Guidelines for the Specification and Design of Large-Scale Semantic Applications

Óscar Muñoz-García and Raúl García-Castro

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

Abstract. This paper presents a set of guidelines to help software engineers with the specification and design of large-scale semantic applications by defining new processes for *Requirements Engineering* and *Design* for semantic applications. To facilitate its use to software engineers not experts in semantic technologies, several techniques are provided, namely, a characterization of large-scale semantic applications, common use cases that appear when developing this type of application, and a set of architectural patterns that can be used for modelling the architecture of semantic applications. The paper also presents an example of how these guidelines can be used and an evaluation of our contributions using the W3C Semantic Web use cases.

1 Introduction

A large-scale semantic application is an application that makes use of semantic technologies and that manipulates huge quantities of heterogeneous decentralized knowledge and semantic data presenting different degrees of quality. The application produces and consumes its own and external data and retrieves knowledge automatically by exploring different sources.

As a particular domain for large-scale semantic applications, the Semantic Web is a large-scale source of knowledge that requires to design a new generation of Semantic Web applications, which are very different from classic knowledge-based systems (KBS) [1]. In classic KBS the ontologies (usually one) and instances are bound to a particular domain. On the other hand, the next generation of Semantic Web applications permits the execution of the applications in multiple domains, integrates heterogeneous proprietary and legacy solutions, and makes use of big networks of ontologies. In addition, the next generation of Semantic Web applications needs to deal with significant problems associated with the scale, heterogeneity, interoperability and distribution of the information processed, such as the need for searching, accessing and integrating the appropriate knowledge according to the task at hand [1]. These problems do not appear in the Semantic Web but also in other knowledge management or data interpretation systems.

On the other hand, software engineers without expertise in the development or use of semantic applications do not know how to define or implement the semantic functionalities of applications, and therefore, it is difficult for them to carry out the development process of these types of applications.

Software development methodologies are broadly used in Software Engineering and Knowledge Engineering. Nevertheless, while there are methodologies that support the development of data models (i.e., ontologies) for semantic applications [2,3], there are no methodologies that support the development of such applications. Since semantic applications are a subset of software applications, they could be built by applying any general-purpose software development methodology. However, a set of guidelines that specifically deals with large-scale semantic applications will lead to a more efficient development of these types of applications.

Our goal in the present paper is to provide guidelines for the requirements analysis and architectural design of a new generation of practical, large-scale semantic applications that draw on contextualized networked ontologies, heterogeneous data and other knowledge-level resources. The guidelines can be easily adapted and integrated in existing development processes by application developers whose aim is to design the architecture of semantic applications from scratch or to include semantic components into traditional information systems.

To do so, we have extended the work presented in [4] by refining the process and techniques presented there, and by defining a new process and technique for designing the architecture of a large-scale semantic application.

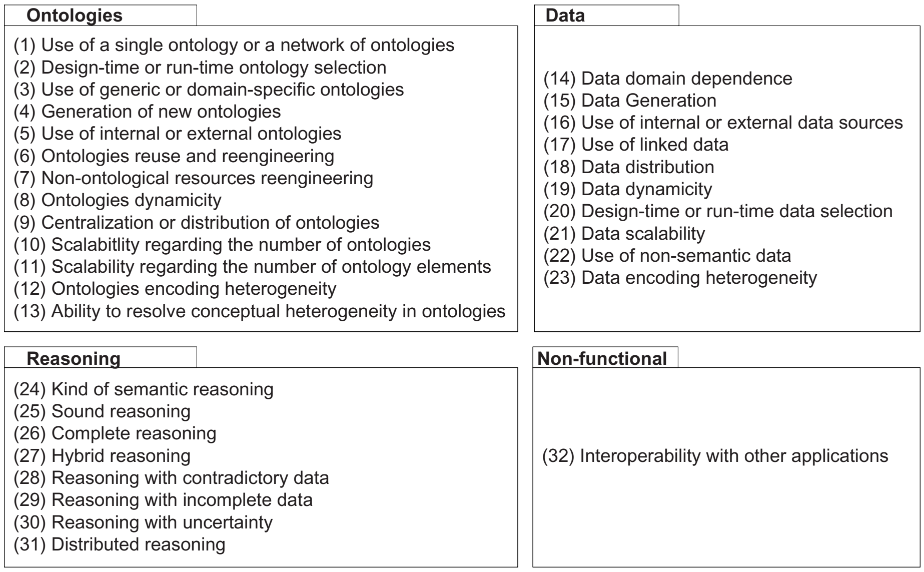
The main research results here described are the definition of the *Requirements Engineering* and the *Design* processes for semantic application development and the associated techniques for carrying out these processes, namely, a set of questionnaires for identifying the semantic requirements of the application being developed, catalogues of common use cases that appear when developing these types of applications, system models for understanding the context of the application, and patterns used for modeling the architecture of the application.

This paper is structured as follows. Section 2 presents previous work from which the guidelines are based. Section 3 provides an overview of the guidelines, while sections 4 and 5 detail the activities covered by the guidelines. Section 6 illustrates the guidelines with an example application, and Section 7 shows the results obtained after evaluating the guidelines. Finally, Section 8 presents the conclusions of this work and future lines of research.

2 Related Work

In order to elicit and analyse the requirements of a semantic application, it is necessary to understand the characteristics that commonly appear in such applications and the different scenarios where semantic solutions are applied. Besides, to obtain the architectural design of large-scale semantic applications it is also necessary to define, among others, a set of independent components commonly used in semantic applications.

We have extracted a **characterization of semantic applications** regarding the characteristics of this type of applications presented in [1,5,6,7]. Figure 1



Ontologies

(1) Use of a single ontology or a network of ontologies
(2) Design-time or run-time ontology selection
(3) Use of generic or domain-specific ontologies
(4) Generation of new ontologies
(5) Use of internal or external ontologies
(6) Ontologies reuse and reengineering
(7) Non-ontological resources reengineering
(8) Ontologies dynamicity
(9) Centralization or distribution of ontologies
(10) Scalabitlity regarding the number of ontologies
(11) Scalability regarding the number of ontology elements
(12) Ontologies encoding heterogeneity
(13) Ability to resolve conceptual heterogeneity in ontologies

Data

(14) Data domain dependence
(15) Data Generation
(16) Use of internal or external data sources
(17) Use of linked data
(18) Data distribution
(19) Data dynamicity
(20) Design-time or run-time data selection
(21) Data scalability
(22) Use of non-semantic data
(23) Data encoding heterogeneity

Reasoning

(24) Kind of semantic reasoning
(25) Sound reasoning
(26) Complete reasoning
(27) Hybrid reasoning
(28) Reasoning with contradictory data
(29) Reasoning with incomplete data
(30) Reasoning with uncertainty
(31) Distributed reasoning

Non-functional

(32) Interoperability with other applications

Fig. 1. Characteristics of Large-scale Semantic Applications

shows the result of the analysis made. As can be observed, we have clustered the characteristics according to the nature of the ontologies used, the data produced and consumed, the kind of reasoning applied, and other non-functional characteristics.

In [8], the following set of **scenarios for applying ontologies** to applications is presented: (1) *Neutral Authoring*, where an information artefact is authored in a single language and converted into a different form so that it can be used in multiple target systems; (2) *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; (3) *Common Access to Information*, where information is required by one or more persons or by computer applications; this information, however, is expressed in unfamiliar vocabulary or in an inaccessible format; and (4) *Ontology-based Search*, where an ontology is used for searching an information repository for desired resources.

In [9] there is a classification of the type of ontology usage in Semantic Web applications from where several scenarios can be derived: (1) *Usage as a Common Vocabulary*, (2) *Usage for Search*, (3) *Usage as an Index*, (4) *Usage as a Data Schema*, (5) *Usage as a Media for Knowledge Sharing*, (6) *Usage for a Semantic Analysis*, (7) *Usage for Information Extraction*, (8) *Usage as a Rule Set for Knowledge Models*, and (9) *Usage for Systematizing Knowledge*. The work presented in [10] adds the scenario of *Collaborative Construction of Knowledge* to those here presented.

The **Semantic Web Framework** (SWF) [11] is a component-based framework from which Semantic Web applications can be organized and developed;

this framework provides the skeleton for the specification of the independent components needed for the component-based engineering of Semantic Web applications. The SWF describes the functionalities that the components of Semantic Web applications provide and require, classifies these components, and identifies the main dependences between them. The SWF components are defined at the conceptual level and are decoupled of the technology that implements such components.

3 Overview of the Processes Described

The main objective of the guidelines here presented is to lead application developers from the elicitation of semantic application requirements to the description of the architecture of pure large-scale semantic applications, as well as to the description of the semantic part of applications that include semantic components. To achieve such a goal, we have described the *Requirements Engineering* and *Design* processes bearing in mind the development processes defined for Component Based Software Engineering [12] and the agile methods employed in software development. Figure 2 shows an overview of the overall process.

During the *Requirements Engineering* process, the requirements of the application must be analysed, agreed and documented. On the other hand, *Design* is the process of describing the structure of the software to be implemented and the interfaces between system components [12]. These processes cover different

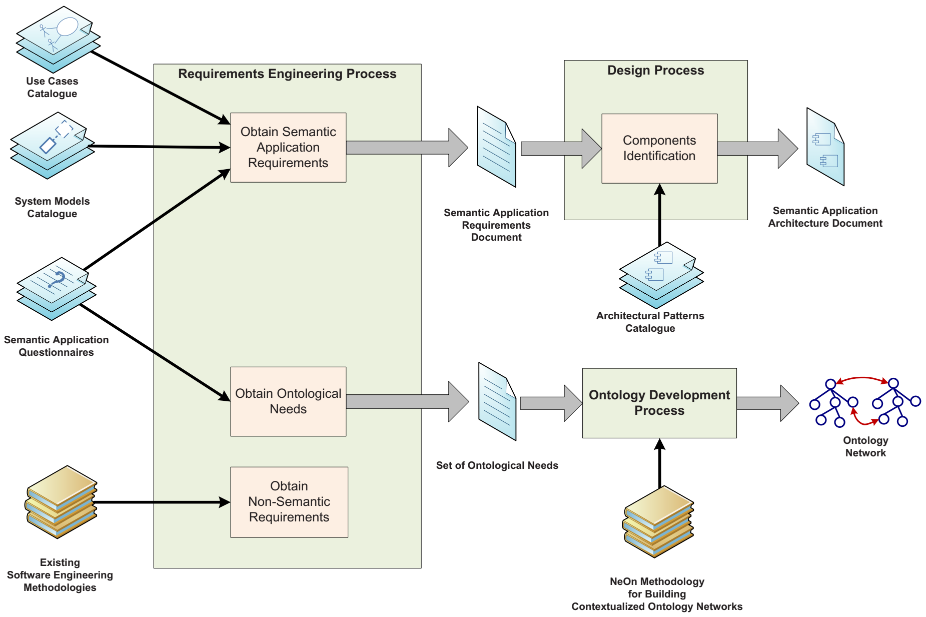

Use Cases Catalogue
System Models Catalogue
Semantic Application Questionnaires
Existing Software Engineering Methodologies
Requirements Engineering Process
Obtain Semantic Application Requirements
Obtain Ontological Needs
Obtain Non-Semantic Requirements
Semantic Application Requirements Document
Set of Ontological Needs
Design Process
Components Identification
Architectural Patterns Catalogue
Semantic Application Architecture Document
Ontology Development Process
NeOn Methodology for Building Contextualized Ontology Networks
Ontology Network

Fig. 2. Overview of the *Requirements Engineering* and *Design* processes

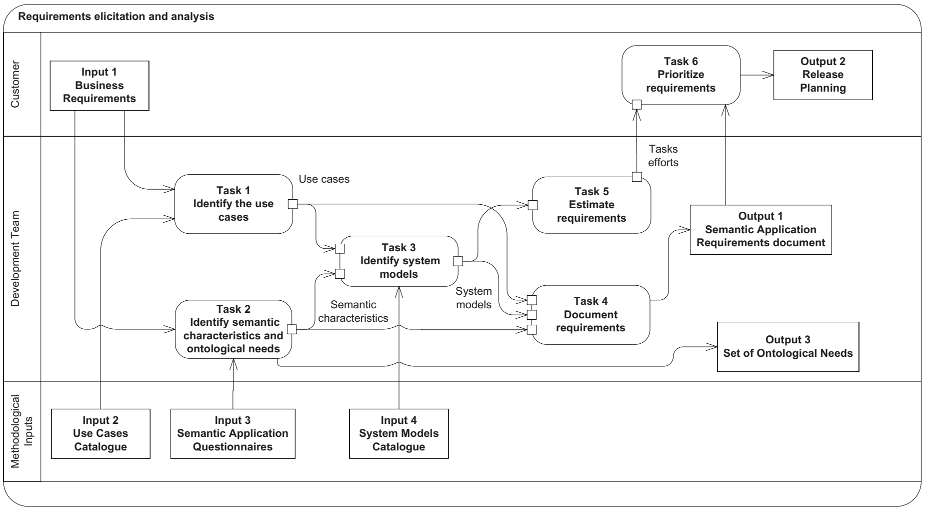

Requirements elicitation and analysis
Customer
Development Team
Methodological Inputs
Input 1 Business Requirements
Task 6 Prioritize requirements
Output 2 Release Planning
Tasks efforts
Use cases
Task 1 Identify the use cases
Task 5 Estimate requirements
Output 1 Semantic Application Requirements document
Task 3 Identify system models
System models
Semantic characteristics
Task 2 Identify semantic characteristics and ontological needs
Task 4 Document requirements
Output 3 Set of Ontological Needs
Input 2 Use Cases Catalogue
Input 3 Semantic Application Questionnaires
Input 4 System Models Catalogue

Fig. 3. Description of the *Requirements Elicitation and Analysis* activity

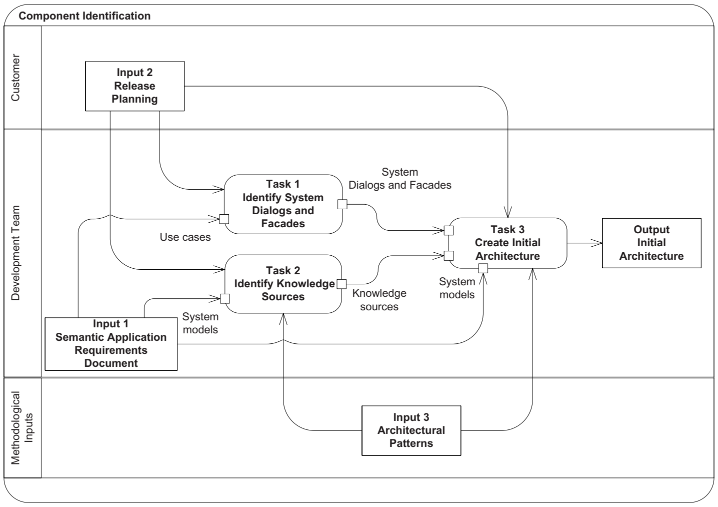

Component Identification
Customer
Development Team
Methodological Inputs
Input 2 Release Planning
Task 1 Identify System Dialogs and Facades
System Dialogs and Facades
Use cases
Task 3 Create Initial Architecture
Output Initial Architecture
Task 2 Identify Knowledge Sources
Knowledge sources
System models
System models
Input 1 Semantic Application Requirements Document
Input 3 Architectural Patterns

Fig. 4. Description of the *Component Identification* activity

activities. In this paper, however, we only provide guidelines for carrying out the *Requirements Elicitation and Analysis* and *Component Identification* activities included in the *Requirements Engineering* and *Design* processes, respectively. Other activities can be carried out by following any of the current software development methodologies. Figures 3 and 4 show a high level description of the activities commented above. Next, we summarize each of these activities. A complete description of the guidelines can be found in [13].

4 Guidelines for Requirements Elicitation and Analysis

To facilitate the requirement analysis, our guidelines propose that the requirements be divided into three different types: (1) the **Non-semantic Requirements** gather the application requirements not related to semantic functionalities; (2) the **Semantic Application Requirements** bring together the software requirements that tackle the semantic functionalities of the application; (3) finally, the **Set of Ontological Needs** reflects the ontological needs to be taken into account when developing the ontologies required by the semantic application. Such ontologies can be constructed following the guidelines given by any ontology development methodology, as, for example, the one described in [14]. Since any software engineering methodology supports the discovery of non-semantic requirements, our guidelines only provide techniques for obtaining the last two groups of requirements.

Semantic Application Questionnaires. Accompanying the guidelines, we provide a set of questionnaires that can be used by application developers for identifying the semantic characteristics of a given application (see Figure 1). In these questionnaires, each characteristic is covered with one question. The questionnaires also serve to identify the set of ontological needs and the data sets used by application.

Use Cases Catalogue. A catalogue of use case templates is also supplied. The catalogue describes the scenarios commonly appearing in semantic applications, such as the performance of a search based on ontologies or the semantically browsing of resources. Each template is graphically represented using UML 2.0 use case diagrams and includes detailed textual descriptions. The use case templates have been abstracted from the scenarios analysed in Section 2 (see Table 1). For identifying the use cases the following guidelines are provided: (1) to select the appropriate template from the catalogue; (2) to adapt the selected by modifying the use case information fields; and (3) to append the use case to the application requirements.

System Models Catalogue. We also provide a catalogue of system models. System models are graphical representations commonly used in Software Engineering that describe business processes, the problem to be solved, and the

Table 1. Mapping between the use cases and the scenarios analysed

	State of the art scenarios	
Use case	**Scenario in [8]**	**Scenario in [9]**
1. Query Information	3. Common access to Information	4. Usage as a Data Schema 5. Usage as a Media for Knowledge Sharing
2. Search Resources	4. Ontology-Based Search	2. Usage for Search
3. Browse Resources		3. Usage as an Index
4. Extract Information		7. Usage for Information Extraction 6. Usage for a Semantic Analysis
5. Manage Knowledge	1. Neutral Authoring 2. Ontology as Specification	1. Usage as a Common Vocabulary 8. Usage as a Rule Set for Knowledge Models 9. Usage for Systematizing Knowledge

system that is to be developed [12]. In our case, the system models let application developers to preliminarily specify the system from (1) an external perspective, where the context or environment of the application is modelled by showing the limits of the application and the external systems or applications that will interoperate with the application, and (2) a structural perspective, where the structure of the ontologies and the data processed by the application are modelled.

The system models catalogue contains a set of basic symbols (e.g., ontological and non ontological resources, applications) and the relationships between these symbols, which reflect the aforementioned structural perspective of the system. The system models will reflect the scenarios identified during the use case identification task, which is constrained by the application characteristics.

Figure 5 shows an example of a system model template that represents multiple data sources expressed according to several ontologies or non-ontological schemas and aligned with a shared vocabulary. The template has been obtained from the different approaches to ontology-based integration of information described in [15]. Figure 6 illustrates an example of an instantiation of the template shown in Figure 5, where several data sources that conform to an ontology or to a non-ontological schema are integrated through a shared ontology. Also in Figure 6, there are an ontology and a set of instances that are discovered at run-time (e.g., in the Semantic Web).

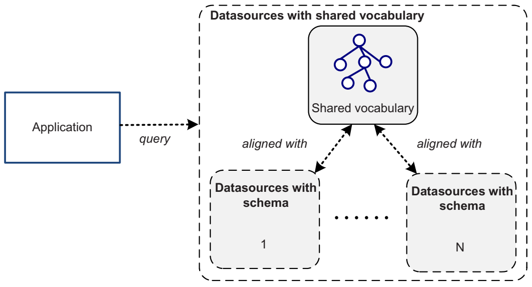


Fig. 5. Template example: *Query Information with a Hybrid Ontology approach*

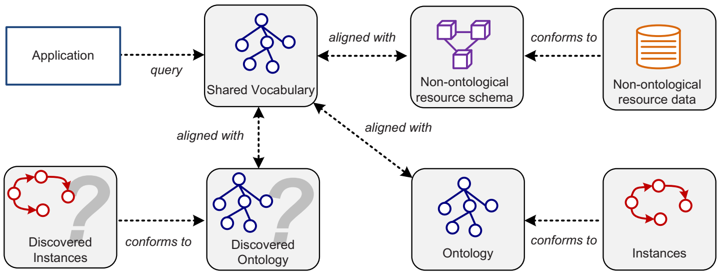


Fig. 6. Example of an instantiation of the system model template in Figure 5

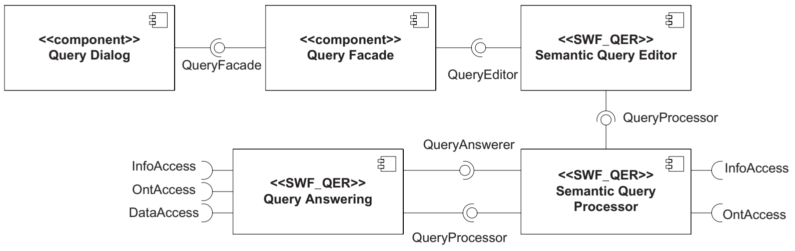

<<component>> Query Dialog
QueryFacade
<<component>> Query Facade
QueryEditor
<<SWF_QER>> Semantic Query Editor
QueryProcessor
InfoAccess
OntAccess
DataAccess
<<SWF_QER>> Query Answering
QueryAnswerer
QueryProcessor
<<SWF_QER>> Semantic Query Processor
InfoAccess
OntAccess

Fig. 7. Pattern example

For elaborating the system model we provide the following guidelines: (1) to associate a basic symbol to each of the resources that the application will deal with; (2) to identify the existing relationships between the basic symbols; (3) to identify the system model templates associated to the identified use cases and application characteristics; (4) to combine the symbols, relationships, and system model templates in order to conform a unique system model.

5 Guidelines for Component Identification

Our guidelines are focused on obtaining the structure (i.e., the architecture) of the semantic application using the technique explained next.

Architectural Patterns Catalogue. This catalogue provides 28 architectural patterns that reflect common organizations of semantic-related software components in large-scale semantic applications; the patterns were obtained from the analysis of the architecture of existing applications such the ones described in [11]. The components in the architectural patterns are those described in the SWF [11]. The patterns are represented as UML 2.0 component diagrams.

During the *Component Identification* activity, the patterns are selected regarding each of the symbols, relationships and templates used to depict the application system model.

Figure 7 shows an example of an architectural pattern used for solving the scenario depicted in Figure 5. In the pattern, the *Semantic Query Editor* component takes care of all issues related to the user interface. The *Semantic Query Processor* component is in charge of all the issues related to the physical processing of a query, while the *Query answering* component is responsible for all the issues related to the logical processing of a query. The *Query Dialog* component implements the *Query Information* use case logic, and the *Query Facade* component provides the operations to meet the use case responsibilities.

6 Example

This section presents an example of how to carry out the *Requirements Elicitation and Analysis* and *Component Identification* activities; the example is drawn from a fictitious case study whose *Business Requirements* are explained next:

A logistics company has proved that setting dynamic shipment routes will decrease their shipment risks and delivery time, while it will increase its income due to factors such as weather, transport companies availability and fares, etc.

The company wants to upgrade its system to enable intelligent search of optimal routes. To do this, the system will take into account weather information coming from different Internet providers and information owned by transport companies, for example, delivery times, transportation costs, and availability of service for a certain route stretch. The candidate routes are obtained from maps available on the Web. Besides searching 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 of their shipments.

The information that the new application will use is encoded according to different formats: the weather information providers expose their information as instances expressed according to a given ontology; the transport companies provide a set of XML resources to facilitate the interoperability with the logistics companies; and the maps are published in the Semantic Web formats. Additionally, the logistics company will also use information stored in a relational database included in its own information system.

The logistics company works with several known transport companies. The information about weather previsions will be discovered at run-time and integrated with the rest of the information.

6.1 Requirements Elicitation and Analysis Activity

This subsection presents how to carry out the three first tasks of a *Requirements Elicitation and Analysis* episode starting from the *Business Requirements*.

Task 1. To Identify the Use Cases. The development team starts by identifying the use cases and then finding the two use cases that are shown in Figure 8.

The purpose of first use case, *Obtain Optimum Route*, is to identify the interactions between the logistics company and the different external systems when an optimum route is obtained, whereas the purpose of the 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 use cases can be seen as realizations of the use

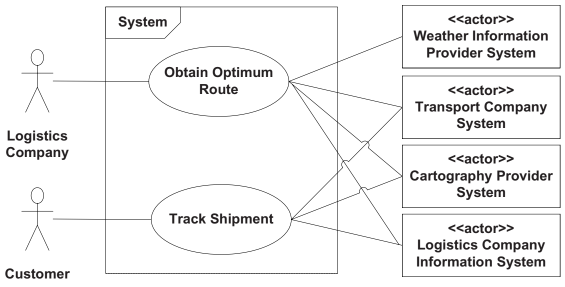

System
Obtain Optimum Route
Track Shipment
Logistics Company
Customer
<<actor>> Weather Information Provider System
<<actor>> Transport Company System
<<actor>> Cartography Provider System
<<actor>> Logistics Company Information System

Fig. 8. Use cases identified for the sample case study

case template *Query Information*, contained in the catalogue. The template in the catalogue has to be instantiated by identifying the primary actor and the stakeholders, including the external systems, and by modifying the flow specified in the template.

Task 2. To Identify Application Characteristics and Ontological Needs. As previously seen, the set of characteristics that commonly appear on semantic applications is intended to help developers to identify the semantic requirements of the application under development. To answer the questionnaires, developers are not required to master semantic technologies, but, at least they should have a minimum knowledge of such technologies.

Next, we provide part of the responses to the questionnaires and the values obtained for some characteristics.

- Will the ontologies be identified by developers at design-time or located by the application at run-time?
    Response: “Mixed (some at design time and some at run-time)”.
    Characteristic: Design-time or run-time ontology selection. Value: “Mixed”
- Will the application aggregate non-semantic data?
    Response: “Yes (transport companies data and corporate database)”.
    Characteristic: Use of non-semantic data. Value: “Yes”
- Will the application deal with contradictory data?
    Response: “Yes (e.g. contradictory weather previsions)”.
    Characteristic: Dealing with contradictory data. Value: “Yes”

Task 3. To Identify System Models. Table 2 shows the resources identified with their associated basic symbols. Table 3 depicts the relationships identified, obtained from the catalogue of system model templates.

As previously in *Task 1*, both use cases are associated to the use case template *Query Information*. Therefore, the development team has chosen the system model template *Query Information with a Hybrid Ontology Approach* (see Figure 5) because of the characteristics previously discovered.

Table 2. Symbols associated to the resources used by the example application

Resource Identifier	**Resource Description**	**Basic Symbol**
Cartography Ontology	Ontology of the cartography provider	Static Ontology
Cartography Instances	Instances of the cartography provider	Static Instances
Transport Schema	XML schema of the transport company provider	Static Non-ontological Resource Schema
Transport Data	XML data of the transport company provider	Static Non-Ontological Resource Content
Weather Ontologies	Ontologies of the weather information providers	Dynamic Ontology
Weather Instances	Instances of the weather information providers	Dynamic Instances
Logistics DB	Corporate database of the logistics company	Non-ontological Resource Content that Conforms to a Given Schema Abbreviation

Table 3. Relationships between the resources used in the example application

Resource 1	Resource 2	Relationship
Cartography Instances	Cartography Ontology	1. Instances that Conform to a Given Ontology
Transport Data	Transport Schema	2. Non-ontological Resource Content that Conforms to a Given Schema
Weather Instances	Weather Ontologies	1. Instances that Conform to a Given Ontology

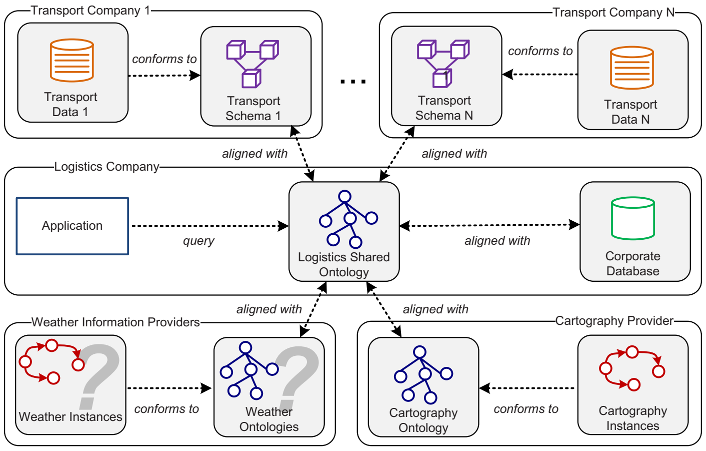


Fig. 9. System model identified for the sample case study

As the template chosen indicates, it is necessary to create and incorporate another ontology, having a shared vocabulary, that will be aligned with the rest of the ontologies and schemas to facilitate information integration. Therefore, several *Aligned With* relationships must be included in the system model.

By integrating the basic symbols and their relationships with the *Query Information with a Hybrid Ontology Approach*, the system model we obtain is the one shown in Figure 9.

6.2 Component Identification Activity

This subsection presents how to carry out the three tasks of the *Component Identification* activity, considering the use cases and system model obtained in the previous subsection.

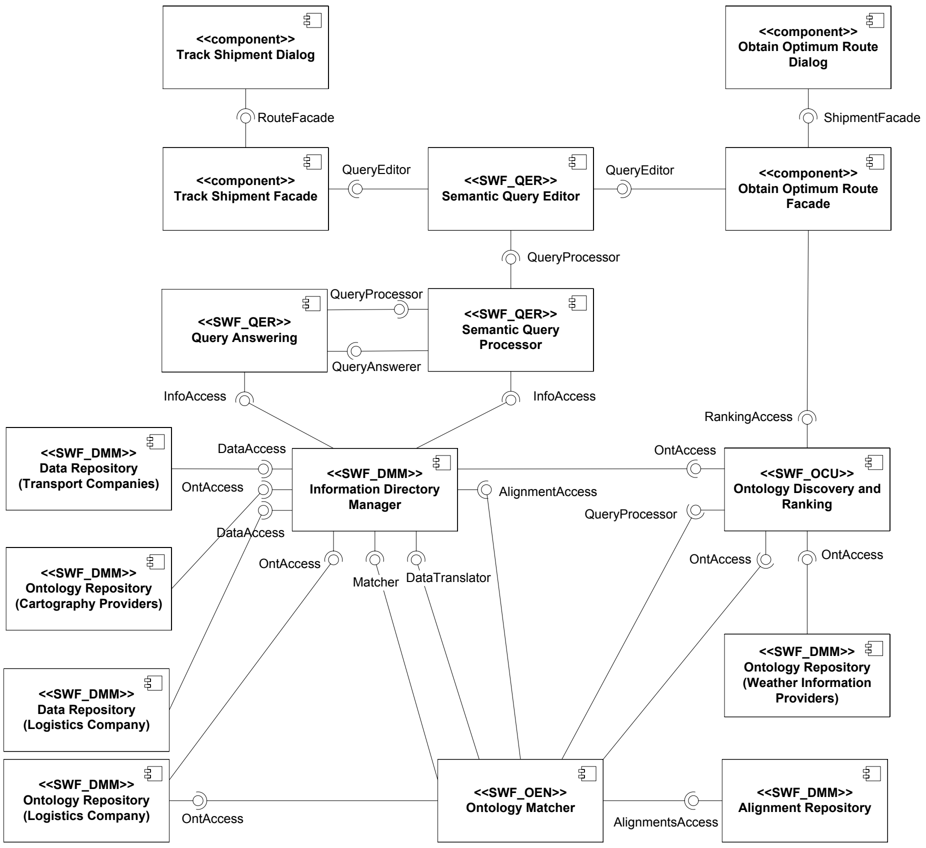
Task 1. To Identify Dialogs and System Facades. Within this task the development team introduces in the architecture a system dialog and a facade for each use case identified as specified in [16]. The *dialog* components implement the logic of each use case, that is, the software that handles the dialog between the actors of a given use case and the system. The *facade* components provide operations for every step specified in the use case flow definition and are used by the *dialog* components.

Table 4. Patterns associated to the repositories used by the sample case study

Resource	System	Pattern
Transport Data	Transport Companies	2. Data Repository
Transport Schema	Transport Companies	2. Data Repository
Logistics Shared Ontology	Logistics Company	1. Ontology Repository
Corporate Database	Logistics Company	2. Data Repository
Weather Ontology	Weather Information Prov.	3. Dynamic Ontological Resource Access
Weather Instances	Weather Information Prov.	3. Dynamic Ontological Resource Access
Cartography Ontology	Cartography Providers	1. Ontology Repository
Cartography Instances	Cartography Providers	1. Ontology Repository

Task 2. To Identify Interfaces to Knowledge Sources. Within this task, the developers catalogue the repositories containing the ontological and non-ontological data that the application will use. For each ontological and non-ontological resource reflected in the system model, its containing repository is identified. Table 4 shows the system and patterns associated to each resource.

Task 3. To Create the Initial Architecture. The architecture shown in Figure 10 is obtained directly by integrating all the components and patterns.



<<component>> Track Shipment Dialog
<<component>> Obtain Optimum Route Dialog
RouteFacade
ShipmentFacade
<<component>> Track Shipment Facade
QueryEditor
<<SWF_QER>> Semantic Query Editor
QueryEditor
<<component>> Obtain Optimum Route Facade
QueryProcessor
<<SWF_QER>> Query Answering
QueryProcessor
QueryAnswerer
<<SWF_QER>> Semantic Query Processor
InfoAccess
InfoAccess
RankingAccess
<<SWF_DMM>> Data Repository (Transport Companies)
DataAccess
OntAccess
DataAccess
<<SWF_DMM>> Information Directory Manager
OntAccess
AlignmentAccess
QueryProcessor
<<SWF_OCU>> Ontology Discovery and Ranking
<<SWF_DMM>> Ontology Repository (Cartography Providers)
OntAccess
Matcher
DataTranslator
OntAccess
OntAccess
<<SWF_DMM>> Ontology Repository (Weather Information Providers)
<<SWF_DMM>> Data Repository (Logistics Company)
<<SWF_DMM>> Ontology Repository (Logistics Company)
OntAccess
<<SWF_OEN>> Ontology Matcher
AlignmentsAccess
<<SWF_DMM>> Alignment Repository

Fig. 10. Architecture identified for the sample case study

7 Evaluation

For evaluating our work, we have analyzed ten use cases, described in the *W3C Semantic Web Case Studies and Use Cases* web page[1]. These test cases are not related to the applications analysed for developing the guidelines described in this paper. For each use case we have applied the activities and techniques proposed in this paper to obtain (1) the characteristics of each application, (2) the use cases that the application covers, (3) the system models, and (4) the architecture of the application.

Figure 11 summarizes the values obtained for some characteristics of the applications analysed; it also shows how many times these values appear in the whole set of applications. With regard to the scenarios that the applications cover, all the use cases templates provided by the guidelines address those scenarios. Since all the use cases described by the guidelines appear in the applications analysed, almost all the system model templates provided by the guidelines can be used to model the structure of some of these applications. However, with respect to the patterns applied to build the architecture, it should be explained that not all the patterns have been used. The reason is that some patterns described in the guidelines are used when ontological and/or non-ontological resources are discovered by the application at run-time, and such dynamic behaviour is not present in the analysed applications.

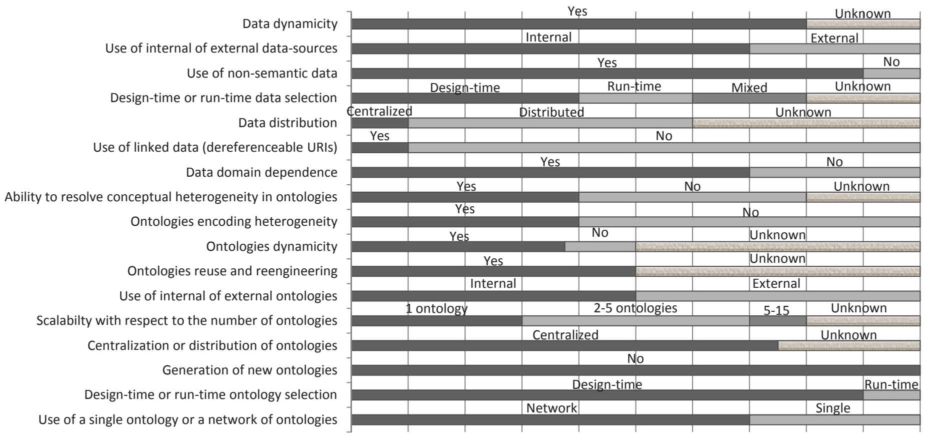


Fig. 11. Values for some characteristics of the W3C Semantic Web Use Cases

Another measure taken during the evaluation has been the time devoted to analysing the requirements and designing each application. The result is that the average time spent in each application is of one day (including the study of the application description in the W3C web page and related papers or technical reports), which is a short period of time.

[1] http://www.w3.org/2001/sw/sweo/public/UseCases/

8 Conclusions

Large-scale semantic applications require different software development methods and techniques from those for classic knowledge-based systems because they manipulate huge quantities of heterogeneous decentralized information, integrate semantic and non-semantic data, and explore different sources at runtime. Therefore, software engineers without expertise should be provided with methodological guidelines for the development of semantic applications.

For this purpose, we have adapted the *Requirements Engineering* and *Design* processes from methodologies widely accepted in Software Engineering. This adaptation allows to design the architecture of semantic applications from scratch and to include semantic components into traditional information systems, by integrating the activities and techniques here described into existing application development processes. The techniques described are novel and especially oriented to the specification and design of the semantic functionalities of an application.

The architectural patterns dealt with are not bound to a particular implementation. Therefore, after using the guidelines here presented, the application architecture will remain independent of concrete component implementations. Architecture realizations in particular settings are out of the scope of this paper.

The catalogues and patterns presented can be extended, and for this purpose a collaborative space (e.g., a wiki) will be enabled to facilitate community feedback, extension and enrichment. The immediate lines of work include to continue defining the rest of the development processes (i.e., *Implementation*, *Integration* and *Testing*). Other future line of work is to specialize the guidelines in order to deal with particular settings, for example, the Open Linked Data initiative.

Another extension should be to give software support to the guidelines by building or adapting an existing CASE tool and by formalizing the processes, activities, methods, catalogues and patterns of the guidelines with ontologies. The purpose here is twofold: to automatically document the large-scale semantic application development process and to support the application code generation. For this last issue, it is necessary to define the rest of the processes and to provide interoperable implementations of the components involved in the semantic application. Finally, the questionnaires will be used to characterize and categorize existing semantic applications and then to carry out an analysis of the current panorama of semantic applications.

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. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In: Ontological Engineering on Spring Symposium Series, Stanford (1997)
3. Staab, S., Schnurr, H.P., Studer, R., Sure, Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems (2001)
4. Muñoz-García, O., García-Castro, R., Gómez-Pérez, A.: Facilitating Requirements Engineering of Semantic Applications. In: Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP 2008), Rome, Italy, CEUR Workshop Proceedings (2008)
5. Motta, E., Sabou, M.: Next Generation Semantic Web Applications. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 24–29. Springer, Heidelberg (2006)
6. Domingue, J., Fensel, D.: Towards a Service Web: Integrating the Semantic Web and Service Orientation. IEEE Intelligent Systems (2008)
7. Krummenacher, R., Simperl, E., Fensel, D.: Scalability in Semantic Computing: Semantic Middleware. In: Proceedings of the IEEE Conference on Semantic Computing, pp. 538–544 (2008)
8. Jasper, R., Uschold, M.: A Framework for Understanding and Classifying Ontology Applications. In: Twelfth Workshop on Knowledge Acquisition Modeling and Management, KAW 1999 (1999)
9. Kozaki, K., Hayashi, Y., Sasajima, M., Tarumi, S., Mizoguchi, R.: Understanding Semantic Web Applications. In: Domingue, J., Anutariya, C. (eds.) ASWC 2008. LNCS, vol. 5367, pp. 524–539. Springer, Heidelberg (2008)
10. 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)
11. García-Castro, R., Gómez-Pérez, A., Muñoz-García, O., Nixon, L.J.: Towards a Component-Based Framework for Developing Semantic Web Applications. In: Domingue, J., Anutariya, C. (eds.) ASWC 2008. LNCS, vol. 5367, pp. 197–211. Springer, Heidelberg (2008)
12. Sommerville, I.: Software Engineering, 8th edn. International Computer Science Series. Addison-Wesley, Reading (2007)
13. Muñoz-García, O., García-Castro, R., Gómez-Pérez, A., Sini, M.: D5.5.1. NeOn Methodology for the development of large-scale semantic applications. Technical report, NeOn Project (2009)
14. Gómez-Pérez, A., Suárez-Figueroa, M.: NeOn Methodology: Scenarios for Building Networks of Ontologies. In: 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW 2008), Conference Poster, Italy (2008)
15. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-based integration of information - a survey of existing approaches. In: IJCAI workshop on Ontologies and Information Sharing, pp. 108–117 (2001)
16. Cheesman, J., Daniels, J.: UML Components. A Simple Process for Specifying Component-Based Software. Component Software Series. Addison-Wesley, Reading (2001)