Using QoS for Relevance Feedback in Service Discovery: A Preliminary Empirical Investigation

Konstantinos Zachos, Neil Maiden
Centre for HCI Design, City University
London, UK
{kzachos, n.a.m.maiden}@soi.city.ac.uk

Glen Dobson, Pete Sawyer
Computing Department, Lancaster University
Lancaster, UK
{dobsong, sawyer}@comp.lancs.ac.uk

***Abstract*—Service-centric systems pose new challenges and opportunities for requirements processes and techniques. This paper describes our requirements-based service discovery tool that exploits an ontology-based quality specification mechanism to receive early feedback on candidate services that best match quality requirements. An empirical evaluation of the tool is presented that assesses the feasibility of the approach to filtering candidate services based upon quality using real-world scenarios from our industrial partners. The results reveal that commitment to a common ontology helps achieving the desired quality-based filtering.**

Keywords – Service Discovery; QoS; Quality-based filtering.

I. Developing with Web Services

Service-centric system engineering is an important emerging paradigm in which applications are constructed from reusable component services [1]. One important consequence for requirements processes is that service registries available over the Internet provide users with immediate access to elements of the solution space. In the SeCSE Project [4] we have developed tools and techniques to form and execute queries on service registries from requirements specifications.

SeCSE supports an iterative requirements discovery process as shown in Fig. 1 [5]. Requirements analysts form service queries from requirements specifications to retrieve web services compliant with the requirements. Descriptions of retrieved services are presented to analysts who use them to enable more accurate service retrieval in a cyclical retrieval and requirement refinement process.

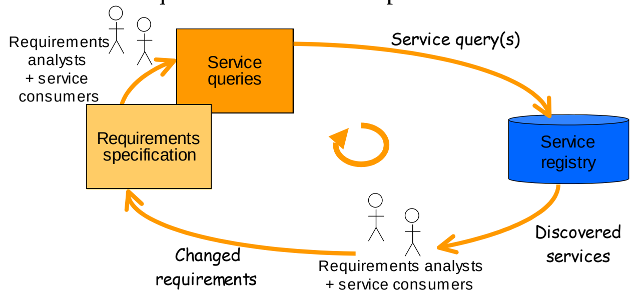

Requirements analysts + service consumers
Service queries
Service query(s)
Requirements specification
Service registry
Discovered services
Changed requirements
Requirements analysts + service consumers

Figure 1. SeCSE's Requirements Process

To ensure industrial uptake SeCSE's requirements process uses established techniques based on structured natural language. Analysts specify service-centric system behaviour with UML use case specifications and required system properties in a testable form with VOLERE shells [7]. The process extends the Rational Unified Process (RUP) [8] without mandating additional specification or service retrieval activities.

Our approach builds on Fischer et al.'s [6] observations about how design queries are incrementally improved by critiquing results from previous queries. Relevance feedback, as this is known, provides information about whether requirements can be satisfied by available services, to guide the analysts to consider alternative build, buy or lease alternatives or explore trade-offs to see whether most requirements can be met at acceptable cost by the available services.

To support this process of requirements-based service discovery we have also developed a means to represent service semantics. Relevance feedback may be used to inform analysts' formulation of requirements based on representations of service semantics held in service registries. Crucially, this may include information about non-functional service characteristics. Of particular interest are the set of characteristics that broadly fall into the area of Quality of Service (QoS). With SeCSE it is possible to get an early indication of the quality of available services and therefore the potential quality achievable in the final application. Along with information about discovered services' functionality, QoS information can be fed back into the requirements in order to help formulate requirements for acceptable QoS characteristics for the future system.

QoS has been identified [13] as a key element in the wider uptake of web services, particularly in a competitive market in third-party services [2]. Here, the consumer has no control over the service provision and implementation, and so assessing QoS is vital in establishing trust in a service. It is therefore desirable for the service consumer to be able to clearly state QoS requirements and identify the level of service compliance with those requirements. In recent years, we have reported new tools and techniques to increase requirements completeness from retrieved web services based on functional requirements [e.g. [14],[16],[17]]. In this paper we present new techniques to match non-functional requirements to service qualities during service selection using an ontology.

There are many QoS properties and for any property, there are often several metrics with which it may be expressed. This creates problems for QoS-informed service discovery since the analyst and service provider may express required and provided QoS properties in different ways.

Where different metrics are used false negatives will result from simple syntactic matching. As we have reported previously [15], a key feature of our tools is that these tools can tolerate such inconsistencies. When we last reported on this work, however, we had not yet performed an empirical evaluation of our work. To remedy this, this paper reports results from an industrial evaluation that was designed to answer our central hypothesis, that *the SeCSE service specification and SeCSE's requirements-based discovery tools are tolerant of mismatches between how a service provider specifies their services' QoS properties, and how a service consumer specifies their QoS requirements.* In other words, the analyst is insulated from needing to know the metrics and units used by different service providers, provided they (the analyst) use a recognized metric/metrics and unit/units when they specify their requirement.

Section 2 describes SeCSE's requirements-based service discovery environment and the QoSOnt ontology. Section 3 introduces the evaluation method, and Section 4 reports results from the evaluation. Section 5 uses this data to answer the central hypothesis and discusses threats to validity of the results. The paper ends with implications of the results for iterative requirements-based service discovery processes and service discovery algorithms to support this process.

II. DISCOVERING AND SELECTING SERVICES USING QOS REQUIREMENTS

To support SeCSE's requirements process we implemented the SeCSE service specification and discovery environment. It has 4 main components: (i) Service Registries – a federated and heterogeneous mechanism for storing service descriptions along with the Service Specification Tool with which service providers can populate the Service Registries; (ii) the UCaRE Requirements Component, which supports web-enabled specification of requirements and use cases, and formulation of service queries from these specifications; (iii) the Service Querying Component, which uses service queries to discover web services with different types of similarity; and (iv) the Service Explorer Component that displays the descriptions of retrieved services to enable analysts to understand, select between and use these services to discover new requirements.

A. The Test Scenario

To clarify our discussion we will present the test scenario for the evaluation. This has been taken from a system developed by SeCSE's industrial partners that integrates various services into a client within an in-car device. The services include map/point of interest services; weather services; car park booking services; telecoms services, and logistical services [3]. The use case that has been used for the purpose of this evaluation to test the hypothesis described in the introduction is partly shown in Table 1 –*Acquiring Weather Information* use case. Table 2 defines two simple associated requirements. The first, a functional requirement (FR61), specifies what the service shall do, and the second, the non-functional requirement (AR7), specifies desirable qualities of the service.

TABLE I. PART OF ACQUIRING WEATHER INFORMATION USE CASE

Name	Acquiring weather information
Précis	A driver is driving his car. The driver requests weather forecast information for a selected destination using the car's on-board weather forecast information service. The service retrieves information about weather forecasts for the selected destination based on the estimated arrival time. The service displays the information to the driver.
Actors	Driver, On-board weather forecast information service
Problem Statement	The driver needs information that is relevant and useful to him/her and his/her location.

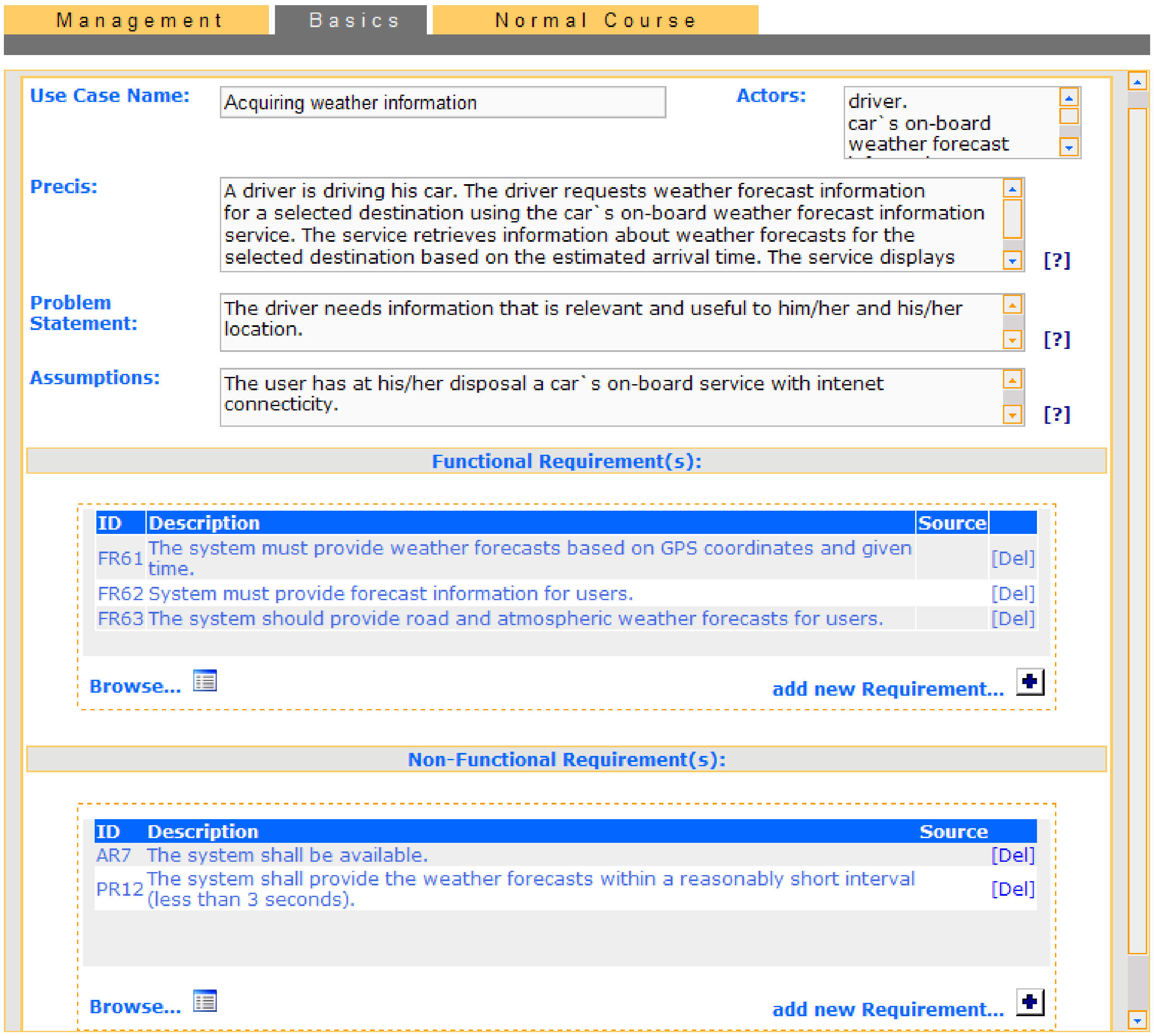
TABLE II. REQUIREMENTS ON THE ON-BOARD WEATHER FORECAST INFORMATION SYSTEM

FR61: The system shall provide weather forecasts based on GPS coordinates and estimated arrival time.
AR7: The system must be available remotely in order to be integrated into the in-car device.

B. The Service Registries

The SeCSE environment discovers web services specified within registries that link to service implementations that applications invoke. However, registries such as UDDI (Universal Description, Discovery and Integration) are inadequate for retrieving services using semantic criteria such as QoS and exception handling. To counter this shortcoming SeCSE has defined an extensible *faceted* specification mechanism [9]. Facets are projections over service properties and serve to partition a service specification according to the properties that the service provider wishes to make public. There are currently 10 facet types that can be used to describe a service. Those that are relevant here are *signature*, which mimics the WSDL (Web Services Description Language) specification of service binding and signature information and is always needed, a *description* facet which provides a brief natural language service description designed to aid discovery and selection by SeCSE's discovery tools, and the *QoS* facet used to describe a service's QoS characteristics. Although we commonly assume that the service provider is responsible for providing service facets, some could be provided or authenticated by third parties. This applies particularly to the QoS facet where the information may be the service's QoS that is claimed by the service provider, or it may represent data on the service's actual QoS as monitored by a third party.

Service discovery in SeCSE uses the description and QoS facets to retrieve web services [[15],[17]]. The role of the QoS facet in service discovery is to refine selection of services discovered using the description facet. Retrieved service descriptions are presented to requirements analysts and service consumers to enable them to select the most appropriate services based upon their QoS requirements. A full description of the part of the SeCSE conceptual model that relates to QoS and the QoS facet is provided in [15].



Use Case Name:	Acquiring weather information
Actors:	driver. car`s on-board weather forecast
Precis:	A driver is driving his car. The driver requests weather forecast information for a selected destination using the car`s on-board weather forecast information service. The service retrieves information about weather forecasts for the selected destination based on the estimated arrival time. The service displays
Problem Statement:	The driver needs information that is relevant and useful to him/her and his/her location.
Assumptions:	The user has at his/her disposal a car`s on-board service with intenet connecticity.

Functional Requirement(s):

ID	Description	Source	
FR61	The system must provide weather forecasts based on GPS coordinates and given time.		[Del]
FR62	System must provide forecast information for users.		[Del]
FR63	The system should provide road and atmospheric weather forecasts for users.		[Del]

Non-Functional Requirement(s):

ID	Description	Source	
AR7	The system shall be available.		[Del]
PR12	The system shall provide the weather forecasts within a reasonably short interval (less than 3 seconds).		[Del]

Figure 2. Example use case specification specified in UCaRE

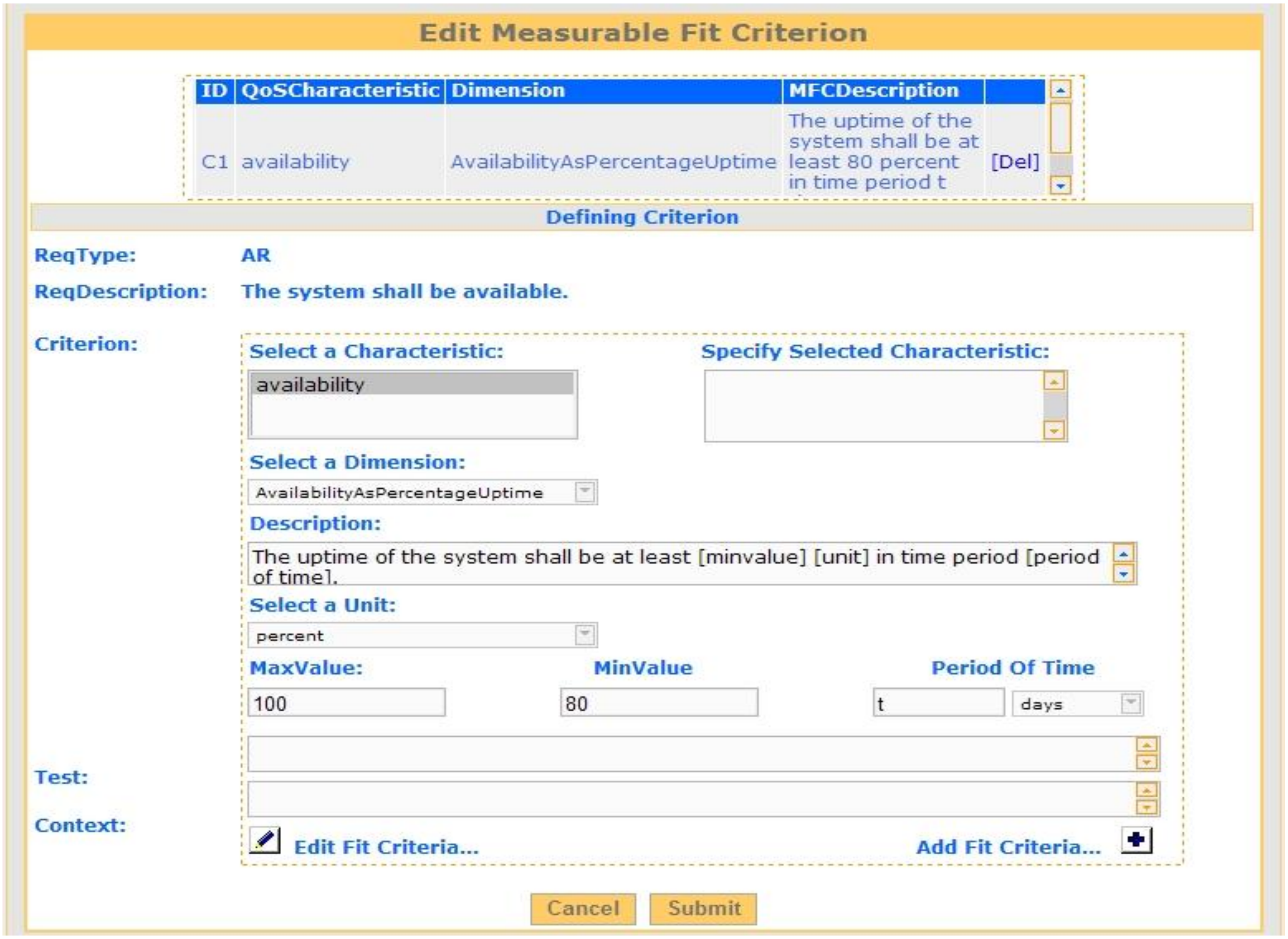
C. *The UCaRE Requirement Component*

Analysts express requirements for new applications using UCaRE [18], a web-based .NET application depicted in Fig. 2. A requirements analyst manages requirements and use cases through a web client.

At the start of SeCSE's requirements process, analysts work with future service consumers to develop simple use case précis that describe the required behaviour of the service-centric application. Table 1 includes the précis for the *Acquiring Weather Information* use case.

UCaRE supports the analyst in using the VOLERE shell to specify requirements such as AR7. VOLERE enables the analyst to specify the requirement's type, rationale, source, owner and importance scores. For non-functional requirements, UCaRE also supports the specification of measurable fit criteria (MFC) that are essential for selecting between discovered services on QoS criteria. MFCs are quantified goals that describe in detail how the system must behave in order to be deemed to have satisfied the requirement. While the description of the requirement is written in the language of the stakeholders, the MFC is written in a precise quantified manner so that solutions can be tested against the requirement. By stating requirements with measurable fit criteria that are aligned with the metrics of QoSOnt it is possible to use these to directly compare against the QoS facet. Fig. 3 shows the measurable fit criterion for the availability requirement AR7 that was used in the evaluation to filter the resulting list of candidate services relating to QoS.

The availability requirement in Fig. 3 shows that even if there was a broad agreement on the important characteristics of quality it is highly likely that a degree of translation would be required to allow different parties to compare QoS. For instance, even if exactly the same metric is used (say, mean time to repair) by both analyst and service provider it may be stated in any unit of time. Simple syntactic matching would lead to many false negatives in this case. Consider that the requirement AR7 of at least 80% availability (as percentage uptime over some period) has been specified for a weather service. Imagine that a specification of a candidate service includes a QoS property with a mean time between failure of 42 days and a mean time to repair of 1 hour. Without conversion capabilities the service's QoS property would not match the requirement and would give a false negative. However, with the integration of QoSOnt (section 2.5) mismatches of this kind are avoided. The metrics are first converted to compatible units (42 days = 1008 hours). The availability metric is then computed from these two component metrics 1008/(1 + 1008), allowing the inference that the availability of the service is 99.9%, thus that it meets the requirement.



ID	QoSCharacteristic	Dimension	MFCDescription	
C1	availability	AvailabilityAsPercentageUptime	The uptime of the system shall be at least 80 percent in time period t	[Del]

ReqType: AR
ReqDescription: The system shall be available.
Criterion:
Select a Characteristic: availability
Specify Selected Characteristic:
Select a Dimension: AvailabilityAsPercentageUptime
Description: The uptime of the system shall be at least [minvalue] [unit] in time period [period of time].
Select a Unit: percent
MaxValue: 100
MinValue 80
Period Of Time t days
Test:
Context:

Figure 3. Measurable fit criterion for the availability requirement AR7

D. *Service Querying and Service Explorer Components*

In the SeCSE service discovery environment analysts can manipulate specified use cases and requirements to generate service queries. These are fired at service registries with the EDDiE service discovery engine [17] to retrieve web services from the same domain as the current problem. EDDiE implements advanced term disambiguation and query expansion algorithms to add different terms with similar meanings to the query using the WordNet online lexicon, thus increasing the number of web services retrieved from the registries.

The service discovery environment presents retrieved services to analysts and service consumers in the Service Explorer Component (as shown in Fig. 4). It orders services that attain a minimum threshold of relevance in a ranked order. The analyst can click on each service to view all properties of the service description facet and view each property next to its corresponding use case and requirement property to enable comparison. The Service Explorer component also provides functionality with which to select between retrieved services and refine the services available based on a service's QoS information.

E. *QoSOnt as a means to filter and select candidate services*

The SeCSE QoS Facet makes use of the QoS Ontology, QoSOnt [10] that implements the OMG model of QoS [12] using the OWL [11] ontology language. An ontology is a description of the concepts which exist in some domain and the relationships between them. QoSOnt does not just provide a set of terms but provides a machine interpretable model of QoS knowledge. The structure of QoSOnt, which is really three linked ontologies that model QoS properties, QoS metrics and units, is further described in [10].

Returning to our consideration of comparing the quality of candidate services during the requirements phase; determining whether a service supports certain metrics is of limited use without being able to compare the analyst's requirements against the services' capabilities. This is where the use of QoSOnt provides great advantages for this usage scenario.

It is through commitment to the same ontology (QoSOnt) in the specification tool, registry and discovery tool that comparison between service quality is made possible. UCaRE prompts the user to specify MFCs using terms directly from the ontology. This means that if the underlying ontology evolves then these changes will automatically be reflected in the tool. The SeCSE specification tool [9] uses the same approach, meaning that service QoS requirements are stated in compatible terms.

In practice this does mean that some level of agreement on a QoS vocabulary is a pre-requisite. However the use of an ontology easily allows the use of multiple synonyms and the expression of new concepts in terms of other existing concepts.

III. EVALUATION METHOD

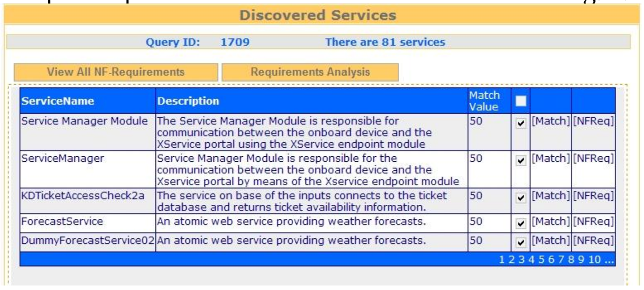
We made use of a scenario from SeCSE's industrial partners in the design of our experiment. For each service involved in the scenario the industrial partners specified and published a QoS facet and a description facet. They then described the corresponding use case using the UCaRE tool. With these artifacts in place it was possible to attempt requirements-based service discovery, i.e. matching the use case requirements against the service descriptions in the SeCSE registry.

In order to evaluate how useful this was we needed more data however. In particular, in order to assess whether the conversion capabilities provided by QoSOnt were allowing a greater number of candidate services to be compared, we needed more examples of QoS facets. At the same time, in order to be evaluated using UCaRE the description facet of the corresponding services had to remain largely the same. To achieve this we took the specifications of industrial partners and created dummy service specifications in which the description was unaltered, but the QoS facet was varied. In the dummy services the units of various metrics were altered to compatible units assuming a uniform random distribution. Where another way of expressing metrics was known, metrics were also re-expressed randomly in the dummy versions of the services. Having populated the registry with dummy services for each original service we then re-ran the discovery and QoS-based filtering process in UCaRE.

We focused the experiment on the QoS availability characteristic. Availability is a crucial service property for many applications, and the scenarios provided by our industrial partners reflected this. Further, there are several metrics that can be used to express availability and availability can be considered a compound of some metrics that actually represent other QoS characteristics. Finally, different metrics can use different units. As we described earlier, these factors all meant that a scenario in which availability was a crucial QoS characteristic had the potential to provide a test of our hypothesis.

A. *Requirements-based Test Queries*

A service query was formed from specification elements for the Acquiring Weather Information use case, including the availability requirement AR7 that provided the basis of the service query evaluation. The service query was then used to discover services in the SeCSE registry using EDDiE. Once generated and fired at the SeCSE service registry to retrieve similar web services, the Service Explorer component presents discovered services as shown in Fig. 4.



ServiceName	Description	Match Value	☐		
Service Manager Module	The Service Manager Module is responsible for communication between the onboard device and the XService portal using the XService endpoint module	50	✓	[Match]	[NFReq]
ServiceManager	Service Manager Module is responsible for the communication between the onboard device and the Xservice portal by means of the Xservice endpoint module	50	✓	[Match]	[NFReq]
KDTicketAccessCheck2a	The service on base of the inputs connects to the ticket database and returns ticket availability information.	50	✓	[Match]	[NFReq]
ForecastService	An atomic web service providing weather forecasts.	50	✓	[Match]	[NFReq]
DummyForecastService02	An atomic web service providing weather forecasts.	50	✓	[Match]	[NFReq]

1 2 3 4 5 6 7 8 9 10 ...

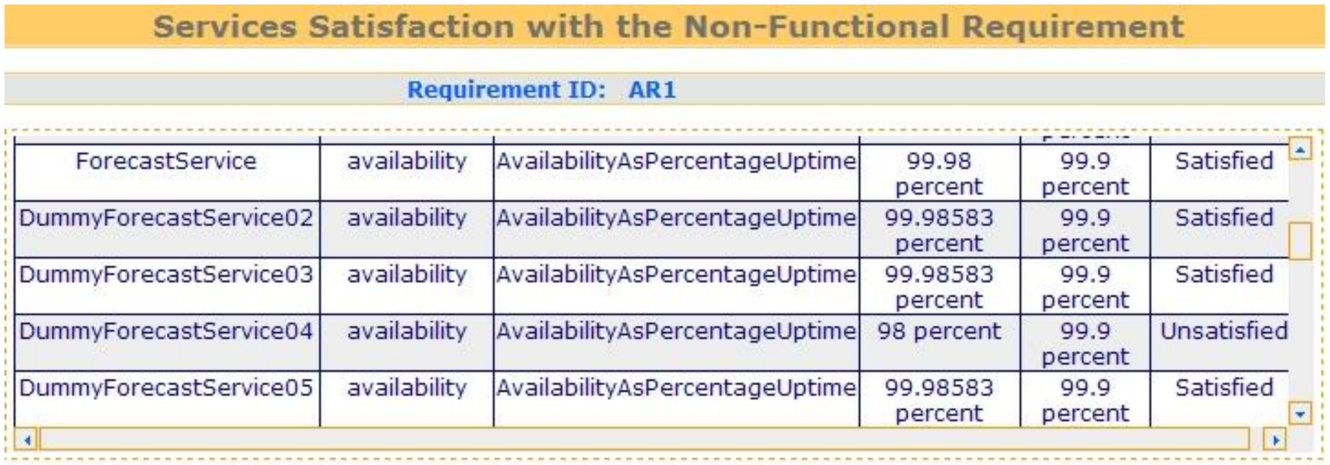
Figure 4. Discovered service descriptions shown in the Service Explorer

The Service Explorer orders services that attain a minimum threshold of relevance in a ranked order. Each service can be selected to view all properties of the service description facet and view each property next to its corresponding use case and requirement property to enable comparison.

The Service Explorer component also provides functionality with which to select between retrieved services and refine the services available based on a service's QoS information. Fig. 5 shows part of the list of services that have been evaluated based on the metrics. Satisfied indicates that the availability of a service is at least 99.9% and thus that it meets the relevant measurable fit criterion. On the other hand, Unsatisfied indicates that the availability of a service is below 99.9% and thus that it does not meet the relevant measurable fit criterion.

B. *The QoS Specifications used in the SeCSE Registry*

To evaluate whether QoSOnt can provide benefits to the discovery process, we seeded the SeCSE repository with 14 dummy specifications. These represented clones of the real weather forecast service, ForecastService, which acted as the baseline. Each clone specified its Availability QoS properties



Services Satisfaction with the Non-Functional Requirement

Requirement ID: AR1

ForecastService	availability	AvailabilityAsPercentageUptime	99.98 percent	99.9 percent	Satisfied
DummyForecastService02	availability	AvailabilityAsPercentageUptime	99.98583 percent	99.9 percent	Satisfied
DummyForecastService03	availability	AvailabilityAsPercentageUptime	99.98583 percent	99.9 percent	Satisfied
DummyForecastService04	availability	AvailabilityAsPercentageUptime	98 percent	99.9 percent	Unsatisfied
DummyForecastService05	availability	AvailabilityAsPercentageUptime	99.98583 percent	99.9 percent	Satisfied

Figure 5. Discovered services evaluated based on QoS

differently from the baseline. As summarised in the following table, in some cases this involved the use of different units, in others the use of different metrics, i.e. percentage uptime, Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR).

Seven of the 14 dummy specifications used different QoS metrics to the baseline specification. Instead of directly specifying an Availability metric, the MTBF and MTTR properties were specified. QoSOnt can infer an availability measurement by combining other metrics, and the purpose of these seven specifications was to evaluate this ability.

Seven of the dummy specifications also used different units in order to specify the metrics. QoSOnt is able to convert between different units (for example, hours and minutes), and these specifications would help evaluate this. The baseline and dummy specifications are summarized in Table 3 below.

IV. Evaluation Results

Table 3 shows the expected outcome of performing a QoSOnt supported discovery activity on these specifications, based on the use case described in section 2.1 (99.9% availability). Table 3 also shows the converted availability value that QoSOnt should infer for the specifications that do not directly specify their availability. The final two columns show the expected and actual result from applying QoSOnt and the service discovery tools to the registry containing the set of service descriptions.

As shown in Table 3, by using the conversion functionality provided within QoSOnt UCaRE/EDDiE was able to determine a %ageUptime metric for 13 of the 15 Weather Forecast services. Seven of these services were found to satisfy the required Availability measurement as described within the use case. Dummy services 2, 5, 6, 7, 9, 11 and 12 all used different metrics and units, and QoSOnt successfully performed conversions to infer availability values for them. Dummy services 8 and 10 were not returned. In the case of number 8 this was because it only contained one metric, MTBFs, and this on its own was not enough to infer an availability value (MTTR being required as well). Dummy service 10, on the other hand, contained no metrics that could be used to derive an availability value.

We re-ran the experiment without support from QoSOnt. Here, only syntactic matching was possible and only 5 results were obtained. All 5 were for service descriptions that used the percentage uptime metric: the baseline service and dummy services 4, 13, 14 and 15. The other services that satisfied the requirement but expressed using a different metric (2, 3, 5 and 6) were not found. Overall, in this experiment, by not using QoSOnt only 43% of the service specifications that would actually satisfy the use case were discovered.

V. Hypothesis Revisited

We used results and data from the evaluation to demonstrate QoSOnt's ability to convert between metrics and units for any (QoS characteristic, metric, unit) tuple that is known to it. Hence, we have provided evidence to support our hypothesis, that *the SeCSE service specification and SeCSE's requirements-based discovery tools are tolerant of mismatches between how a service provider specifies their services' QoS properties, and how a service consumer specifies their QoS requirements*. Of course, this is only true where both service consumer and service provider use characteristics, metrics and units known to the ontology. However, integration of QoSOnt in SeCSE's service specification and service discovery tools make consistency easy to achieve in a way that is transparent to both actors. It is this transparency that is crucial if the vision of an open service marketplace, with all the diversity of practice that implies, is ever to be realised.

Clearly there are threats to results validity. One threat to the conclusion validity of the evaluation results is the sample size – 1 service query from 1 use case specification with 1 quality type fired at 1 registry. However the current small body of research into matching non-functional requirements to service qualities during service selection led us to run a formative-predictive evaluation to generate a first set of results to provide a framework and focus for more subsequent rigorous evaluation.

VI. CONCLUSIONS

In this paper we have described an empirical evaluation that assessed the feasibility of an approach to filtering candidate services based upon QoS. The approach in question relied upon the use of the QoSOnt ontology both in the QoS facets used to specify services and in the measurable fit criteria stated in service requirements. We have demonstrated that commitment to a common ontology did aid in achieving the desired QoS-based filtering. We have also begun to demonstrate the wider advantages of the use of such ontology. In particular the supporting conversion rules were found to avoid a number of cases, which would have produced false negative matches.

Our experiment showed that the number of comparisons, which were made possible only with conversion capabilities, could potentially be significant. In practice it is not clear how the use of different metrics and units would vary across the population, but in the absence of such knowledge we believe that our experiment gives us a first approximation of the effects of conversion in service selection.

Although more direct evaluations are needed to answer questions like ‘Do the QoS facet and QoSOnt assist QoS-critical service discovery?’ – evaluations that are planned to be undertaken in the near future – the results presented in this paper support our hypothesis that the SeCSE specification and requirements-based service discovery mechanisms are tolerant of mismatches between how a service provider specifies their services' QoS properties, and how a service consumer specifies their QoS requirements. By implementing the OMG model of QoS [12] in QoSOnt we have tried to ensure coverage of recognised QoS properties. New QoS properties may emerge that have particular utility for service-centric systems. QoSOnt will need to evolve as this happens and has been released as open source to help encourage a user community to play its role in maintaining its relevance to service-centric systems engineering.

REFERENCES

[1] I. Sommerville. “Service-Oriented Engineering” in Software Engineering, 8th Edition, Pearson Education, 2006, 743-770.

[2] J. Bloomburg, “Competitive SOA”, ZapFlash, 2007.

[3] SeCSE, “Enriched mixed scenario for final demonstration” Deliverable A6.D18, http://www.secse-project.eu/?page_id=102&page=3, (last accessed 10/11/09)

[4] EU Integrated Project 511680, “Service Centric System Engineering (SeCSE)”, http://www.secse-project.eu/, (last accessed 21/07/10)

[5] S.V. Jones, N.A.M Maiden, K. Zachos, and X. Zhu, “How Service-Centric Systems Change the Requirements Process”, Proc. REFSQ’2005 Workshop, 2005, pp.13-14

[6] G. Fischer, S. Henninger, and D. Redmiles, ‘Intertwining Query Construction and Relevance Evaluation’, Proc. CHI’91, 1991, pp. 55-62.

[7] S. Robertson and J. Robertson, Mastering the Requirements Process, Addison-Wesley-Longman, 1999.

[8] I. Jacobson, G. Booch, and J. Rumbaugh, ‘Unified Software Development Process’, Addison-Wesley-Longman, 2000.

[9] J. Walkerdine, J., Hutchinson, P. Sawyer, G. Dobson, and V. Onditi, "A Faceted Approach to Service Specification", Proc. 2nd Int’l Conf. on ICIW, Mauritius, 2007.

[10] G. Dobson, R. Lock, and I. Sommerville, “Quality of Service Requirements Specification using an Ontology” Proc. SOCCER at 13th Int’l RE Conf. (RE 05), 2005.

[11] D. L. McGuinness and F. van Harmelen (eds.) “OWL Web Ontology Language Overview”, W3C Recommendation, http://www.w3.org/TR/owl-features/, 2004, (last accessed 05/09/09)

[12] “UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics & Mechanisms”, OMG, 2004.

[13] G. Dobson and P. Sawyer: "Revisiting Ontology-Based Requirements Engineering in the Age of the Semantic Web", at Dependable RE of Computerised Systems, NPPs, 2006.

[14] K. Zachos and N.A.M. Maiden, ‘Inventing Requirements from Software: An Empirical Investigation with Web Services’, in Proc. 16th International Conference on RE, 2008.

[15] K. Zachos, G. Dobson, and P. Sawyer, 'Ontology-aided Translation in the Comparison of Candidate Service Quality', in Proceedings of SOCCER workshop at RE08, 2008.

[16] K. Zachos, N.A.M. Maiden, and R. Howells-Morris, ‘Discovering Web Services to Improve Requirements Specifications: Does It Help?’, in REFSQ, June 2008.

[17] K. Zachos, N.A.M Maiden, S.Jones, and X.Zhu, “Discovering Web Services To Specify More Complete System Requirements"’ Proc. 19th Conference on CAiSE, 2007.

[18] K. Zachos, X. Zhu, N.A.M. Maiden, and S. Jones, “Seamlessly integrating service discovery into UML requirements processes” Proc. SOSE '06, 2006, 60-66.

TABLE III. EXPECTED AND ACTUAL OUTCOMES OF THE DISCOVERY ACTIVITY

ServiceName	Specified Metrics	Specified Values	%age uptime	99.9% uptime?	
				Expected	**Expected**
ForecastService (baseline)	%age uptime	99.98 %	*99.98 %*	Yes	Yes
Dummy02	MTTR, MTBF	1 Hour, 42 weeks	99.98 %	Yes	Yes
Dummy03	MTTR, MTBF	60 Minutes, 294 Days	99.98%	Yes	Yes
Dummy04	%age uptime	98 %	*98 %*	No	No
Dummy05	MTTR, MTBF	3600 Seconds, 42 Weeks	99.98 %	Yes	Yes
Dummy06	MTTR, MTBF	58 Minutes, 42 Days	99.90%	Yes	Yes
Dummy07	MTTR, MTBF	100000 Seconds, 1 Month	99.62 %	No	No
Dummy08	MTBF	30 Days	-	No	-
Dummy09	MTTR, MTBF	4 Hours, 28 Days	99.40 %	No	No
Dummy10	None specified	-	-	No	-
Dummy11	MTTR, MTBF	1 Week, 2 Months	89.68 %	No	No
Dummy12	MTTR, MTBF	10 Days, 1 Year	97.33%	No	No
Dummy13	%age uptime	99 %	*99 %*	No	No
Dummy14	%age uptime	99.999 %	*99.999 %*	Yes	Yes
Dummy15	%age uptime	99.9 %	*99.9 %*	Yes	Yes