XML Application for Non-Functional Requirements Management in Systems Project

Carla T. L. L. Silva, Lúcia R. D. Bastos,

Roberto S. M. Barros and Jaelson F. B. Castro

Universidade Federal de Pernambuco - Cin - Recife (PE) – Brasil
Caixa Postal 7851
50732-970 - Recife - PE – Brasil
{ctlls, lrdb, roberto, jbc}@cin.ufpe.br

Abstract. The objective of this paper is to present an environment for the management of functional and non-functional requirements in systems project, developed using the XML (Extensible Markup Language) standard. This environment receives as inputs two entry structures for software requirements modelling: the UML´s Use Case diagram, which describes functional requirements; and the I* framework used by the Tropos methodology, which defines non-functional requirements. Then, it automatically transforms these structures in a third structure that extends the Use Case definitions with the non-functional requirements and provides management facilities of these extensions to UML´s Use Case diagrams.

1 Introduction

Currently, companies are continually changing and turning their attention to improving their business strategies. Accordingly, stakeholders are demanding more flexible and complex systems. To support this complexity, the conceptual models have to describe some aspects beyond entities and activities. Recent research in requirements engineering [8] shows that treating goals, in order to capture associated intentions to complex situations, can be decisive to capture this new reality of software engineering.

Functional requirements describe the behavior (operations, functions or services) of the system that supports user goals, tasks and activities. Non-functional requirements (NFR) include constraints and qualities. *Qualities* are properties or characteristics of the system that its stakeholders care about and hence will affect their degree of satisfaction with the system.

To define the system functional requirements we can underlie scenario-based techniques. Scenarios are used to describe interaction between the users and the software. As an example, we can indicate the *UML Use Cases* [2] that are widely used to capture system functional requirements.

Unfortunately, the *UML Use Cases* and other scenario-based techniques are not suitable to model organizational and non-functional requirements [11]. To represent these aspects, we can use the i* framework [8], which is used by the Tropos methodology [10]. The I* framework provides an adequate representation for alternatives and introduces primitive modeling concepts such as goal and softgoal. A language to support I* framework concepts was implemented and is called GRL (Goal-oriented Requirement Language) [1].

In this context, and because *UML Use Case* diagram is insufficient to define and describe organizational and non-functional requirements, this work is aimed at providing an XML-based environment for automatically integrating the *Softgoals* from i* with the *UML Use Cases*. Finally, the environment will generate and manage extensions of projects and Use Cases definitions. This will allow users (managers) to manage and assess conformance on project decisions related to non-functional requirements, such as performance, availability, security, maintainability and others. A full version of this work can be find in [12].

The rest of this paper is organized as follows: In sections 2 and 3, we will show the basic concepts used in the project: Functional requirements and the semantics of Use Case, non-functional requirements and the GRL's actor, softgoals and task notation and semantics. In section 4, we present the mapping of the UML and GRL concepts to define the UseCase.dtd. In section 5 we show the transformations and extractions implemented in order to automatically generate an XML document containing Project information. In section 6, we present the management tool of non-functional requirements associated to use cases of a system project being developed, and finally, in section 7 we present the final considerations about this work.

2. Functional Requirements and Use Case Semantics

Functional requirements capture the intended behavior of a system. This behavior may be expressed as services, tasks or functions, which the system is required to perform. Use cases have quickly become a standard for capturing functional requirements. As examples of functional requirements we can indicate: checking account, open account or authorize access.

The Use Case is a coherent unit of externally visible functionality provided by a system unit and expressed by sequences of messages exchanged by the system unit and one or more actors of the system unit [3]. The purpose of the use case is to define a piece of coherent behavior without revealing the internal structure of the system.

XMI (XML Metadata Interchange) [6] specifies a structure for exchanging models, and it uses XML (Extensible Markup Language) [4]. The XMI proposal can be considered as a "union" of the object-oriented technology and the data exchange standard in the Internet. The XMI generated from UML is a physical mechanism to exchange UML models in conformity with the UML/OMG metamodel [2]. The uml.dtd was generated directly from the physical UML metamodel, i.e., the representation of the abstract syntax of the UML language [6].

3. Non-Functional Requirements and the Semantics of GRL

Informally, we can think of functional requirements as those capturing *what* the system must do, and of non-functional requirements as those describing *how well* these functional requirements are satisfied. This "how well" is to be externally judged by some observable/measurable properties of the system behavior, not by its internal implementation. As examples of non-functional requirements we can indicate:

- Usability (ease-of-use, ease to learn, efficiency, etc.);
- Requirements of quality of service such as performance (throughput, response time, transit delay, latency, etc.); and
- Safety properties (so-called because they "prevent bad things from happening"), such as security and fault tolerance.

In this work we use the GRL (*Goal-oriented Requirement Language*) [1] to model the non-functional requirements of the system. GRL is a language for supporting the goal-oriented modeling and reasoning of requirements, especially for dealing with non-functional requirements. The GRL language is provided by the *i** framework [8]. This language provides constructs for expressing various types of concepts that appear during the requirement process. GRL supports the reasoning about scenarios by establishing correspondences between intentional GRL elements and non-intentional elements in scenario models of Use Cases. Modeling both goals and scenarios is complementary and may aid in identifying further goals and additional scenarios, contributing to the completeness and accuracy of the requirements.

4. Mapping the XML Document with Uses Cases and SoftGoals

One of the main tasks in the development of an XML application is to write a Document Type Definition (DTD) [4]. A DTD defines modeled data and their relationships.

On this application design, firstly, we created a DTD called UseCase.dtd, which defined the full structure from the association of the Use Case and GRL diagrams. Table 1 shows how the mapping was done.

Table 1. The UseCase.dtd mapping

XMI - UML.DTD	GRL.DTD	UseCase.DTD
<<UseCase>> class	-----	<<UseCase>>
<<UseCase>> Attribute		<<UseCase>> attribute
	<<Softgoal>>	<<NFR>>
	<<Task>>	<<UseCase>>
<<Actor>>		<<Actor>>

5. The Generator of XML Documents complying with *UseCase.dtd*

This section describes the generation of an XML document that complies with the UseCase.dtd, by using XML source documents based on GRL and XMI DTDs. To make such generation we used XSLT (XSL Transformations) [7]. XSLT is a language to transform XML documents into other document types and it is a subset of XSL (Extensible Stylesheet Language).

The XSLT was used to extract information from two XML documents, one containing Use Case diagram information and complying with the XMI dtd, and another containing GRL diagram information and complying with the GRL dtd, in order to create a new XML document combining such information. Fig. 1 shows the XSLT reception and formatting for the source files.

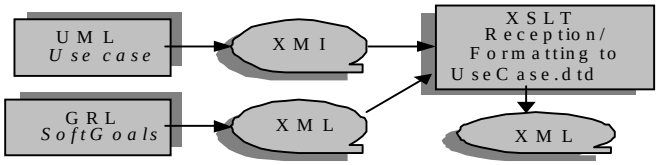

Fig. 1 – The XSLT Reception/formatting diagram

After concluding the XML document storage in relational database, the next step will be to increment the data contained in it, by giving additional information about the use cases, in order to generate a documentation to be used as a guide during the software engineering process.

6. The NFR Management Application

In this section we propose an application project for the management and maintenance of quality structural requirements, i.e., Non-Functional Requirements (NFR), by extending the information of the Use Case diagrams generated by UML [2].

The main functions of this application to manage non-functional requirements are:

- Access Control - where safety functions for NFR's data and users profiles are situated: authorization and responsibility levels associated to DB;
- Interface to enter complement information for Project and Use Case - where users complement the information generated by the diagrams. Users complement it with information about demands, results and conformity information related to NRF requests;
- User's conformity – In this function users set the project development "evaluation" and the demands (request) "acceptance", by verifying if each one is in conformity with user's request, or what is left to be done to become in conformity;

– Several searches to the project and use cases database.

Using the data provided by both the XML document and user's information added later, other documents can be generated for user management, or for data exchange among designers, applications and repositories (Fig. 2 shows the diagram).

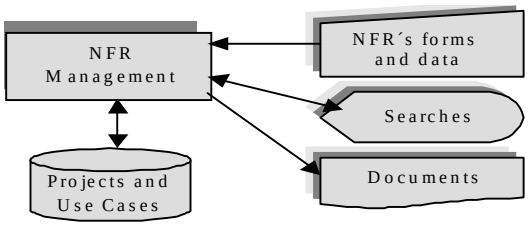

Fig. 2. The NFR Management application can be built being used in several hardware and software platforms, either local or distributed. The database for *Project* and *Use Case*, can be built in any database supporting the ODBC-JDBC bridge

The Fig. 3 shows the data model and the relations among the Project, Use Case and Non-Functional Requirements (NRF) entities. Each project can be accompanied by some users (managers) through registrations of complementary information using the *ComplementProject* entity, for Projects instances, or using the *ComplementUseCase* entity, for Use Cases instances.

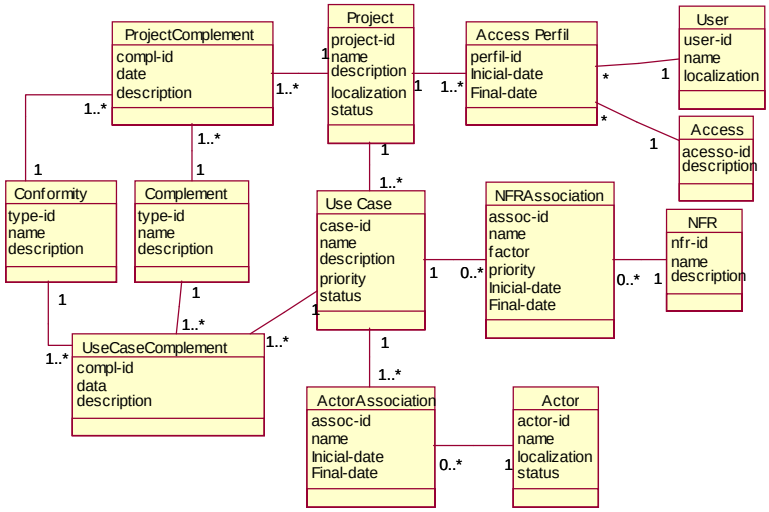

Fig. 3. The data model

7. Conclusions and Future Work

The success key to deploy a quality product is the effective communication in the system development process between users, managers and developers. An effective way of establishing this communication line is using a visual language such as UML diagrams. The Use Case diagrams are widely accepted as a standard to represent business functions between actors that participate of these functions.

In this paper we showed a strategy to manage non-functional requirements and to integrate them with the functional requirements of a system project. We believe that to consider NFRs in the early stages of a system development will make the conceptual model more complete and with higher quality, according to manager and market expectations.

The practical foundation of this work was to build a prototype tool in order to show the viability of integrating organizational models, developed by the I* framework, to scenarios described as Use Cases. Moreover, this integration enables the requirement engineer to choose the best alternative to develop the software and to analyze more deeply those Use Cases that really represent the satisfaction of the organizational goals [9]. In the traditional development process, Use Cases and scenarios in general don't take into account motivations, intentions and alternatives for system development, because they don't address organizational goals. Using organizational models, such as I*, helps in this sense, since it supports goal-oriented modeling, by using concepts such as intentional elements, links and actors.

References

1. GRL Ontology. http://www.cs.toronto.edu/km/GRL/. Access in February 2002.
2. MOF e UML, The Unified Modeling Language, version 1.4, The Object Management Group, May 2001. http://www.omg.org/.
3. Rumbaught, J., Jacobson, I. and Booch, G. The Unified Modeling Language–Reference Manual. Addison Wesley, 1999.
4. XML-Extensible Markup Language. http://www.w3.org/TR/1998/REC-xml-19980210 and http://www.xml.com/. Access in February 2002.
5. Use Case papers. http://www.pols.co.uk/usecasezone/use-case-papers.htm. Access in February 2002.
6. XMI http://www.software.ibm.com/ad/features/xmi.html.
7. XSLT http://www.xml.com/pub/rg/XSLT_Tutorials.
8. i* http://www.cs.toronto.edu/km/istar.
9. Santander, V., Castro, J.: Deriving Use Cases from Organizational Modeling, to appear in: RE`02. IEEE Joint Int. Requirements Engineering Conference. University of Essen, 2002.
10. Castro, J., Kolp, M., Mylopoulos, J.: *A Requirements-Driven Development Methodology* In: CAISE'01, The 13Th Conference on Advanced Information Systems Engineering, 2001, Interlaken, Switzerland.
11. Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J.: "Non-Functional Requirements in Software Engineering". Kluwer Publishing, 2000.
12. Full Paper http://www.cin.ufpe.br/~roberto/XMLApplicationForNFRManagement.pdf.