How to Analyse User Requirements for Service-Oriented Environmental Information Systems

Thomas Usländer and Thomas Batz

Fraunhofer IOSB, Fraunhoferstr. 1,
76131 Karlsruhe, Germany
{thomas.uslaender,thomas.batz}@iosb.fraunhofer.de

Abstract. Environmental Information Systems (EIS) allow the user to store, query and process environmental information and visualize it in thematic maps, diagrams and reports. Although service-orientation is the predominant architectural style of EIS there is no design methodology that brings together the requirements and the expert knowledge of EIS users with the services and information offerings of existing EIS, and, in addition, explicitly obeys the guidelines and constraints of geospatial standards of the Open Geospatial Consortium (OGC) as side-conditions. This paper focuses on the analysis phase as a prelude to service-oriented design. It proposes a way of gathering, describing and documenting user requirements in terms of extended use cases which may then be used to perform the abstract design step following SERVUS which denotes a Design Methodology for Information Systems based upon Geospatial Service-oriented Architectures and the Modelling of Use Cases and Capabilities as Resources.

Keywords: Service-oriented architecture, service-oriented analysis and design, requirements analysis, environmental information system, SERVUS.

1 Introduction

Environmental Information Systems (EIS) allow the user to store, query and process environmental information and visualize it in thematic maps, diagrams and reports [3]. More advanced functions cover advanced mapping functions of environmental data such as geospatial predictions and simulations (e.g. geo-statistical interpolation algorithms) or geospatial visual analytics techniques by combining data mining and information visualization techniques. In order to cope with this variety of functions, EIS are inherently componentized and distributed. This trend is also due to the need for collaboration between the various public and private stakeholders involved and various environmental science disciplines [1].

From the technological point of view EIS are information systems that deal with geospatial information and services with a reference to a location on the Earth. EIS are associated with heterogeneous sensors and/or environmental models that deliver measured or calculated observations about environmental phenomena. Basically, EIS provide a restricted view upon the environment which is limited by temporal, spatial and thematic boundaries. Information fusion is then enabled by a loose coupling of

J. Hřebíček, G. Schimak, and R. Denzer (Eds.): ISESS 2011, IFIP AICT 359, pp. 161–168, 2011.

EIS whereby the actual configuration of the resulting system-of-systems is dependent upon the environmental question to be answered.

These system requirements are best met by the principles of Service-oriented Architectures (SOA) [3] as a "framework for integrating business processes and supporting IT infrastructure as secure, standardized components – services – that can be reused and combined to address changing business priorities" [7]:

- SOA principles enable the sharing of geospatial information and services and their composition across organizational and administrative boundaries in a loosely-coupled but controlled manner. This is essential for EIS as environmental phenomena are not limited to boundaries drawn by humans.
- Effective and flexible interactions between EIS require an agreement within the developer community about the syntax and semantics of service interfaces and information models. Thus, as a crucial side-condition in the design of infrastructures for EIS, geospatial interoperability standards of ISO and the Open Geospatial Consortium (OGC) have to be considered.

There are several initiatives on national, European and world-wide scale that define geospatial SOAs as an underlying foundation for EIS, e.g. the Sensor Service Architecture (SensorSA) as a result of the European research project SANY [4]. However, one of the challenges is the design of applications that exploit the potential and the capabilities of such geospatial service networks. Software service engineering emerges as an own research discipline, strongly inheriting from the principles of software engineering, but enhancing them towards the open-world assumption of the SOA approach, i.e. a world of "unforeseen clients, execution contexts and usage" of services operating in "highly complex, distributed, unpredictable, and heterogeneous execution environments" [8]. Numerous methodologies for service-oriented analysis and design have been described in the literature and partly embedded in software development tools [2].

The deficiency today is that there is no design methodology that brings together the requirements and the expert knowledge of EIS users with the capabilities (i.e. services and information offerings) of existing EIS, and, in addition, explicitly obeys the guidelines and constraints of geospatial standards as side-conditions, which are, for the design of EIS, the reference model [10] and derived geospatial architectures of the OGC such as the SensorSA.

This paper describes a method for the co-development of requirements and capabilities of EIS (section 2) as a necessary prelude for a service-oriented design of EIS (section 3) and the analysis of user requirements (section 4). Section 5 concludes the paper with a project reference where this approach is being used.

2 Co-development of Requirements and Capabilities

Taking requirements and capabilities as a conceptual foundation, the kernel challenge for a service-oriented design of EIS boils down to the question of how the requirements of the user can be assessed against the already existing capabilities of service platforms. Basically, functional, informational and qualitative requirements at one

abstraction layer (A) have to be semantically matched to capabilities of another abstraction layer (B) taking side conditions into account (figure 1).

A pre-requisite to the matching is the discovery of possible capabilities which may fulfill the requirements. These are called *candidate capabilities*. The matching activity selects among the candidate capabilities those who fit best to the requirements, taking side conditions, such as the need to deliver standards-compliant services, explicitly into account. Discovery and matching need some associated semantic description of both requirements and capabilities in order to be effective. Such *semantic descriptions* give meanings to the terms used in the specifications, e.g. by means of semantic annotation to ontologies. Their representation forms range from text in combination with a glossary in which all important terms are defined for a given project up to specifications in description logics.

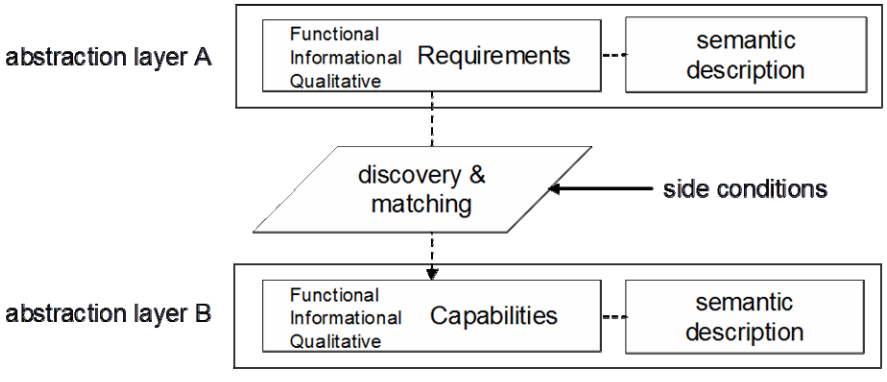

Fig. 1. Mapping of Requirements to Capabilities [3]

The discovery and matching problem repeatedly occurs when user requirements are broken down into multiple steps across several abstraction layers. In fact, capabilities turn into requirements for the next design step. Furthermore, there is a widely recognized need to package the development of requirement artifacts on a higher abstraction level and the development of architectural artifacts on a lower level into one single design step, leading to a so-called co-development of requirements (e.g. use cases, section 4) and architectural artifacts, e.g. service specifications or information models [9]. This requires a step-wise refinement of the design artifacts:

1. The **Analysis** step in which the user analyses the problem and expresses the outcome in the form of user requirements. Example: A use case that requests to "get a diagram containing the average nitrate concentration of the groundwater bodies in the Upper Rhine Valley of the last 10 years".
2. The **Abstract Design** step in which the user requirements are transformed by the system designer into system requirements which then have to be matched with the capabilities of an abstract service platform, i.e. a service platform that abstracts from the peculiarities of service platform technologies. Example: Provide a service that enables to "get observation values with a sampling time in the

interval [2000-01-01, 2009-12-31] for the environmental parameter "nitrate" for all groundwater monitoring stations that are located in the Upper Rhine Valley".

3. The **Concrete Design** step in which the capabilities of the abstract service platform turn into requirements for the design of the concrete service platform and finally result in a specification of its capabilities. Example: The *getObservation* operation request of the OGC Sensor Observation Service.
4. The **Engineering** step in which the specified capabilities of the concrete service platform have to be implemented as service components and deployed in the context of a service network.

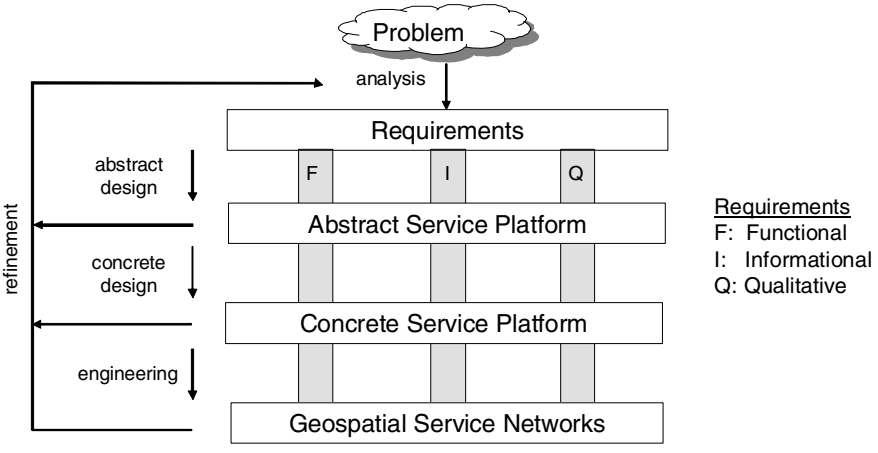

Fig. 2. Analysis, Design and Engineering Steps [3]

In the following a method for the *analysis* step is presented as a prelude of the SERVUS Design Methodology [3]. SERVUS focuses on the *abstract design* step as presented in the following section.

3 The SERVUS Design Methodology

SERVUS denotes a Design Methodology for Information Systems based upon Geospatial Service-oriented Architectures and the Modelling of Use Cases and Capabilities as Resources (SERVUS). SERVUS describes individual design activities that are interconnected by a common modelling environment interconnecting the Enterprise, Information and Service Viewpoint of geospatial architectures [10].

SERVUS relies upon a semantic resource model (see its UML specification in figure 3) as a common modelling language to which both use cases and capabilities may be mapped. Hereby, a resource is an information object that is uniquely identified, may be represented in one or more representational forms (e.g. as a diagram, XML document or a map layer) and support resource methods that are taken from a limited set of operations whose semantics are well-known (uniform interface). A resource has own characteristics (attributes) and is linked to other resources forming a resource

network. Furthermore, resource descriptions may refer to concepts of the domain model (design ontology) using the principle of semantic annotation, yielding so-called *semantic resources*. The basic idea of the SERVUS resource model is derived from the Representational State Transfer (REST) architectural style for distributed hypermedia systems as conceived by Fielding [5].

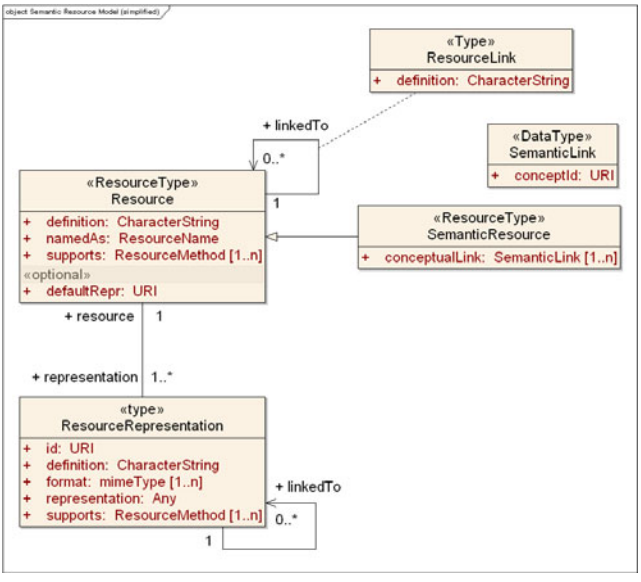

Fig. 3. SERVUS Semantic Resource Model [3]

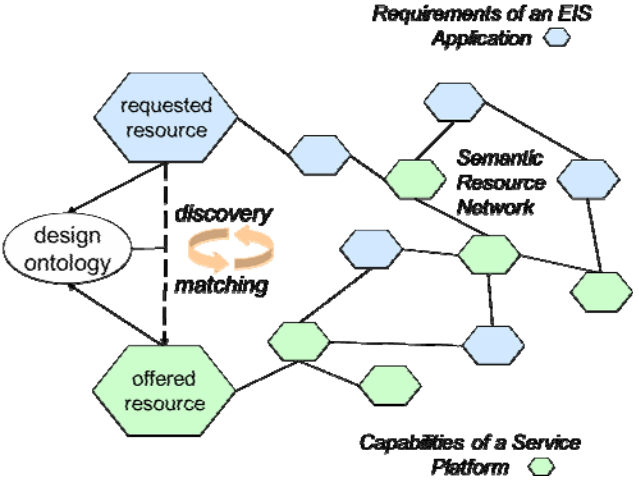

Fig. 4. Discovery and matching of requested and offered resources [3]

The abstract design step is then understood to be an iterative discovery and matching activity: *requested resources* derived from user requirements have to be mapped to fitting *offered resources* that represent the information objects being accessed and manipulated by geospatial services.

Hence, in the analysis step the set of *requested resources* has to be identified. This paper illustrates how user requirements may be captured through the identification and description of extended application uses cases.

4 Analysis of User Requirements

Figure 5 illustrates the analysis phase as a prelude of the SERVUS Design Methodology. As part of the project planning there needs to be some agreement of how to document use cases. For this continuous activity a project space has to be created which preferably should be supported by a project management server that is accessible by all participants of the analysis process.

As a first step of an analysis iteration loop a set of preliminary use cases (UC) is identified, mostly be those thematic experts who drive the project. For each of them an entry in the project space has to be generated. The methodology proposes that use

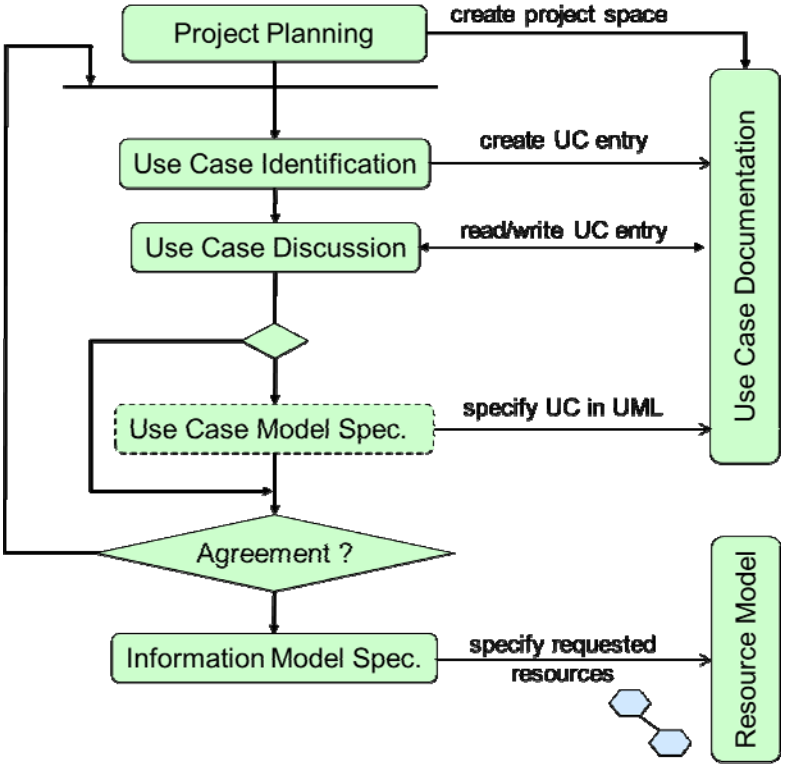

Fig. 5. Analysis phase of the SERVUS Methodology

cases are initially described in structured natural language but already contain the list of requested resources. This small extension with respect to the approach of Cockburn [11] heavily facilitates the transition to the abstract design step (here: the specification of the information model in UML) but is still very easy to understand by thematic experts. Hence, this description is the language which is used in the UC discussion that takes place in workshops that are facilitated by the system analyst. Depending on the level of agreement that can be reached the iteration loop is entered again in order to refine or add new use cases.

In order to identify inconsistencies and check the completeness of the UC model, the system analyst may transform the semi-structural UC description into formal specifications in the Unified Modelling Language (UML). However, these UML diagrams should still be on a high abstraction level such that a discussion with the end-user is possible. However, in addition to the usual UML use cases they already comprise the links to the set of requested (information) resources, their representation forms and the requirements to create, read, write or delete them. An example of such a use case diagram is contained in these proceedings [12].

Once an agreement is reached about the set of use case descriptions and related UML specifications it is then up to the system analyst to specify the resulting information model taking the (semantic) resource model (see above) as a modeling framework.

5 Conclusion and Outlook

The usability of the analysis method presented in this paper as a prelude to the SERVUS Design Methodology [3] has been validated in various facilitated workshops with users and software architects when analyzing user requirements for extensions of the EIS of the German federal state of Baden-Württemberg [6]. It is now applied to a related project that aims at analyzing the requirements for an information system that shall support the implementation and documentation of the Integrated Rhine Programme along the Upper Rhine valley in Germany [12]. These interdisciplinary projects comprise a multitude of stakeholders of different organizations. It can be stated that the application of this methodology heavily facilitates the discussion with the thematic experts and accelerates the consensus finding process. As a positive side effect, especially when being supported by a dedicated Web-based information management server, it decreases the burden of editing requirements analysis documents and helps in ensuring their consistency.

Service-orientation will be the major design paradigm for newly developed and future environmental information systems. However, in order to exploit its potential, it must be accompanied by powerful service-oriented design methodologies as a tool for system designer but also by adequate service-oriented analysis methodologies for system analysts that are tailored to the expertise and language of the thematic experts. It is the goal of the resource-oriented paradigm of the SERVUS Design Methodology to provide a modelling bridge between these two worlds of thinking.

References

1. Schimak, G. (ed.): Environmental Knowledge and Information Systems. Elsevier Special Issue 20(12) (2005)
2. Kohlborn, T., Korthaus, A., Chan, T., Rosemann, M.: Service Analysis - A Critical Assessment of the State of the Art. In: European Conference of Information Systems (ECIS 2009), Italy (2009)
3. Usländer, T.: Service-oriented Design of Environmental Information Systems. PhD thesis of the Karlsruhe Institute of Technology (KIT), Faculty of Computer Science. KIT Scientific Publishing (2010) ISBN 978-3-86644-499-7, http://digbib.ubka.uni-karlsruhe.de/volltexte/1000016721
4. Usländer, T. (ed.): Specification of the Sensor Service Architecture, Version 3.0 (Rev. 3.1). OGC Discussion Paper 09-132r1. Deliverable D2.3.4 of the European Integrated Project SANY, FP6-IST-033564 (2009)
5. Fielding, R.T.: Architectural Styles and the Design of Network-Based Software Architectures. Doctoral dissertation, University of California, Irvine (2000)
6. Keitel, A., Mayer-Föll, R., Schultze, A.: Framework Conception for the Environmental Information System of Baden-Württemberg (Germany). In: Hřebíček, J., et al. (eds.) Proceedings of Towards eEnvironment, pp. 461–468 (2009) ISBN 978-80-210-4824-9
7. Bieberstein, N., Bose, S., Fiammante, M., Jones, K., Shah, R.: Service-Oriented Architecture (SOA) Compass – Business Value, Planning and Enterprise Roadmap. IBM Press developerWorks Series (2006) ISBN 0-13-187002-5
8. van den Heuvel, W.J., Zimmermann, O., Leymann, F., Lago, P., Schieferdecker, I., Zdun, U., Avgeriou, P.: Software Service Engineering: Tenets and Challenges. In: Proceedings of ICSE 2009 Workshop - Principles of Engineering Service Oriented Systems (PESOS), IEEE Computer Society, Los Alamitos (2009)
9. Pohl, K., Sikora, E.: The Co-Development of System Requirements and Functional Architecture. In: Krogstie, et al. (eds.) Conceptual Modelling in Information Systems Engineering, pp. 229–246 (2007)
10. Percivall, G. (ed.). OGC Reference Model Version 2.0, Open Geospatial Consortium Document 08-062r4 (2008), http://orm.opengeospatial.org/
11. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, Reading (2001) ISBN-13: 9780201702255
12. Usländer, T., Junker, R., Pfarr, U.: Towards User Requirements for an Information System of the Integrated Rhine Programme. In: Hřebíček, J., Schimak, G., Denzer, R. (eds.) ISESS 2011. IFIP AICT, vol. 359, pp. 651–656. Springer, Heidelberg (2011)