# Facilitating Requirements Engineering of Semantic Applications

Óscar Muñoz-García, Raúl García-Castro, Asunción Gómez-Pérez

Ontology Engineering Group, Departamento de Inteligencia Artificial.
Facultad de Informática, Universidad Politécnica de Madrid, Spain
{omunoz,rgarcia,asun}@fi.upm.es

**Abstract.** This paper presents a detailed Requirements Engineering process for the development of semantic applications. It presents the activities for requirements elicitation and analysis and shows how to follow these activities in an example case study. To facilitate its use by software engineers that are not experts in semantic technologies, several resources are provided, namely a comprehensive collection of the characteristics of semantic applications and two catalogues of use cases and system models.

## 1 Introduction

A large-scale semantic application is a semantic application that makes use of semantic technologies and that manipulates huge quantities of heterogeneous decentralised knowledge that present different degrees of quality. It produces and consumes own and external data and retrieves knowledge automatically exploring different sources [1].

As a particular domain for semantic applications, the Semantic Web is a large-scale source of knowledge that requires designing applications which are very different from classic knowledge-based systems [1]. The next generation of Semantic Web applications needs to deal with significant problems associated with the scale, heterogeneity, and the varying degrees of quality of the information contained in the Semantic Web and other problems such as information provenance. These problems are not only appearing in the Semantic Web but also in other knowledge management systems or data interpretation systems.

Software engineers without expertise in the development of semantic applications do not know how to define or implement the semantic functionalities of applications and it is difficult for them to carry out the development process of these kind of applications. Thus, it is necessary to provide them a solution consisting on guidelines that they can easily adapt and integrate in their development processes.

This paper introduces a process that facilitates Requirements Engineering of semantic applications, that is, the process of finding out, analysing, documenting and checking the requirements of a semantic application. To this end, this paper defines a process for requirements elicitation and analysis of semantic applications, helping application developers during the first stages of building semantic applications from scratch or of including semantic components into traditional information systems.

This paper is structured as follows. Section 2 presents previous work about the characterisation of semantic applications and about existing scenarios of semantic applications. Section 3 specifies the proposed Requirements Engineering process with the tasks that compose it. Section 4 illustrates how to carry out the process with an example case study. Finally, Section 5 presents the conclusions of this work and future lines of work.

## 2 Semantic Applications

In order to elicit and analyse the requirements of a semantic application, it is necessary to understand the characteristics that commonly appear in semantic applications and the different scenarios where semantic solutions are applied.

A **characterisation of semantic applications** can be extracted taking into account the characteristics of this kind of applications presented in [1–4]. Table 1 shows the result of the characterisation according to the nature of the ontologies that the application uses, the data that the application produces and consumes, the kind of reasoning that the application applies, and the interoperability characteristics of the application with other systems.

In [5] a set of **scenarios for applying ontologies** in applications are presented. Next, these scenarios are summarised: *Neutral Authoring*, where an information artifact is authored in a single language and is converted into a different form to be used in multiple target systems; *Ontology as Specification*, where an ontology of a given domain is created and used as a basis for the specification and development of some software; *Common Access to Information*, where information is required by one or more persons or computer applications but is expressed using unfamiliar vocabulary or in an inaccessible format; and *Ontology-based Search*, where an ontology is used for searching an information repository for desired resources. In [6] there is a classification of type of usage of ontologies for Semantic Web applications from where several scenarios can be derived. In this work they present the following categories: *Usage as a Common Vocabulary*, *Usage for Search*, *Usage as an Index*, *Usage as a Data Schema*, *Usage as a Media for Knowledge Sharing*, *Usage for a Semantic Analysis*, *Usage for Information Extraction*, *Usage as a Rule Set for Knowledge Models* and *Usage for Systematizing Knowledge*. The work presented in [7] adds the scenario of *Collaborative Construction of Knowledge* to the presented ones.

## 3 Requirements Engineering in Semantic Applications

Requirements are the descriptions of the functionalities provided by the application and its operational constraints and reflect the needs of the customer for a system that helps solving some problem [8].

| **Ontologies Dimension** | |
|---|---|
| *Typology* | Specifies if the application is based on a single ontology or makes use of several ontologies that are organised in a network. |
| *Domain Openness* | Specifies if the ontologies are bound to a particular domain or not. |
| *Ubiquity* | Specifies if the ontologies are centralised in a single resource or distributed using data from multiple sources. |
| *Dynamicity* | Specifies if the ontologies are constantly changing or if on the contrary, they are gathered and engineered at design-time. |
| *Hugeness* | Specifies if the application can or cannot operate at scale in a context such as the Semantic Web with a huge number of ontologies. |
| *Heterogeneity* | Specifies if the ontologies are heterogeneous along several dimensions (encoding, quality, complexity, modelling, views, etc.) or, on the contrary, if the application is characterised by the use of ontologies that are selected, designed and integrated manually and carefully. |
| **Data Dimension** | |
| *Domain Openness* | Specifies if the data used is bound to a particular domain or not. |
| *Ubiquity* | Specifies if the data is centralised in a single resource or distributed using data from multiple sources. |
| *Dynamicity* | Specifies if the instances are constantly changing or if on the contrary they are gathered and engineered at design-time. |
| *Hugeness* | Specifies if the application can or cannot operate at scale in a context with a huge data quantity. |
| *Heterogeneity* | Specifies if the data is characterised by the heterogeneity on encoding or by the combination of semantic and non-semantic data. |
| **Reasoning Dimension** | |
| *Reasoning Type* | Specifies if the application will use heavyweight, lightweight or hybrid (machine learning, linguistic, statistical techniques, etc.) reasoning mechanisms. |
| *Contradictory Information* | Specifies the capability of the application to deal with contradictory information. |
| *Incomplete Information* | Specifies the capability of the application to deal with incomplete information. |
| **Systems Interoperability Dimension** | |
| *Legacy Interoperability* | Specifies if the application integrates heterogeneous proprietary and legacy systems. |
| *Distributed Services* | Specifies if the application will offer or consume distributed services in order to interoperate with other applications. |

**Table 1.** Characterisation of semantic applications

Sommerville [8] describes the *Requirements Engineering* process as a three-stage activity where the activities are organised as an iterative process around a spiral. The spiral process accommodates approaches to development in which the requirements are developed to different levels of detail. Agile methods, an approach to iterative development, follow this perspective.

The activities involved in the *Requirements Engineering* process are the following: (1) *Requirements elicitation and analysis*, in which the features that the system should provide, the required performance of the system, and so on, are discovered; (2) *Requirements specification*, in which the user and system requirements are specified; (3) *Requirements validation*, which shows that the requirements actually define the system that the customer wants.

This paper is focused on providing methodological guidelines for carrying out the *Requirements Elicitation and Analysis* activity. Methodological guidelines for the rest of the activities involved in the *Requirements Engineering* process are not provided because we have not specialised them with respect to existing software engineering methods.

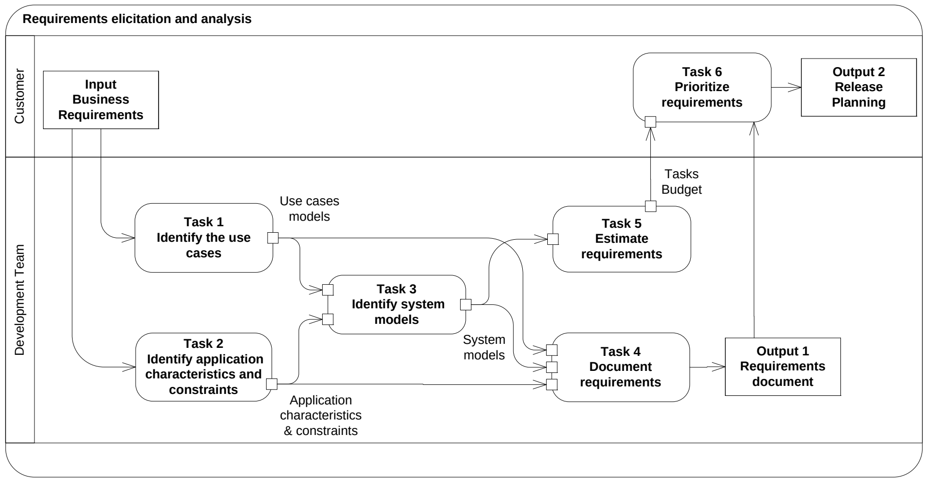
Requirements elicitation and analysis
Customer
Development Team
Input Business Requirements
Task 1 Identify the use cases
Use cases models
Task 2 Identify application characteristics and constraints
Application characteristics & constraints
Task 3 Identify system models
System models
Task 4 Document requirements
Task 5 Estimate requirements
Tasks Budget
Task 6 Prioritize requirements
Output 1 Requirements document
Output 2 Release Planning

**Fig. 1.** The requirements elicitation and analysis activity

The input of the *requirements elicitation and analysis activity* is a set of *Business Requirements* and its goal is to state the functionalities the system should provide as well as the constraints on the system and when it should be done. This is reflected in the *Software Requirements Document* and in the *Release Planning* respectively, the outputs of the activity. The requirements document is written incrementally at the beginning of the project and, as the project proceeds, in the beginning of every application release development.

The tasks for carrying out the requirements elicitation and analysis activity can be seen in the activity diagram shown Figure 1 and are explained below.

**Task 1. To identify the use cases.** The objective of this task is to gather information about the application from the business requirements facilitated by the customer and to distil use cases from this information.

Scenario-based elicitation and, in concrete, use cases are an appropriate approach to Requirements Engineering when implementing an agile method. However use cases are not as effective for eliciting constraints or high-level and non-functional requirements, as for example those related to the characteristics presented in Section 2.

In order to speed-up this task, we provide a catalogue of use cases that commonly appear in semantic applications. When performing this task, these common use cases can be selected, adapted and appended to the identified set of requirements. The use cases have been obtained by analysing the scenarios presented in Section 2 from the viewpoint of the system user goals. The use cases identified are the following: (1) *Query Information*, where the user goal is to obtain integrated information from several resources given a query; (2) *Search resources*, where the goal is to find resources related to a given search; (3) *Browse information*, where the user wants to navigate through categorised information; (4) *Analyse Information*, where the goal is to obtain some analysis

| **Use Case Template UCT2: Query Information from External Providers** | |
|---|---|
| ***Primary actor*** | The person or system who requests the System under development to deliver some information. |
| ***Stakeholders and interests*** | The *Primary Actor* requires some information that the System must gather and integrate from other Information Providers. The *Information Providers* provide to the System the information to be processed. |
| ***Preconditions*** | The *Primary Actor* can access to the System. |
| ***Success guarantee*** | The System returns to the *Primary Actor* the required information after integrating correctly the information obtained from the *Information Providers*. |
| ***Main success scenario*** | |
| 1. The *Primary Actor* requests to the System to deliver some integrated information. | |
| 2. The System requests to an *Information Provider* the information that requires from it. | |
| 3. The *Information Provider* responses to the System with the requested information. | |
| *Steps 2-3 are repeated until there are no more Information Providers to be requested.* | |
| 4. The System integrates the gathered information and returns to the *Primary Actor* the required information. | |
| ***Extensions*** | |
| *a. At any time, System fails: | |
| 1. The System informs to the *Primary Actor* about the failure. | |
| 2-3a. At any time the communication with the *Information Provider* fails. | |
| 1. System keeps requesting other *Information Providers* continuing in step 2. | |
| 1a. The information requested to the *Information Provider* is mandatory in order to give a correct response to the *Primary Actor*. | |
| 1. The System informs to the *Primary Actor* about the failure. | |

**Table 2.** *Process External Information* use case template

from collected information; (5) *Extract Information*, where the goal is to extract meaningful information by processing a search result; and (6) *Manage Knowledge* where the aim is to collaboratively construct and evolve shared knowledge.

As an example, Table 2 shows a variation of the *Query Information* use case, whose template has been extracted from the catalogue of common use cases.

**Task 2. To identify application characteristics and constraints.** The objective of this task is to collect non-functional requirements, that is, constraints on the services or functions offered by the system.

In order to speed-up this task, we provide a set of characteristics that commonly appear in semantic applications. When performing this task these common characteristics can be selected and appended to the identified set of requirements. The common characteristics have been obtained through an analysis of the State of the Art, are summarised in Section 2, and have been classified according to four dimensions regarding the nature of the *ontologies* and *data* that the application will use, the kind of *reasoning* that the application will apply and the *interoperability* of the application with other systems.

**Task 3. To identify system models.** The objective of this task is to preliminary specify the system in form of system models, which are an important bridge between the requirements engineering and the design processes because they are often more understandable than detailed natural language descriptions of the system requirements [8]. The system models will reflect the scenarios identified during the use case identification task, constrained by the application characteristics.

The system models allow representing the system from the following different perspectives: (1) an external perspective, where the context or environment of the application is modelled by showing the limits of the system where the

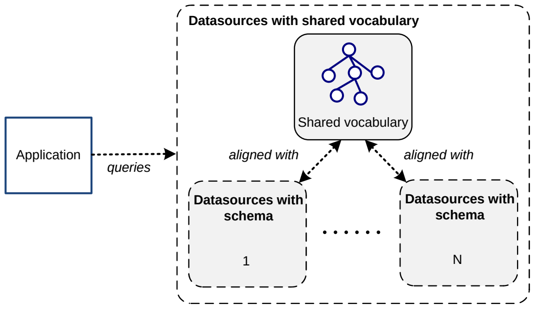

**Fig. 2.** Example of a system model template

application to develop will be deployed as well as the external systems; and (2) a structural perspective, where the structure of the data processed by the system is modelled. According to the process presented in this paper, the scenarios described in Section 2 can be viewed as system models.

In order to speed-up this task, we provide a catalogue of system model templates. When performing this task, these system model templates can be selected according to the identified use cases and the application characteristics and constraints. As an example, Figure 2 shows a the template used for an application that queries information to a group of data sources that are aligned with a shared vocabulary.

**Task 4. To document requirements.** In this task, the requirements discovered in the previous tasks are consolidated in a single description as the official statement of what the application developers should implement. The result of this task is the requirements document.

**Task 5. To estimate requirements.** In this task, the effort needed for implementing a set of requirements is estimated. As happens in Extreme Programming [9], the developers work together to break the system models down into development tasks. The result of this task reflects the estimation cost for each identified task.

**Task 6. To prioritise requirements.** In this task, the customer prioritises the development tasks estimated in Task 5. The output of this task is the release planning that reflects the system models to be implemented for the next application release.

## 4 Example Case Study

This section presents an example showing how to carry out the first tasks of the *Requirements Elicitation and Analysis* activity, given a fictitious case study.

**Business Requirements.** A logistics company has proved that setting dynamic shipment routes will decrease their shipment risks and delivery time and will increase its income because of factors such as weather, the transport companies availability and fares, etc.

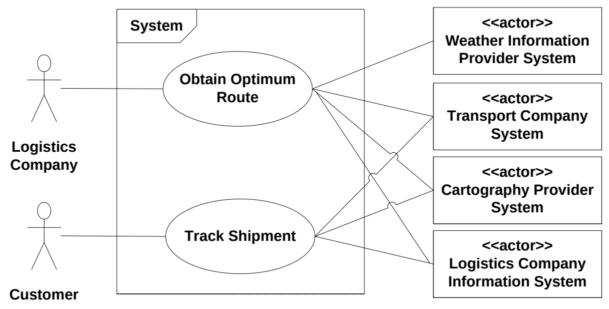
System
Obtain Optimum Route
Track Shipment
Logistics Company
Customer
<<actor>> Weather Information Provider System
<<actor>> Transport Company System
<<actor>> Cartography Provider System
<<actor>> Logistics Company Information System

**Fig. 3.** Identified use cases

The company wants to upgrade its system in order to enable intelligent search of optimum routes taking into account weather information coming from different internet providers and information owned by transport companies such as delivery times, transportation costs, availability of service for a certain route stretch, etc. The candidate routes are obtained from cartographies available in the Web.

Besides the search for the most adequate routes and transport companies, the logistics company wants to make use of the aforementioned integrated information to provide its clients with real time tracking about their shipments.

The information that the new application will use is encoded according to different formats. The weather information providers expose their information in XML according to a given XML schema. The transport companies provide a set of Web services in order to facilitate the interoperability with the logistics company. The cartographies are published in the Semantic Web as instances that conform to a given ontology. The logistics company will also use information stored in a relational database coming from its own information systems.

There are several information providers of each type: weather information, transport companies and cartography providers. None of the different XML schemas, Web service descriptors or ontologies that these providers use to specify its information formats are unique, that is, each provider models its information according to different criteria.

Given this set of *Business Requirements*, the three first tasks of an hypothetical *Requirements Elicitation and Analysis* episode are summarised next.

**Task 1. Identify the Use Cases.** Taking into account the *Business Requirements* facilitated by the logistics company, the development team starts identifying the use cases, finding the two use cases shown in Figure 3.

The first use case, *Obtain Optimum Route*, identifies the individual interactions between the logistics company system and the different external systems when obtaining an optimum route. The purpose of second use case, *Track Shipment*, is to show the interactions between the customer of the logistics company with the system and the interactions of the system with the external information provider systems.

Both of the use cases can be seen as realisations of the *Query Information* use case presented in Section 3. The template in the catalogue has to be instantiated by the development team by identifying the concrete primary actor and the set of stakeholders including the external systems and modifying the flow of the main success scenario and extensions.

**Task 2. Identify Application Characteristics and Constraints.** With respect to the *Ontologies Dimension*, the application is characterised by the consumption of own and foreign ontologies. Thus, the application is not bound to the particular domain of the logistics company. The ontologies also are not centralised in a single resource but distributed in multiple sources so there exists *Ontologies Ubiquity*. With respect to the *Ontologies Dynamicity* characteristic, it is not specified in the business requirements if the ontologies are constantly changing, so it is supposed that the ontologies are gathered and engineered at design time. With respect to the *Ontologies Hugeness* characteristic, the application will not operate at scale with a huge number of ontologies. Because the application will deal with different ontologies produced by different organisations, these ontologies may have different encodings (e.g., RDF(S) or OWL), they can exhibit differences in quality, computational heterogeneity (lightweight and heavyweight ontologies), they can be modelled according to different criteria and they will exhibit conceptual heterogeneity that should be solved with the use of mappings; thus, the *Ontologies Heterogeneity* characteristic is guaranteed. The use of different ontologies that somehow have to be interrelated will lead to a network-of-ontologies typology.

With respect to the *Data Dimension*, the application is characterised by the consumption of own and foreign data. In consequence, there exists *Data Domain Openness*. As happened with the ontologies, the data is not centralised in a single resource , that is, there exists *Data Ubiquity*. Also, the application is characterised by the use of dynamic data that is constantly changing so there exist *Data Dynamicity*. The data is gathered, selected, combined and processed at run-time. With respect to the *Data Hugeness* characteristic, the application has to operate at scale with a huge number of instances. There is *Data Heterogeneity* because data is encoded in different forms that are semantic and non-semantic.

With respect to the *Reasoning Dimension*, the development team can advance that heavyweight logical reasoning is not a key aspect of the application. Because the application has to deal with the heterogeneity of ontologies and instances explained before, the reasoning has to be hybrid and enabled by the combination of lightweight logical reasoning and other kinds of reasoning. Besides the application should deal with contradictory information coming from different cartography providers, contradictory weather previsions, etc. And, furthermore, it is not guaranteed that the information used will be always complete because the system will process information from different providers that the logistics company does not own.

Finally, with respect to the *System Interoperability Dimension*, the system will interoperate with legacy databases and will use the distributed services provided by the transport companies.

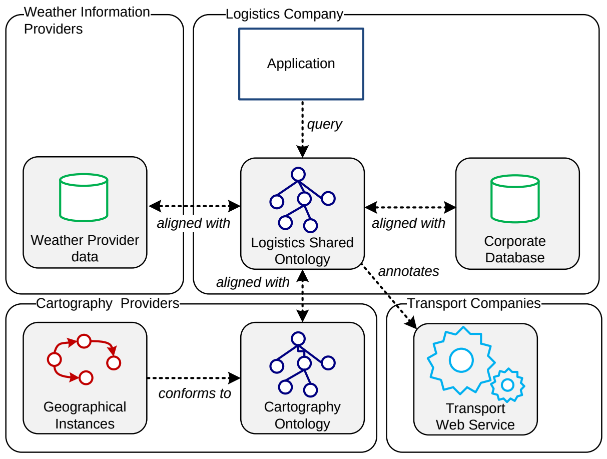

**Fig. 4.** Identified system model

**Task 3. Identify System Models.** As an example, the system model sketched for the first use case is the one shown in Figure 4. The different information sources in every stakeholder system are represented: the weather information providers have XML data that conforms to certain XML schemas; the cartography providers own a set of semantic geographical data that conforms to given cartography ontologies; the transport companies publish on the Web a set of services used to interoperate with their own systems; and the logistics company owns a database with information generated by a legacy system.

## 5 Conclusions and Future Work

In this paper we have introduced a method for carrying out the *Requirements Elicitation and Analysis* activity for semantic applications. We have defined the tasks involved within the activity in the context of an iterative development lifecycle. This is particularly useful when application requirements are changing continuously during the whole application construction.

For this, we have adapted several requirement elicitation techniques taken from Software Engineering. We provide catalogues of general use cases and system models in order to be adapted by the software engineer during the requirements process. In addition, a set of applications characteristics and constraints that commonly appear in semantic applications have been provided and are intended to help the developers in identifying the semantic requirements of the application under development. The set of application characteristics and the catalogues of use cases and system models are being defined within the NeOn[1] project. They will be available in a collaborative public space in the future in

[1] http://www.neon-project.org/

order to be commented, specified and populated. The evaluation of the results is intended to be done through the aforementioned community website as well as within the real-world NeOn case studies.

The use cases and system models produced following this process allow an easy understanding of the system requirements for users and developers while incorporating the semantic particularities of the application.

The definition of the *Requirements Engineering* process is the first stage in the construction of a rigorous and sound methodology for the development of a new generation of large-scale semantic applications. This methodology will provide the necessary framework to organise and manage the development of semantic applications and will be specially focused on facilitating the use of semantic technologies to software engineers. The next steps will be to define the *Design Process*, wich will lead to the definition of the architecture of a semantic application using reusable components, and to evaluate the whole methodology within several case studies in the context of the NeOn project.

**Acknowledgements**

This work has been partially supported by the NeOn project (IST-2005-027595).

## References

1. Aquin, M., Motta, E., Sabou, M., Angeletou, S., Gridinoc, L., Lopez, V., Guidi, D.: Towards a New Generation of Semantic Web Applications. IEEE Intelligent Systems **23** (2008)
2. Motta, E., Sabou, M.: Next Generation Semantic Web Applications. In: 1st Asian Semantic Web Conference, Beijing (2006)
3. Domingue, J., Fensel, D.: Towards a Service Web: Integrating the Semantic Web and Service Orientation. IEEE Intelligent Systems (2008)
4. Krummenacher, R., Simperl, E., Fensel, D.: Scalability in Semantic Computing: Semantic Middleware. In: Proceedings of the IEEE Conference on Semantic Computing. (2008) 538–544
5. Jasper, R., Uschold, M.: A Framework for Understanding and Classifying Ontology Applications. In: Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW 99. (1999)
6. Kozaki, K., Hayashi, Y., Sasajima, M., Tarumi, S., Mizoguchi, R.: Understanding Semantic Web Applications. In: 3rd Asian Semantic Web Conference (ASWC 2008). (2008)
7. Coskun, G., Heese, R., Luczak-Rösh, M., Oldakowski, R., Schäfermeier, R., Streibel, O.: Towards Corporate Semantic Web: Requirements and Use Cases. Technical report, Freie Universität Berlin (2008)
8. Sommerville, I.: Software Engineering. Eighth edn. International Computer Science Series. Addison-Wesley (2007)
9. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional (1999)