specification approach will be closely tied-in with tool support. Development of a facet management tool is ongoing with an initial version already released. As the project progresses and we gain a better understanding of what is required for service specification, these three areas will continue to be developed and refined.

References

1. UDDI Technical White Paper, September 2000. http://www.uddi.org
2. DAML-S (2004) OWL-S 1.1. http://www.daml.org/services/owl-s/1.1/
3. Web Service Modelling Ontology. http://www.wsmo.org/.
4. W3C. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl
5. Miller, J., Verma, K., Rajasekaran, P., Sheth, A., Aggarwal, R., Sivashanmugam, K. (2004) WSDL-S: A Proposal to W3C WSDL 2.0 Committee. http://lsdis.cs.uga.edu/projects/WSDL-S/wsdl-s.pdf
6. Kruger, I. H., (2002) "Towards Precise Service Specification with UML and UML-RT", Proc. Workshop on Critical Systems development with UML, Dresden, Germany. http://www4.in.tum.de/~csduml02/27.pdf.
7. Hall, R., Zisman, A. (2004) "Behavioural Models as Service Descriptions", Proc. ISOC'04. New York, USA.
8. Chen, Z., Liang-Tien, C., Silverajanm B., Bu-Sung. L., (2003) "UX – An Architecture Providing QoS-Aware and Federated Support for UDDI" Proc. International Conference on Web Services (ICWS '03), Las Vegas, Nevada, USA. http://www.ntu.edu.sg/home5/PG04878518/UDDIExtension.html
9. UDDIe project. http://www.wesc.ac.uk/projects/uddie/uddie/
10. Bilgin, A., Singh, M. (2004) "A DAML-Based Repository for QoS-Aware Semantic Web Service Selection", Proc. IEEE International Conference on Web Services (ICWS'04), San Diego, Ca. USA.
11. Tosic, V., Patel, K., Pagurek, B., (2002) "WSOL - Web Service Offerings Language", Proc Workshop on Web Services, e-Business, and the Semantic Web (WES) Toronto, Canada, pp. 57-67.
12. Shen, D., Yu, G., Nie, T., Li, R., Yang, X. (2004) "Modeling QoS for Semantic Equivalent Web Services". Proc. Fifth International Conference on Web-Age Information Management (WAIM 2004), Dalian, China.
13. Liu, Y., Ngu, A., Zeng, L. (2004) "QoS Computation and Policing in Dinamic Web Services" Proc. WWW2004, New York, New York, USA.
14. Tian, M., Gramm, A., Naumowicz, T., Ritter, H., Schiller, J. (2003) "A Concept for QoS Integration in Web Services" 1st Web Services Quality Workshop (WQW 2003), Rome, Italy. www.wsqos.net
15. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R. (2003) "Web Service Level Agreement (WSLA) Language Specification". http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf
16. Web Services Modelling Language. http://www.wsmo.org/wsml/.