

Does Service Discovery Enhance Requirements Specification? A Preliminary Empirical Investigation

Konstantinos Zachos, Neil Maiden, Xiaohong Zhu and Sara Jones

City University London, Centre for HCI Design
London EC1V 0HB, UK

k.zachos@soi.city.ac.uk, n.a.m.maiden@city.ac.uk, x.zhu@soi.city.ac.uk, s.v.jones@city.ac.uk

Abstract

Service-centric systems pose new challenges and opportunities for requirements processes and techniques. This paper describes new techniques developed by the EU-funded SeCSE Integrated Project that enable service discovery during early requirements processes and exploit discovered services to enhance requirements specifications, and reports 2 preliminary evaluations of software tools by our industrial partners that implement SeCSE's processes and techniques. The first evaluation investigated the usability and functionality of UCaRE, a web-based tool for specifying requirements prior to discovering services. The second evaluation investigated the utility and usability of UCaRE combined with EDDiE, the service discovery engine, during a requirements discovery workshop. Results from both evaluations inform the design of the second versions of SeCSE's processes, techniques and tools.

1. The Changing Requirements Process

One consequence of service-centric systems is that requirements processes might change due to the availability of services. Discovering candidate services will enable analysts to increase the completeness of system requirements based on available service features. However, for this to happen, new tools and techniques are needed to form and execute service queries from incomplete requirements specifications – tools and techniques developed through research that we are undertaking in SeCSE. SeCSE's mission statement is to create new methods, tools and techniques for requirements analysts, system integrators and service providers that support the cost-effective development and use of dependable services and service-centric applications [11]. The four-year research project covers four main activity areas – service engineering, service discovery, service-centric systems engineering, and service delivery. In the remainder of the paper we describe a preliminary evaluation of new tools for discovering services that can be applied to make requirements specifications more complete and correct. The purpose of the evaluation is formative – to inform the further design of SeCSE's processes, techniques and tools, as well as to learn about new research directions in requirements engineering and service discovery.

Sections 2 and 3 describe SeCSE's service-centric requirements process. Section 4 reports the results from 2 evaluations of SeCSE's requirements specification and

service discovery tools. The paper ends with a discussion of future work in light of the results from these 2 evaluations.

2. Discovering Services in SeCSE

In previous SeCSE work we report an iterative and incremental requirements process for service-centric systems [4]. Requirements analysts form queries from a requirements specification to discover services that are related to the requirements in some form. Descriptions of these discovered services are retrieved and explained to stakeholders, then used to refine and complete the requirements specification to enable more accurate service discovery, and so on. Our approach builds on Fischer et al.'s [2] observations about how design queries are incrementally improved by critiquing results from previous queries. Relevance feedback, as this is known, has important advantages for the requirements process. Stakeholders such as service consumers will rarely express complete requirements or requirements at the correct levels of abstraction and granularity to match to the descriptions of available services. Relevance feedback enables service consumers and analysts to specify new requirements and re-express current ones to increase the likelihood of discovering compliant services. Furthermore, accurate relevance feedback 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. In this sense SeCSE's requirements process is similar to commercial off-the-shelf (COTS)-based software development processes such as COMPOSE [5] and SCARLET [6] that use relevance feedback from selected packages to improve the requirements specification. However, SeCSE goes further by seeking to exploit the large numbers of candidate services that will be available in service registries, in contrast to the COTS focus on one or two software packages at a time.

SeCSE's requirements process is depicted in Figure 1. Service queries are extracted from a requirements specification, then fired at service registries to retrieve service descriptions that are explained to analysts and service consumers to enable them to select the most appropriate

service(s). These services are then used to change the requirements.

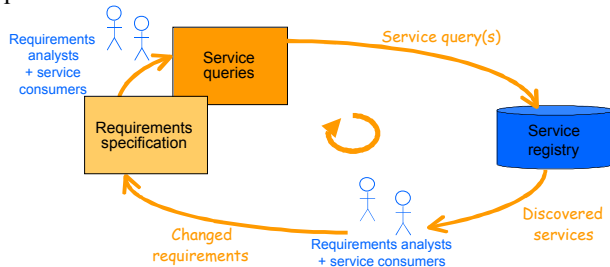


Figure 1. SeCSE's requirements process

The process has 2 important features. Firstly, to ensure its widespread industrial uptake, the process uses established specification techniques based on structured natural language. For example, to specify system behaviour the process uses UML use case specifications [3]. To specify the required properties in a testable form for generating service monitoring policies it uses the VOLERE requirements shell [8]. As such the process extends the Rational Unified Process (RUP) without enforcing its use or mandating unnecessary activities. It treats both the use case specification and requirements statements as the specification.

Secondly, the process uses services that are discovered from service registries to support requirements processes in different ways [4]. During early requirements processes it uses services to challenge system boundaries and discover new requirements. For example, if no services are found with an initial query, the tools provide advice on how to broaden the query to find services that, though not exactly matching the needs of the future system, might provide a useful basis for further specification. During late requirements processes it uses services to decompose and refine requirements and restructure them to enable more effective service monitoring. Service descriptions provide the requirements team with important quality-of-service information, for example about likely system performance and reliability, used to specify measurable fit criteria for requirements [8].

In this paper we report two evaluations of how services can inform early requirements engineering processes. In particular we explore the usability of the environment, and if usable, whether analysts can discover new requirements from discovered services, to validate the conjectured SeCSE requirements process. We support SeCSE's requirements process using the service discovery environment reported in the next section.

3. SeCSE's Service Discovery Environment

The architecture of SeCSE's service discovery environment is shown in Figure 2. It has 4 main components: (i) UCARE, which supports web-enabled specification of requirements and use cases, and formulation of service queries from these specifications: (ii) EDDiE, which uses service queries to discover service descriptions; (iii)

SeCSE's service registry – a federated and heterogeneous mechanism for storing service descriptions that is searched by EDDiE; (iv) the Service Explorer that combines the text and graphics descriptions of retrieved services to enable analysts to understand, select between and use these services to discover new requirements. This section outlines UCARE, EDDiE and SeCSE's service registries and Service Explorer components.

3.1 The Service Registries

The environment discovers services from SeCSE service registries that consist of an existing UDDI registry and one or more facets that describe different aspects of a service. UDDI registries are inadequate for discovering services using criteria such as cost, quality of service and exception handling. Therefore SeCSE has defined 6 facets – signature, service description, operational semantics, exception, quality-of-service, cost/commerce, and testing [9] – to describe features that are important when discovering services. Each facet is described using an XML data structure, linked to the UDDI service entry, and accessed using Java APIs. A set of SeCSE APIs with WSDL interfaces allow access to the UDDI registries and service facets, hiding the internal query mechanisms. Requirements-based service discovery currently uses the service description and quality-of-service facets. In this paper we describe discovery using the service description facet. The quality-of-service facet is used to refine selection once services are discovered.

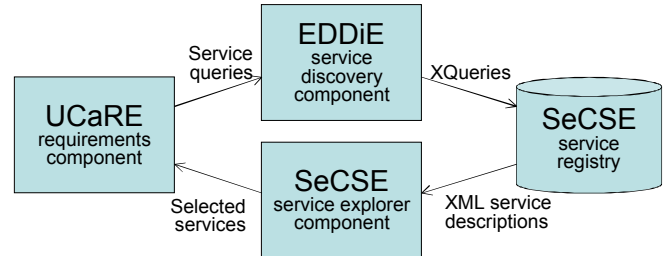


Figure 2. SeCSE's service discovery environment

Part of the XML structure of the service description facet is described in Figure 3. Again, to facilitate industrial take-up, we assume that service providers describe each facet of a service using structured natural language, due to the excessive effort needed to document services more formally. For example the *<ServiceGoal>* facet describes the purpose of the service as an end-state expressed in structured natural language. The *<ShortServiceDescription>* facet describes the service's behaviour using a short paragraph similar to a use case précis, whilst the *<LongServiceDescription>* facet describes this behaviour in more detail using structured English similar to a use case normal course [1].

```

<LanguageSpecificSpecification>
  <FacetType> Service Description </FacetType>
  <FacetSpecificationLanguage> English </FacetSpecificationLanguage>
  <FacetSpecificationOwner> City University </FacetSpecificationOwner>
  <FacetSpecificationLastEdited> 25/05/05 </FacetSpecificationLastEdited>
  <FacetSpecificationData>

```

```

<DescriptionSpec>
  <VersionNumber/>
  <LaunchDate/>
  <ServiceGoal/>
  <ServiceConsumers/>
  <ShortServiceDescription/>
  <LongServiceDescription>
    <ServiceOperations>
      <Operation>...</Operation>
    </ServiceOperations>
  </LongServiceDescription>
  <ServiceRationale/> ...
</DescriptionSpec>
</FacetSpecificationData>
</LanguageSpecificSpecification>

```

Figure 3. The XML structure of part of SeCSE's service description facet

SeCSE's service registries are implemented using eXist, an Open Source native XML database featuring indexed XQuery processing, automatic indexing, and tight integration with existing XML development tools. EDDiE queries these registries using XQuery, a query language designed for processing XML data and data whose structure is similar to XML. Generated queries are transformed into XQueries that are fired at the service description facets of services in the SeCSE service registries.

3.2 The UCARE Requirement Component

We implemented SeCSE's requirements component – UCARE – as a web-based .NET application depicted in Figure 4. UCARE is structured using a three-layer model. Its presentation layer defines the graphical user interface. The application layer defines the system logic specified in ASP.NET pages that generate the dynamic GUI and C# components where classes are grouped by application area. The database layer stores all persistent data about use cases and retrieved services.

Figure 4. Example use case and requirement specification specified in UCARE

A requirements analyst manages requirements and use cases through a web client. UCARE supports tight integration of use case and requirements specifications – a requirement expressed using the VOLERE shell can describe a system-wide requirement, a requirement on the behavior specified in one use case, or a requirement on the behavior expressed in one use case action. UCARE

allows analysts to create service queries from use case and requirements specifications and fire these queries to discover services. It also provides word auto-completion and a project glossary that form part of word sense disambiguation procedures.

At the start of the requirements process, analysts work with future service consumers to develop simple use case précis that describe the required behaviour of the service-centric application. Figure 5 shows a typical use case précis in UCARE, defining what a driver might want from an in-car route planner application. Figure 6 expresses some simple requirements that are associated with the précis for that application. The first, a functional requirement, specifies what the service shall do, and the second, the non-functional requirement, specifies desirable qualities of the service.

A driver is driving his car. The driver wants to find out how long it will take him to get from his current location to a specified destination. The driver activates the onboard trip management service. He selects a destination and requests an estimate of the time it will take him to arrive at that location from the current location.

Figure 5. A simple use case précis for an in-car route planner application, which is used to formulate queries with which to discover services

FR1: The trip management service will provide the driver with route information at different levels of detail.

PR1: The trip management service will provide the driver with route information within 10 seconds.

Figure 6. Requirements on the application

Each service query is formed of one or more elements in the requirement specification according to the conceptual structures depicted in Figure 7. A requirements element is a structured natural language string of a pre-defined type such as a VOLERE requirement description or rationale or use case précis, pre-condition or action. The analyst using UCARE imports specification elements directly to construct service queries composed of one or more query elements.

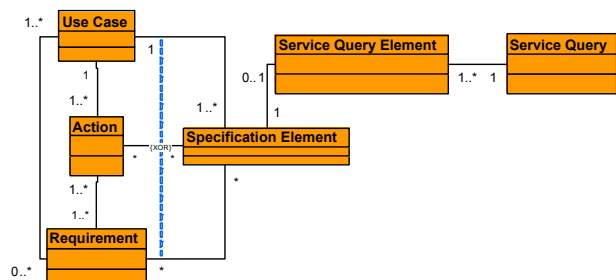


Figure 7. The relationship between a requirements specification and service query

Figure 8 shows the initial XML service query derived from the use case précis and requirements shown in Figures 5 and 6. The query follows an XML schema that defines its structure, content and semantics. Later in the paper we contrast it to the expanded and disambiguated query generated by EDDiE.

```

<Query queryID="53">
  <UseCase useCaseID="2">
    <Precis> A driver is driving his car. The driver wants to find out how
    long it will take him to get from his current location to a specified destination. The driver activates the onboard trip management service. He selects a destination and requests an estimate of the time it will take him to arrive at that location from the current location.
    </Precis>
    <FuncReq FcID="65">
      <FuncDescription>The trip management service will provide the driver with accurate route information at different levels of detail.
      </FuncDescription>
    </FuncReq>
    <NonFuncReq NfID="10">
      <NonFuncDescription> The trip management service will provide the driver with accurate route information within 10 seconds.
      </NonFuncDescription>
    </NonFuncReq>
  </UseCase>
</Query>

```

Figure 8. The XML service query formed of elements in the requirement specification

3.3 EDDiE: The Service Querying Component

The main function of EDDiE's service discovery algorithm is to discover descriptions of candidate services using the service description facet with queries. The algorithm has the 4 key components. In the first the service query is divided into sentences, then tokenized and part-of-speech tagged and modified to include each term's morphological root (e.g. *driving* to *drive*, and *drivers* to *driver*). Secondly, the algorithm applies procedures to disambiguate each term by defining its correct sense and tagging it with that sense (e.g. defining a *driver* to be a *vehicle* rather than a *type of golf club*). Thirdly, the algorithm expands each term with other terms that have similar meaning according to the tagged sense, to increase the likelihood of a match with a service description (e.g. the term *driver* is synonymous with the term *motorist* which is also included in the query). In the fourth component the algorithm matches all expanded and sense-tagged query terms to a similar set of terms that describe each candidate service, expressed using the service description facet, in the SeCSE service registry. Query matching is in 2 steps: (i) XQuery text-searching functions to discover an initial set of services descriptions that satisfy global search constraints; (ii) traditional vector-space model information retrieval, enhanced with WordNet, to further refine and assess the quality of the candidate service set. This two-step approach overcomes XQuery's limited text-based search capabilities. Using this two-step process EDDiE returns an ordered set of services based on the match score with the service query. Unfortunately the full EDDiE algorithm is too complex to describe in full in this paper, but it is described at length in [13].

3.4 The Service Explorer Component

The service discovery environment presents retrieved services to analysts and service consumers. 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. SeCSE also provides a graphical component with which to select between retrieved services using dynamic querying and visualizations.

4. Evaluating UCaRE and EDDiE

We developed SeCSE's service discovery environment as a research prototype in summer 2005, however no industrial evaluation of the environment had taken place. Therefore we undertook 2 evaluations of the environment's utility and usability with our SeCSE industrial partners. The next two sections describe results from these two evaluations.

4.1 Evaluating the Usability of UCaRE's Requirements Component

The purpose of this evaluation was to undertake a formative evaluation of the usability, and to a lesser degree the utility of the UCaRE requirements component. Effective requirements specification is essential if requests and queries for early service discovery are to be formed. Hence a formative evaluation of UCaRE was undertaken early in the project by one of SeCSE's industrial partners.

Computer Associates (CA) undertook the evaluation. CA has extensive requirements engineering experience and expertise. It has assisted many of its customers to develop business models and undertake requirements analysis. Some consultancy assignments result in specifications of a sales/marketing/production area. Deliverables from such consultancy processes were lists of requirements resulting from requirement capture workshops, business workflows & definitions of processes, business concept model, use cases. These deliverables from previous requirements processes in CA provided the baseline of material with which to evaluate UCaRE.

The first step was to develop evaluation and test criteria for UCaRE using the above-mentioned existing specifications and analyst experience in doing "business modeling projects" with large project groups (a mixture IT practitioners and end-users, 10 to 20 people). The evaluation criteria covered all aspects of requirements and use case specification as well as general UCaRE features. Full criteria are described in [6]. During the second step, UCARE was evaluated from the viewpoint of an analyst in a requirements project. The evaluation checked UCaRE for its completeness, user friendliness, fitness for use, maintainability and flexibility. A highly experienced CA consultant and business analyst undertook the evaluation. He spent most of August 2005 undertaking evaluation tasks, rating UCaRE against the evaluation criteria, and documenting findings. The evaluation took place at CA offices in Brussels. The consultant accessed UCaRE through a thin client web browser to the UCaRE application server running in London. As such the technical set up for the evaluation reflected a realistic usage environ-

ment for SeCSE.

Overall the evaluation was a success, in that the CA analyst was able to work with UCaRE throughout the month to complete the series of tasks identified above. That said, the evaluation gave rise to numerous issues to address in the further design and development of UCaRE. Results described both direct responses to the evaluation tasks and new requirements on future versions of UCaRE that are needed to meet CA's needs on it as a requirements management tool in service-centric development environments. The full data set leading to these results is reported in [6]. To summarize these findings the evaluation identified important new requirements on UCaRE to deliver:

- More flexible switching between domains and systems, to enable more effective browsing and reuse of requirement and use case specifications;
- More flexible management of requirements, including visualizing requirements and their structures, detecting conflicts and dependencies between requirements, grouping requirements according to different criteria, and reusing requirements within use cases;
- Life-cycle support for requirements, including versioning of requirements, and capabilities to 'block' requirements as satisfied by services and not included in service queries;
- More support for business concept modeling, including their use to define project glossaries and lexicons;
- Maintaining requirements and use cases using role-based controlled feedback loops from service discovery;
- Extending service queries from requirements with new features that include: (i) prioritization of use of lexicon terms; (ii) expression and use of business logic and formulae; (iii) using requirements to discover composite use cases;
- Logical interfaces to architecture-based service discovery, through the use of packages to exchange requirements, architecture and services information between requirements and architecture components in SeCSE.

To conclude, effective industrial use of UCaRE required more features often found in commercial requirements management tools and need to be re-implemented in UCaRE. We have already implemented several of these new features, including interfaces to our REDEPEND *i** modeling tool to handle business concept modeling, and a requirements browser feature that enables an analyst to search for and reuse existing requirements during requirements modeling tasks. These new features are described in more detail at the end of the paper.

Based on the successful usability evaluation of UCaRE, we undertook a second investigation to explore the utility of UCaRE and EDDiE in SeCSE's requirements process shown in Figure 1.

4.2 Evaluating Whether Discovering Services can Support Requirements Specification

A second evaluation investigated whether EDDiE's algorithm can discover services that are useful in a requirements workshop. We defined useful as sufficient to trigger specification of requirements that had not been specified prior to service discovery during one iteration of SeCSE's requirements process. Four experienced practitioners from SeCSE industrial partners – 2 from Fiat, one from DaimlerChrysler and one from Computer Associates, discovered and documented requirements for the in-car route planner application. Three of them had extensive experience with such automotive applications. Two of the authors ran the workshop – one facilitated requirements discovery whilst the scribe operated UCaRE.

The workshop was in 2 stages. In the first the facilitator walked the practitioners through the use case précis then normal course to discover requirements for the route planner application that the scribe documented in UCaRE. This process continued until the practitioners were unable to discover more requirements. The scribe then generated a service request from the disambiguated use case précis and queried a SeCSE-enabled registry containing 112 services for applications that included weather reporting and flight booking as well route planning taken from existing public UDDI registries. Each service description was written by the original service providers and not modified prior to the workshop. The request expanded all term types with possible synonyms, hypernyms and gloss terms.

In the second stage the Service Explorer displayed each discovered service description as shown in Figure 9. The practitioners selected which service descriptions to retain as pertinent to the route planner application. The facilitator then walked the practitioners through each service to discover additional route planner requirements from the retained services that the scribe also documented in UCaRE. The facilitator took care to avoid bias and use the same prompts and guidelines before and after service discovery. After the workshop each practitioner completed a questionnaire that ranked each requirement documented during the workshop for its importance and novelty on a simple scale of 1 to 3.

Query ID: 194 There are 8 services

<< Back to discovered services...

ID	ServiceName	Description	MatchValue		
96	NNavigation	This service helps you to find the nearest location where you could talk with our reseller. Calculates the arrival time on the basis of the current car position, road preferences and Car features.	4.0427	☒	[View]
112	FlightCostCalculator	This service allows you to calculate the total cost of a flight including the land travel at the point of departure and destination. This includes costs of taxis, trains, metro, subway, bus and airport taxes. It allows you to select the quickest or cheapest options for the complete journey.	3.2986	☒	[View]
94	XNavigation	This service helps the driver to find locations or point of interests during a trip. The information are available both in textual and in graphic form depending on the user preference and on the display capabilities.	3.101	☒	[View]
13	Proximity	The Proximity Service allows you to find all points of interest (POI) locations within a specified distance to a specified location (e.g., find all POI's within 5 miles of x,y) or to determine the nearest specified number of POI locations to a specified location (e.g., find nearest 3 POI's to x,y).	2.5132	☒	[View]
110	FlightTime	The FlightTime service allows you to calculate the total time taken for multiple flights from a given departure point to a specified arrival point, as well as the cheapest route and the route with the least stop overs.	2.2862	☒	[View]

Figure 9. Discovered service descriptions shown in the Service Explorer component

The workshop took place as planned. The first stage lasted 60 minutes, during which the practitioners discovered 27 requirements that were documented in UCARE. The service query then retrieved 11 services, 8 of which the practitioners retained as relevant. The second stage lasted 50 minutes and led to a further 20 requirements.

At the end of the workshop each practitioner completed the questionnaire (the ordering of the requirements was pseudo-randomized), and their average ratings of the importance and novelty of the 47 requirements are shown in Figure 10. Results revealed a clear pattern – each practitioner ranked requirements specified prior to service discovery as more important than requirements documented after service discovery, but requirements specified from discovered services were more novel, suggesting that the retrieved services fulfilled an important role during the requirements process.

	Practitioner	R	M	S	P
Importance rating	Stage1	2.7	2.5	2.0	2.6
	Stage2	2.3	2.1	1.9	2.2
Novelty rating	Stage1	1.3	2.0	1.6	2.4
	Stage2	1.7	2.2	2.3	2.9

Figure 10. Average importance and novelty ratings for requirements discovered before and after discovery

A post-workshop analysis revealed several recurring patterns of relationships between the service description and the resulting requirement description:

- Some requirements expressed new system features that were a consequence of an application that implemented the retrieved service. For example the requirement *the system shall be able to send a message to the driver when the vehicle is a predefined distance of a point of interest* arises only because a service exists to provide the information contained in the message to be sent. It is interesting to note that the practitioners clearly assumed that the final in-vehicle application would implement such services, even though no such agreement had been established during the requirements workshop. Requirements that were a consequence of implementing a service in the application could account for 8 of the 20 requirements generated during the second half of the workshop;
- Some requirements were expression of a refinement of the features of the discovered service applied to the system under analysis. For example the requirement *the system shall be able to send the estimated arrival time to a specified agenda item* expressed a refinement of service operations for reading and writing to the user's agenda. Requirements that were refinements of the features of the discovered service could account for 4 of the 20 requirements;
- Other requirements expressed required inputs to an application that implemented the discovered service, i.e. the requirements enabled the successful invocation of the service. For example the requirement *the system*

shall be able to receive the arrival time from specified agenda a source was needed to provide the input data needed by the retrieved service;

- One requirement expressed a function that had the potential to satisfy service qualities described in the service description. The requirement *the system shall be able to check outgoing and ignoring incoming messages for viruses* expressed how to expose securely the company agenda;
- Two requirements could be linked to the service descriptions for which the requirements were specified, but through some shared deeper concepts rather than through input, output and consequence relations. The functional requirement *the system shall be able to calculate the estimated journey time based on the driver's capabilities and patterns* was generated from the FlightTime service that calculates the cheapest overall route via multiple stopovers. On the surface no immediate similarities emerge. However, one closer analysis, both the service and requirement encapsulate the idea of optimised plan recognition. Similarly the functional requirement *the system shall be able to select the mode of communication based on current driving conditions* has no surface similarities with the FlightCostCalculator service from which it was specified. Yet both share the deeper concept of different modes – modes of communication and modes of transport – that might explain the requirement. Although we lack corroborating evidence for these claims, these two requirements indicate more complex and richer forms of requirements triggers;
- Finally, in a similar vein, two requirements had no discernible similarities with the triggering service descriptions. The absence of such similarities might indicate different creative reasoning strategies, which suggests that the practitioners sometimes engaged in creative reasoning about new requirements even without explicit triggers to encourage them to do so.

To conclude this workshop, although limited by the number and coverage of services in the available registry, revealed that EDDiE is capable of discovering services deemed relevant from first-cut requirements and a use case specification specified in a one-hour workshop. Furthermore 8 of the discovered services enabled the experienced practitioners to specify new requirements that they deemed more novel than the earlier requirements, thus providing evidence that EDDiE and UCARE can deliver SeCSE's iterative and incremental requirements process. Alas, time constraints did not allow us to reformulate a new service request from new requirements that, in turn, might have discovered further new services.

5. Discussion and Future Work

This paper reports SeCSE's research-based software environment for constructing service queries from natural language requirements specifications, and two formative

evaluations of the usability and the utility of the environment with industrial partners. In the first evaluation use of the environment over a one-month period by an experienced analyst revealed that UCARE is potentially usable, but lacking in important features that we are now adding to make it a complete web-enabled requirements component. In the second evaluation use of the environment in a workshop revealed that it discovered services that experienced practitioners used to discover new requirements for an automotive application.

As reported we are now extending UCARE with new features identified in the first evaluation. One such feature is the need for business concept modeling. We have adopted the *i** approach [12] for business modeling. *i** is an approach originally developed to model information systems composed of heterogeneous actors with different, often-competing goals that depend on each other to undertake their tasks and achieve these goals – like the socio-technical systems found in the automotive and telecommunications domains. We support *i** modeling of business concepts with the REDEPEND graphical modeling and analysis tool plug-in to Visio. REDEPEND gives analysts with *i** modeling and analysis functions, coupled with additional functionality and the reliability of Microsoft's Visio product. It provides a graphical palette from which systems engineers can drag-and-drop *i** concepts in order to develop business models. An analyst can now import these business models into UCARE directly to generate queries for discovering services from modeled concepts and descriptions. We intend to report further evaluations of this new feature in the future.

Similarly we have implemented a new feature to allow an analyst to explore already-specified requirements for different domains and systems. The analyst can select to browse and search for all requirements specified for an entire system or one use case specified for that system, by requirement type. If an existing requirement is found, the analyst can either associate it with a different system or use case, or copy it and edit to specify a similar requirement for the new system being specified. Again, we hope to return to CA to evaluate these new features of UCARE in SeCSE's second evaluation phase.

The second evaluation demonstrated the potential effectiveness of an iterative requirements process. Indeed, the success of the exercise revealed the potential benefits of a more flexible, search-and-specify process. As such we are currently designing new features for UCARE and EDDiE to enable more exploratory specification of requirements based on rapid service discovery. In particular we have used results to define guidelines that encourage stakeholders to:

- Envision the consequence of a service being implemented in application, to discover requirements that become possible only if the service is invoked;
- Discover requirements that refine service operations that, due to the more abstract nature of software ser-

vices, are not specified in the service description. In particular, examine the types of information that is processed by the service and the types of devices through which the service might be invoked;

- Envision what the invoked service will need from the future application, then express requirements that, if satisfied, will bring about the pre-conditions for the service to be invoked successfully.

Elsewhere we are currently extending the environment in several other directions. These include enhancing the algorithm to filter and rank discovered services using quality-of-service requirements, as well as extending the user-defined XQuery functions to compute the semantic similarity measurement during query matching. We also plan to provide new tools that present services in terms automatically generated requirements statements that analysts can accept or reject as relevant.

6. Acknowledgements

SeCSE is funded by the EU 511680 Integrated Project.

7. References

- [1] Arlow J., Neustadt I., 2005, 'UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design', Addison-Wesley Obj. Techn. Series.
- [2] Fischer G., Henninger S. & Redmiles D., 1991, 'Intertwining Query Construction and Relevance Evaluation', Proceedings CHI'91, eds S.P. Robertson, G.M. Olson & J.S. Olson, ACM Press, 55-62.
- [3] Jacobson I., Booch G. & Rumbaugh J., 2000, The Unified Software Development Process, Addison-Wesley-Longman.
- [4] Jones S.V., Maiden N.A.M., Zachos K. & Zhu X., 2005, 'How Service-Centric Systems Change the Requirements Process', Proceedings REFSQ'2005 Workshop, CaiSE'2005, 13-14 2005, Porto, Portugal.
- [5] Kotonya, G., Sommerville, I., and Hall, S., 2003, 'Towards A Classification Model For CBSE Research. Proceedings of the 29th Euromicro Conference, Antalya, Turkey.
- [6] Maiden N.A.M., Spanoudakis G., Zisman A., Zachos K., Zhu X., Jones S.V., Cuypers H., Matthews P., 2006, 'A2.D6: Evaluation of Service Discovery Tools', Technical Report SeCSE project, Centre for HCI Design, City University, London, June 2006.
- [7] Maiden N.A.M., Kim H. & Ncube C., 2002, 'Rethinking Process Guidance for Software Component Selection', Proceedings 1st International Conference on COTS-Based Software Systems, LNCS 2255, Springer-Verlag, 151-164.
- [8] Robertson S. & Robertson J., 1999, Mastering the Requirements Process, Addison-Wesley-Longman.
- [9] Sawyer P., Hutchinson J., Walkerdine J. & Sommerville I., 2005, 'Faceted Service Specification', Proceedings SOCCER (Service-Oriented Computing: Consequences for Engineering Requirements) Workshop, at RE'05 Conference, Paris, August 2005.
- [10] Singhal A., Pereira F., 1999, 'Document expansion for speech retrieval'. In Proceedings of ACM SIGIR, pages 34-41, Berkeley, CA, USA.
- [11] SeCSE 2005, secse.eng.it.
- [12] Yu E. & Mylopoulos J.M., 1994, 'Understanding "Why" in Software Process Modelling, Analysis and Design', Proceedings, 16th International Conference on Software Engineering, IEEE Computer Society Press, 159-168.
- [13] Zachos K., Maiden N.A.M., Zhu X & Jones S., 2006, 'Discovering Web Services To Specify More Complete System Requirements', Technical Report, Centre for HCI Design, City University, London.