How Service-Centric Systems Change the Requirements Process

S.V. Jones, N.A.M. Maiden, K. Zachos & X. Zhu

Centre for HCI Design, City University
Northampton Square, London EC1V OHB, UK

Abstract. Service-centric systems engineering, and in particular systems development with web services, presents new challenges for requirements processes, techniques and tools. This paper reports a new requirements process that has been developed within the SeCSE consortium to address these challenges. It describes the distinguishing characteristics of the process, and demonstrates them with an example from web services development in the automotive domain, The paper ends with a review of related requirements work and describes future work in SeCSE.

1. Service-Centric Systems and Requirements

Service-centric systems are an important and emerging paradigm in computing. Service-centric systems are systems that integrate services from different providers regardless of the underlying operating systems or programming languages of those applications. A software service is a set of operations offered by some software application. Users can access such services through a well-defined interface independently of where the service is actually being executed. Most research into service-centric systems is on the integration of web services that are accessible to developers and users through the Internet, although software services can also be provided through reusable software components or shrink-wrapped off-the-shelf packages.

Although not immediately obvious, the development of service-centric systems has several important consequences for determining the requirements of these systems. Firstly, developers will want to discover candidate software services early in the development process to explore what capabilities, functions and features of a new service-centric application are possible. Therefore, requirements will form elements of queries with which to discover candidate services for an application. Secondly, and perhaps more importantly, registries of service descriptions and executable services available over the Internet will, for the first time, provide end-users with direct access to elements the solution space without the need to go through software developers. This, we conjecture, will change the nature of requirements processes and service-centric applications, and introduce relevance feedback – query reformulation using

the results of queries – to change requirements in light of available services and their descriptions.

In this paper we exploit how service-centric systems engineering offers new opportunities for improving requirements specifications. One opportunity is that service descriptions that are retrieved in response to service queries can then refine and decompose the original requirements that gave rise to the query. For example, imprecise and incomplete requirements for a restaurant location application can be refined quickly using detailed descriptions of the behaviour and non-functional qualities of restaurant location services discovered in service registries. The more precise and complete requirements can then form more precise queries that discover more compliant services, and so on. However, to make the most of the new opportunities we need new requirements processes, techniques and tools for expressing requirements to enable query formulation and exploiting retrieved service specifications in different requirements tasks such as requirements exploration, decomposition and refinement.

We are exploring these opportunities and challenges in the EU-funded FP6 SeCSE (Service-Centric Systems Engineering) integrated project. This paper reports a new requirements process that uses descriptions of services, made available by service providers and discovered by systems integrators to improve the specification of requirements of service consumers. It is in 5 sections. Section 2 describes research drivers in service-centric computing. Section 3 outlines how services are discovered for different purposes in the requirements process. Sections 4 describes the SeCSE requirements process and software tool support, and demonstrates it using a simple example of requirements and services taken from one of SeCSE's application domains – automotive engineering. The paper ends with initial feedback on the process from SeCSE industrial partners and plans for future development and evaluation of the process.

2. It's a Service-Centric World

Recent developments in web services and standards have been rapid. Standards such as SOAP and WSDL are now well established. Major vendors such as IBM, Microsoft, Sun and HP provide support for services in their development platforms, and many companies are offering web service interfaces to their systems. Web service discovery standards have been established and organisations such as UDDI have established directories of web service providers. Leavitt (2004) reports that worldwide spending on web services-based software projects will reach $11 billion by 2008, compared to $1.1 billion in 2003. He also reports a Gartner survey of 110 companies that reports that 54% are already working on web service projects or plan to start soon. Given these trends, development of service-centric systems with web services is a new research challenge for software engineering.

Another important trend is the growth in end-user computing. Sutcliffe & Mehanijev (2004) report that, by 2005 in the US alone, there will be 55 million end-user developers compared with 2.75 million professional software developers. As software becomes more ubiquitous, end-users become more knowledgeable about

what software can offer them, and more articulate about requirements and features for new applications. This empowerment of end users has important implications for the development of service-centric systems. We anticipate that, as service registries grow and access to them becomes easier, end user service consumers as well as systems developers will access them to explore what software can do for them prior to expressing their requirements for a new system. This also has important implications for requirements processes and how requirements will be expressed.

Existing requirements processes are not well suited to service-centric systems engineering. Firstly, service-centric systems blur old-fashioned distinctions between development, deployment and implementation. The availability of executable services means that applications can be deployed quickly, and requirements will be referenced during implementation when services are monitored for requirements compliance. Secondly, using discovered service descriptions to improve specifications means that requirements process will be both highly iterative and incremental, leading to shorter development phases. Thirdly, end-user browsing of service registries is likely to change the nature of requirements expression from statements of abstract properties that a future system shall exhibit to more comparative statements equivalent to "*I want one like that*".

In contrast, commercial requirements methods such as RUP (Jacobson et al. 2000) and research-based processes such as *i** (Yu & Mylopoulos 1994) and KAOS (van Lamsweerde 2004) assume a more top-down approach to requirements specification that do not facilitate the frequent exploration of solution spaces. Similarly, requirements management tools provide little support for evolving service-centric architectures alongside requirements specifications, and offer none of the functionality needed to discover, explain and select between services, then revise requirements in terms of the selected services.

Current developments in service discovery in service-centric computing do not consider requirements and requirements processes explicitly. Systems developers locate new services by browsing existing UDDI registries using the UDDI Inquiry Application Programming Interface (API). This API supports 3 different types of inquiry: browsing, drilling-down and invocation (Alonso et al. 2004, Guruge 2004). When browsing, a developer searches using a broad categorization of interest, and then refines the inquiry using more specific criteria as each set of results is displayed. Categorizations would typically relate to the UDDI data structures of *businessEntity* (containing business information about service providers), *businessService* (containing descriptive information, meaningful to human readers, about sets of services being offered by previously identified businessEntities), *bindingTemplate* (containing binding information necessary to invoke and use a particular, previously described, service) and *tMode*l or 'technical model' (containing more technical descriptions about, and pointers to technical specifications of, a particular service). Drilling-down involves accessing a known, specific service using a UDDI key, which may have been obtained from a previous session of browsing. Finally, invocation involves locating and then running a particular service.

The UDDI data structure most relevant during service discovery is the *businessService* entity. Ideally, the developer queries each service's behaviour and capabilities using requirements that express the desired capabilities and behaviour. However UDDI data structures provide no guidelines for formulating these queries,

for defining their level of abstraction, or for using structures that improve the probabilities of successful discovery.

Web service brokers offer some refinements on these interfaces, for example searching across multiple registries, and displaying only web services that are working. SalCentral (http://www.salcentral.com/Search.aspx) allows users to search on elements such as country, data type, description, input parameters and methods (as well as 16 others). WSIndex (http://www.wsindex.org/) and WebserviceX.NET (http://www.webservicex.net/WS/default.aspx) provide basic categorizations such as *companies*, *communications*, *graphics and multimedia* and *governance*, and WSIndex also allows users to enter free text search strings for which the search engine will look for exact or similar matches. However, again, there is no explicit interface to requirements processes, structures and techniques during these brokering processes.

3. Requirements-Based Service Discovery in SeCSE

The mission statement of SeCSE is to create new methods, tools and techniques for system integrators and service providers that support the cost-effective development and use of dependable services and service-centric applications (SeCSE 2005). The four-year research program is in 4 main activity areas:

1. *Service engineering*, which addresses the needs of service developers who require methods to define and develop dependable services. The focus is on service specification of functional and non-functional aspects of the service for service consumers and integrators, and service validation when the providers do not know who your future users will be;
2. *Service discovery*, which addresses the discovery of the specified services by service integrators from registries using the needs of service consumers. Services are discovered both before deployment to improve requirements specification and architecture modeling, and after deployment to replace services that do not comply with these requirements based on monitoring data;
3. *Service-centric systems engineering*, which supports systems integrators to compose services to obtain behaviours and results not available from a service in isolation. Services are composed through service-oriented architectures using architecture styles for such systems;
4. *Service delivery*, which offers support to manage deployed services by monitoring these services for compliance with agreed service contracts, based on the specified requirements, and dynamically switching services during execution if one fails.

Research results will be applied in the European automotive sector with Fiat and Daimler-Chrysler, the telecommunication sector with Telecom Italia and Telefonica, and software development sector with ATOS and Computer Associates.

This paper's authors are leading the service discovery research that will deliver the innovative processes and software tools to discover and express requirements that can be used to discover relevant service specifications, and to use these specifications to

inform better discovery and expression of requirements. That is what we describe in the rest of this paper.

In SeCSE we are currently developing service discovery tools that will support more sophisticated types of query. Systems integrators will be able to search external registries directly as above, but will also be able to use locally generated service classifications and patterns to help locate relevant services. Different types of queries, such as analogical matching and constraint removal, will enable different search strategies during requirements processes. SeCSE is extending service specifications beyond the UDDI registries to include semantic information about a service's capabilities, qualities such as performance, reliability and usability, and behaviour to handle particular classes of abnormal external event. This will enable SeCSE tools to retrieve descriptions of services compliant with non-functional requirements and fine-grain behaviours that handle unexpected inputs. SeCSE partners such as Microsoft and Computer Associates are working to establish these semantic service registries as standard. All of this is underpinned by one important assumption – that providers will describe services using informal or semi-formal because most will not be able to deliver more formal documentation for a service.

In the next section we describe SeCSE's requirements process, developed in collaboration with our industrial partners, to discover and use services with these tools. The process assumes the existence of a requirements-based service discovery engine that we outline in this paper, but is described in detail elsewhere.

4. The SeCSE Requirements Process

SeCSE's requirements process has been designed to encourage 4 characteristics of a service-centric requirements process: (i) an incremental and iterative process; (ii) the divergence from and convergence to established requirements; (iii) a requirements process that can be tailored to local contexts, and (iv); integration with established requirements processes, techniques and representations. Each is described in turn.

4.1. An Incremental and Iterative Process

The SeCSE requirements process is iterative and incremental. Systems integrators 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 the stakeholders, then used to revise and refine the requirements specification to enable more accurate service discovery, and so on. In this aspect of the process we build on Fischer et al.'s (1991) observations about how queries are incrementally improved by critiquing results from the previous query. Relevance feedback, as this is known, has important advantages in the requirements process. Stakeholders such as service consumers will rarely express requirements at the correct levels of abstraction and granularity to match to the descriptions of available services. Relevance feedback enables service consumers and integrators to re-express their requirements to increase the likelihood of discovering services that are compliant with their requirements. Furthermore, accurate relevance feedback provides information about whether requirements can be satisfied by available services, to guide the

integrators 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 (Kotonya et al. 2003) and SCARLET (Maiden et al. 2002) that use relevance feedback from selected packages to improve the requirements specification.

SeCSE's iterative and incremental process is depicted in Figure 1. Service queries are extracted from the requirements specification. In most socio-technical systems not all of the required system behaviour will be implemented using a service – human actors and other software solutions such as legacy, bespoke, component and COTS solutions will also be selected during the work allocation task to implement behaviour. Queries are fired at service registries to retrieve service descriptions that are explained to service integrators and consumers to enable them to select the most appropriate service(s). These services are then used to change the requirements using different strategies. For example, if no services are found in an initial query, the process and 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.

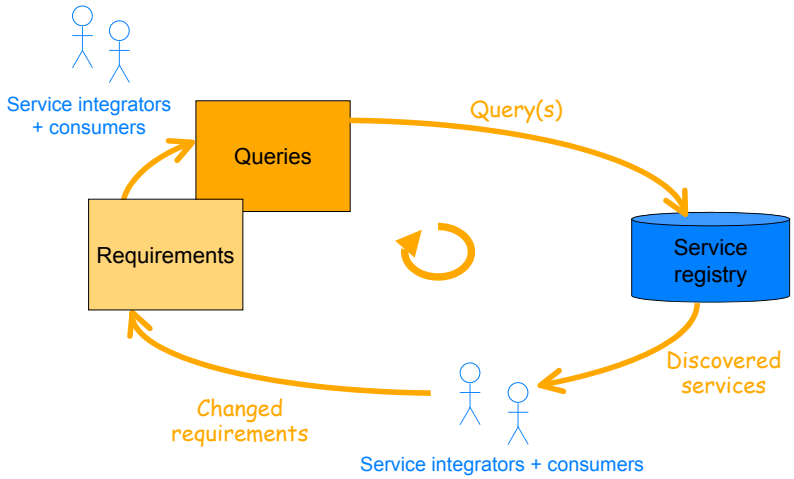

Figure 1. SeCSE's iterative and incremental requirements process.

4.2. Requirements Divergence and Convergence

Requirements engineering is often divided into early and late requirements processes. During early requirements processes a requirements team seeks to establish system boundaries, acquire and discover requirements, and explore dependencies both with adjacent systems and between actors in the system being specified. Its purpose is to surface all of the requirements to be specified, and to understand how the future system will interact with its environment. In contrast, during late requirements processes, the team specifies these requirements, models the system, and analyses

these models for important system properties such as robustness and completeness of the specified functionality.

In SeCSE we use services that are discovered using service query tools to support early and late requirements processes in different ways. During early requirements, we use services to encourage divergence activities to challenge system boundaries and assumptions, and discover new requirements. Until recently requirements engineering has not been recognized as a creative process (e.g. Nuseibeh & Easterbrook 2000). However, the emergence of new systems means that stakeholders increasingly create and invent ideas that they express as requirements. SeCSE uses service descriptions to support tasks that encourage different forms of creative thinking:

- *Analogical matching*: discovering services that are analogical to services that will be implemented in the deployed system, to encourage exploratory creative thinking about requirements;
- *Pattern matching*: using problem-solution patterns to discover services that can perform a function specified in the requirements according to quality of service requirements;
- *Random retrieval*: discovering random services to provoke creative thinking and new ideas about the design of a service-oriented system;
- *Constraint removal*: relaxing constraints on, for example, domain or quality of service characteristics in order to retrieve a broader range of services that can be used to creative and invent new requirements, often by challenging assumptions and changing system boundaries.

All these types of processes, and the queries needed to deliver them, are supported within the SeCSE environment.

During late requirements processes we use services to support convergent activities such as decomposing and refining specifications of requirements, and restructuring the requirements 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 (Robertson & Robertson 1999) for a requirement.

4.3. Flexible Requirements Processes

Requirements processes are rarely the same, and service-centric requirements processes are no exception. In SeCSE we have worked closely with our application partners from the outset to develop requirements processes and tools that can be tailored to different requirements situations. SeCSE offers 5 different configurations of processes, techniques and tools to support:

- Creating simple, single function systems using services;
- Creating systems with several different functions;
- Creating more complex systems;
- Developing new and innovative systems;
- Upgrading existing systems using services.

To specify simple function systems, SeCSE uses established requirements acquisition and business goal or workflow modeling techniques to create simple queries with which to retrieve candidate services that are then prototyped, selected and used to refine the requirements. For systems with different major functions,

SeCSE also uses use case specifications and scenario walkthroughs to structure requirements and elaborate queries for discovering a set of related services. For more complex socio-technical systems, there is greater use of requirements acquisition and business modeling techniques to associate the service-centric system to its environment, and the use of impact analyses to explore the consequences of the service-centric system on its business and domain environment. For more innovative service-centric projects that develop new systems and products, we also run creativity workshops (Maiden et al. 2004) that use service descriptions to provoke creative thinking about the requirements and concepts for the service-centric system. Finally, for service-driven system upgrades, current system modeling and impact analyses drive requirements specification and service discovery activities.

Figure 2 shows the allocation of different SeCSE requirements sub-processes for these 5 different contexts. It provides a guide for service integrators to use the best-fit processes, techniques and tools during a service-centric development project.

	Create single function system	Create multi-function system	Create a more complex system	Develop innovative systems	Upgrade existing systems
Modeling the current system					√
Business goal modeling	√		√	√	
Requirements acquisition	√		√	√	
Create business workflow or use case diagram	√	√	√	√	
Develop use case precis		√	√	√	
Run a creativity workshop				√	
Develop use case specifications		√	√	√	
Walk through scenarios		√	√	√	
Prototype and test services	√	√	√	√	
Assess system impact					√
Select best-fit services	√	√	√	√	√
Review effect of service update	√	√	√	√	√

Figure 2. Different possible configurations of SeCSE requirements processes to implement in different project contexts.

4.4. Using Existing Requirements Artifacts

To ensure the industrial uptake of SeCSE solutions, the process uses established requirements specification techniques as a basis for developing service discovery queries. For example, to specify the required system behaviour the process uses UML-compliant use case specifications and diagrams (Jacobson et al. 2000). To specify the required properties in a testable form suitable for generating service monitoring policies, it uses a modified version of the VOLERE requirements shell (Robertson & Robertson 1999). And to acquire and discover requirements from stakeholders, it combines well-established techniques such as interviews and card sorting (Maiden & Rugg 1996) with scenario walkthroughs (Maiden 2004). As such, the SeCSE process is designed to extend the Rational Unified Process without enforcing its use and mandating unnecessary processes that are unsuitable in different project contexts.

The next section demonstrates key elements of the SeCSE requirements process using a real-world case study from the project. The case study, taken from FIAT, one of our application partners, explores how services for use in automotive systems can inform the requirements and specification of these service-centric systems.

5. An Example of SeCSE Requirements and Services

FIAT, the automotive manufacturer, is looking to use web services in the automotive domain for different purposes. One is to improve customer satisfaction by providing the car with an advanced and customizable telematics device capable of providing the driver with high numbers of services. The second is to improve the customer relationship management process, and consequently customer satisfaction, by activating some services like the remote diagnosis and repair. We are developing the SeCSE requirements process to support the specification of such systems. Service-centric applications include repair diagnostics, navigation and localization, communication and information, and spare parts store management. These services are provided to the driver via a haptic device in the car that the driver can use without decreasing safety whilst driving.

In the example we will demonstrate two simple iterations of the SeCSE process depicted in Figure 1. Early on in the process we use a simple use case précis to discover one candidate service that is compliant with these requirements. Later on we use the discovered service description to revise the précis to produce a use case specification that in turn enables more precise service discovery.

At the start of the requirements process, service integrators work with future service consumers to develop simple use case précis that describe the required behaviour of the service-centric application. Figure 3 describes a typical use case précis, defining what a stakeholder service consumer – the driver – might want from an on-board car service. Figure 4 defines some simple stakeholder requirements for that application that are associated with the précis. The first, a functional requirement, specifies what the service shall do, and the second, the non-functional requirement, specifies desirable qualities of the service. In SeCSE we use a modified version of the VOLERE shell (Robertson and Robertson 1999).

A driver is driving his car. The car's on-board diagnostic system detects an engine problem. The engine is misfiring. The driver activates FIAT's remote-maintenance service. The service provides the location of the nearest garage to repair the car. The driver follows directions to the garage.

Figure 3. A simple use case précis for an onboard remote maintenance application, which is used to formulate queries with which to discover services.

FR1:	The remote-maintenance service will provide the driver with directions to the nearest garage.
RR1:	The remote-maintenance service will provide the driver with reliable directions to the nearest garage.

Figure 4. Requirements on the on-board remote maintenance application.

Figure 5 shows the use case form and VOLERE shell for requirement FR1 implemented in the web-enabled SeCSE environment. Not all of the shell fields need to be completed when a requirement is first identified. The *description* field may then be used to generate a query to help identify candidate services. The results returned from such a query may lead to the requirement being modified and more fields in the requirement shell be completed. The web-enabled tool depicted in Figure 5 shows how specifications (and queries) are formed from structured text, thus allowing service consumer representatives and other end-user developers (e.g. automotive engineers) to use the SeCSE platform.

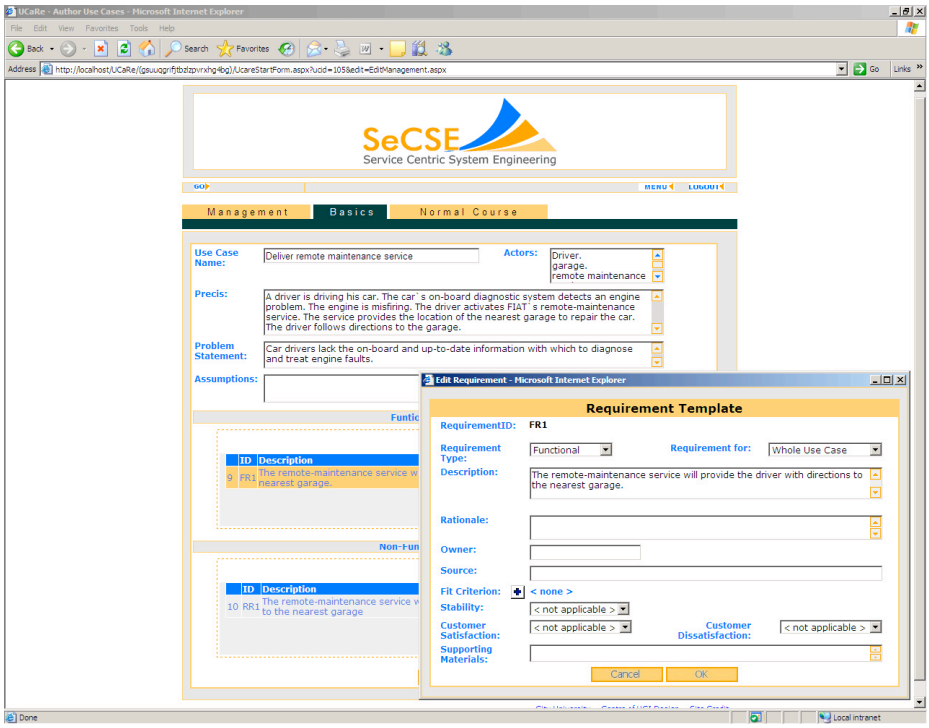

Figure 5. The use case précis form and a requirement partially specified in the SeCSE environment.

Service integrators use the use case précis and functional requirements to generate service queries using the précis text and requirement descriptions fields. If needed, the integrators can also construct more complex queries by including shell attribute values from the non-functional requirements, such as measures of the required quality of service and terms that describe required qualities of the service. In the SeCSE environment, queries are formed using simple functions that export selected elements of the specification to a pre-formed query using a small number of mouse clicks.

SeCSE's requirements-based service discovery algorithm has 2 basic components. The first uses the WordNet 2.0 on-line lexical reference system (Morato et al. 2004) to expand the service query. Query expansion is a process of adding new terms to a given query in an attempt to provide better contextualization, in order to retrieve

documents that are more useful to the user (Baeza-Yates et al. 1999). It is particularly well suited to requirements-based queries that, by definition, are incomplete and inconsistent. In the second component, SeCSE applies word sense disambiguation to an expanded query. An ambiguous query term is a word with multiple senses, where a sense is a group of similar usages of a word dissimilar from other usages (Schütze et al. 1995). It is often impossible to resolve which sense is intended when words are taken out of their context. However, when the context is taken into account, the sense can be determined by various cues, like nearby words and semantics. Several word sense disambiguation algorithms have been developed. In SeCSE we are currently implementing the structural semantic interconnection algorithm (Navigli et al. 2004).

Furthermore, the algorithm uses conceptual structures in WordNet to implement different types of queries by expanding them in different ways to retrieve different types of service description. The query strategies that we are implementing include:

- *Semantic matching*: using synonyms of actor names and terms taken from the requirements in order to discover a wider range of services that might be compliant with the requirements;
- *Analogical matching*: searching for analogous services in different domains, as a means of exploring the requirements space;
- *Constraint removal*: relaxing constraints given in non-functional requirements, in order to explore the requirements space, especially where these may have been over-specified;
- *Pattern matching*: matching generalizations of requirements against pre-defined patterns in the ESD repository, as a route to identifying relevant services and thereby refining requirements.

Returning to our automotive case study, SeCSE's service discovery engine uses the semantic matching strategy to discover service descriptions of services that are compliant with the specified requirement. Each service description is structured using extensions to existing UDDI registries such as that shown in Figure 6. Other service discovery strategies, if applied, will discover other service descriptions to support other requirements tasks, for example analogical retrieval might retrieve services that diagnose computer hardware faults in a network that can provoke discovery of new requirements. Similarly, constraint removal might retrieve services that both diagnose and auto-correct faults directly, providing alternative boundaries and requirements for service integrators and consumers to consider.

Attribute	**Description**
Service ID (key)	Uuid:A761A500-674B-FC35-F678-5468987D5364
Service name	Advanced Diagnostic Service
Service provider business ID	D2348745-4dfgr-456d-45df-743735649j87
Service provider business name	X_Service
Service provider business description	X_Service FIAT solution provider
Service provider business contact info	Roberto Palermo
Service provider industry	Automotive servicers
Service provider location	Italy
Service description	This advanced diagnostic service sends automotive fault data to diagnostic services of the part suppliers identified as responsible of the problem.

Detailed service description	This advanced diagnostic service sends automotive fault data to diagnostic services of the part suppliers identified as responsible of the problem. Having received the response from the supplier services, builds a sorted list of causes with the related workarounds and advices to fix the problem.

Figure 6. An example retrieved service description for the automotive domain.

The SeCSE environment presents and explains each discovered service to service integrators and consumers to enable their selection and acceptance or rejection as relevant to the service-centric application. In our example, the service integrator accepts the described service as potentially compliant with the requirements as specified, and uses it to drive further requirements specification using use case specifications and the more complete VOLERE requirements shell. An example of such a specification is shown in Figure 7. Many of the use case attributes are shown. The service integrator writes normal course specification using information from the discovered service presented in Figure 6. Actions specify different diagnoses activities and data exchange between the different diagnostic services, and requirements specified for action number 7 (**bolded** in the use case) refine the earlier specification. As such, the integrator is specifying the requirement for a system that might implement the discovered service and use its features. SeCSE currently does not support inclusion of abstract services in the use case specification, but we plan to extend it to include references to discovered services to simplify and speed up the specification process.

At this stage in the process, the integrator is beginning to associate particular services with the functionality defined in particular use cases. Elements of use case specifications are used to generate queries which will identify sets of services relevant to particular use cases as described below. Again, SeCSE's web-enabled environment enables FIAT's service integrators to specify such as use case, as shown in Figure 8. The *problem statement*, *added value* and *justification* fields may be used to provide additional context for, for example, disambiguation of terms when constructing the query. The definitions of successful and unsuccessful end states may also be used to refine the query.

Attribute	**Description**
Use Case ID	UC1
Use Case Name	Deliver remote maintenance service
Author	Sara Jones
Date	15th March 2005
Source	FIAT stakeholders, Torino
Actors	Driver, on-board diagnosis system, remote maintenance service, garage.
Problem statement	Car drivers lack the on-board and up-to-date information with which to diagnose and treat engine faults.
Precis	A driver is driving his car. The car's on-board diagnostic system detects an engine problem. The engine is misfiring. The driver activates FIAT's remote-maintenance service. The service provides the location of the nearest garage to repair the car. The driver follows directions to the garage.
Functional Requirements	FR1: The remote-maintenance service will provide the driver with directions to the nearest garage.
Non-functional Requirements	RR1: The remote-maintenance service will provide the driver with reliable directions to the nearest garage
Added Value	Improved customer relationship management, leading to increased return business and revenue.
Justification	The availability of maintenance service information from reliable sources.

Triggering event	The car engine misfires.
.....	
Normal Course	1. The on-board diagnosis system detects the engine problem. 2. The on-board diagnosis system diagnoses the category of engine problem. 3. The on-board diagnosis system informs the driver of the problem. 4. The driver activates the remote maintenance service. 5. The advanced diagnostic service identifies the relevant parts suppliers who are responsible for the problem. 6. The advanced diagnostic service sends automotive fault data to the diagnostic services of the parts suppliers. **7. Each diagnostic service of a parts supplier provides diagnoses of using the fault data.** **FR2: The diagnostic service shall rate each diagnosis with a likelihood score.** **FR3: The diagnostic service shall rate each diagnosis with a severity score.** **PR1. The diagnostic service shall provide the diagnosis within 1 minute of the request being made.** 8. .....
.....	

Figure 8. Part of the use case specification for remote maintenance.

Figure 9. The use case specification modeled in the SeCSE environment.

As in the previous iteration, the service integrator can create a query to discover candidate services for one function of the system using the action description and the requirements associated with that action – that is the bolded text in Figure 8. Retrieved service descriptions can again be used to refine that part of the use case specification, and in particular refine and trade-off the satisfaction of non-functional requirements that cannot be complied with fully from the available services.

To conclude this simple example demonstrates the requirements and use case structures that SeCSE's requirements processes uses as part of a wider service-centric systems engineering process, and outlines the software tools that we are developing to support it. It also demonstrates how even simple discovered service descriptions can provide relevance feedback to inform further requirements specification processes, thus driving an iterative and incremental process.

6. Related and Future Work

This paper reports the results of exploratory research to investigate new problems in requirements engineering – processes, techniques and tools to support service-centric systems engineering. It draws on previous research in relevance feedback during query formulation, query expansion and word sense disambiguation in information retrieval, as well as on established requirements engineering techniques. In the first 6 months of SeCSE we have established agreed processes and information structures that provide the context and requirements for requirements-based service discovery tools that are also outlined in the paper.

Although there is increasing research in web services, there is little related requirements research. Robinson (2003) applies the KAOS requirements method to monitor web services for requirements compliance. Monitors are assigned from obstacle analysis, and derived from requirements specifications. Savigni and Tisato (2004) also address the requirements monitoring challenges of deployed environments. In contrast, there is little reported work on tightly-coupled requirements specification and service discovery tools such as that presented.

The next stages in our work are to evaluate the utility and usability of SeCSE's specification environment with application partners, and to complete the implementation of the first version of the service discovery algorithm and engine. Although the SeCSE process has been developed with our industrial partners, we currently lack formal feedback evaluation from systems developers. During formal evaluation we will explore whether end-user developers such as automotive engineers can use SeCSE to discover service descriptions that inform requirements processes.

Acknowledgements

The authors wish to thank all members of the SeCSE project consortium. The work reported in funded by EU Integrated Project contract 511680.

References

Alonso G., Casati F., Kuno H. and Machiraju V., 2004, 'Web Services: Concepts, Architectures and Applications', Springer, 2004.

Baeza-Yates, R. and Ribiero-Neto, B., 1999 "Modern Information Retrieval", Addison-Wesley, 1999.

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.

Guruge A., 2004, 'Web Services: Theory and Practice', Elsevier Digital Press.

Jacobson I., Booch G. & Rumbaugh J., 2000, 'The Unified Software Development Process', Addison-Wesley

Kotonya, G., Sommerville, I., and Hall, S., 2003 . Towards A Classification Model For CBSE Research. Proceedings of the 29th Euromicro Conference, Antalya, Turkey.

Leavitt N., 2004, 'Are Web Services Finally Ready to Deliver?', IEEE Computer, 37(11), 14-18.

Maiden N.A.M., 'Systematic Scenario Walkthroughs with ART-SCENE', in 'Scenarios, Stories and Use Cases', Eds Alexander & Maiden, to be published by John Wiley;

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, Lecture Notes on Computer Science LNCS 2255, Springer-Verlag, 151-164.

Maiden N., Robertson S. & Gizikis A., 2004, 'Provoking Creativity: Imagine What Your Requirements Could be Like', IEEE Software, September/October 2004 21(5), 68-75

Maiden N.A.M. & Rugg G., 1996, 'ACRE: Selecting Methods For Requirements Acquisition, Software Engineering Journal 11(3), 183-192.

Morato J., Marzal M. A., Llorens J., and Moreiro J., 2004. "WordNet Application", Proceedings of GWC 2004. The Second Global Wordnet Conference 2004, Brno, Czech Republic, Junuary 20-23.

Navigli, R. and Velardi, P., 2004 "Structural Semantic Interconnection: a Knowledge-Based Approach to Word Sense Disambiguation", to appear in Proc. of SENSEVAL-3 Workshop (SENSEVAL), in the 42th Annual Meeting of the Association for Computational Linguistics (ACL 2004), Barcelona, Spain, July 25-26th, 2004.

Nuseibeh B & Easterbrook S. 2000, 'Requirements Engineering: A Roadmap', Proceedings IEEE International Conference on Software Engineering (ICSE-2000), 4-11 June 2000, Limerick, Ireland, ACM Press.

Robertson S. & Robertson J., 1999, 'Mastering the Requirements Process', Addison-Wesley-Longman.

Robinson W.N., 2003, 'Monitoring Web Service Requirements', Proceedings 11th International Conference on Requirements Engineering, IEEE Computer Society Press, 65-74.

Savigni A. & Tisato F., 2004, 'Requirements Monitoring in a Reflective Architecture', Proceedings, SoRE'2004 workshop, 12th IEEE International Conference on Requirements Engineering, Kyoto, Japan, September 2004.

SeCSE 2005, secse.eng.it.

Schütze, H., Pedersen, J.O., 1995 "Information retrieval based on word senses", in Proceedings of the Symposium on Document Analysis and Information Retrieval, 4: 161- 175, 1995.

van Lamsweerde, A. (2004), 'Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice', Proceedings 12th IEEE International Conference on Requirements Engineering, IEEE Computer Society Press, p. 4-7.

Sutcliffe A.G. & Mehanijev N., 2004, 'End-User Development', Communications of the ACM 47(9), 31-32.

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.