A Metamodeling Approach for Requirements Specification[1]

Elena Navarro
Department of Computer Science, UCLM
Avda. España S/N, Albacete, Spain
Phone: +34 967 59 92 00 ext. 2461
enavarro@info-ab.uclm.es

Patricio Letelier
Department of Information Systems and Computation, UPV
Camino de Vera s/n, Valencia, Spain
Phone: +34 96 387 7007 ext. 73589
Fax. +34 96 387 73 59
letelier@dsic.upv.es

José Antonio Mocholí
Department of Information Systems and Computation, UPV
Camino de Vera s/n, Valencia, Spain
Phone: +34 96 387 7007 ext. 83525
Fax. +34 96 387 73 59
jmocholi@dsic.upv.es

Isidro Ramos
Department of Information Systems and Computation, UPV
Camino de Vera s/n, Valencia, Spain
Phone: + 34 96 387 7350
Fax. +34 96 387 73 59
iramos@dsic.upv.es

[1] This document has been accepted for publication in the Journal of Computer Information Systems, 47(5): 67-77

A Metamodeling Approach for Requirements Specification

Abstract

There are many Requirements Engineering approaches and techniques that help to specify, analyze and validate requirements in the context of practically any kind of project. However, they are neither widely accepted nor widely used by industrial software community. One of the main problems faced when applying a requirement technique is to what extent it can be easily adapted to the specific needs of the project. This has often led to unsatisfactory requirements management in industrial software development. Currently, Use Case model is the most widely accepted despite its restricted expressiveness and overloaded semantics. Other more sophisticated modeling techniques have been developed independently of any others and/or for specific application domains. Frequently, techniques that provide richer expressiveness do not include scalability from other more popular techniques nor do they offer integration with other more advanced techniques. In this work, we present an approach for requirements modeling that allows the integration of the expressiveness of some of the more relevant techniques in the Requirements Engineering arena. Our work takes advantage of metamodeling as a medium for integrating and customizing Requirements Engineering techniques. By focusing on the scalability with respect to expressiveness and adaptability to the application domain, we have established some basic guidelines and extension mechanisms that lend coherence and semantic precision to our approach. A case study is presented to describe the application of our approach in a real-life project and the tool support we are using.

Keywords: Software Requirements Engineering, Goal-Oriented, Aspect-Oriented, Variability, Use Case, UML, Metamodeling.

1. INTRODUCTION

The drawbacks in Requirements Engineering (RE) and their negative impact on the success of software development projects have led to a great deal of research in this field. However, these supposed advances are not widely applied in industrial software developments. Some critical obstacles for applying RE techniques are: (a) The increasing adoption and integration of different requirements specification approaches to development projects; (b) The prevalence of adaptation over adoption of method; (c) The increasing importance of the definition of domain specific languages (and approaches) in a development environment which tends towards being model driven .

A point has been reached where it is absolutely necessary to integrate knowledge and techniques that have been already developed and facilitate their use in real-life projects. This integration must consider scalability mechanisms for use at different levels of sophistication as well as providing adaptability in order to smooth their application in specific domains.

The aim of this work is to elaborate a proposal for integrating different RE techniques in a common framework and provide some guidelines for adaptation to the specific needs of the project. Our approach is based on metamodeling, as a way of providing a smooth integration and scalability of RE concepts. To establish the expressiveness needed for requirements specification, we have studied five approaches that we consider to be highly representative in the current state of the art in Requirements Engineering: traditional (based on the IEEE 830-1998 [20]), Use Cases [8], Goal Oriented [28], Aspect Oriented [42] and variability management [17]. We have not explicitly included all the details found in each requirement technique. We have started from a limited set of concepts, so that it will be simple and easy to reach a consensus. Additionally, we provide some guidelines to extend this set of concepts, in such a way that a proper semantic coherence can be maintained.

Our approach took shape in the context of a medium-size real-life project where we faced the problem of integrating several RE techniques to provide support for a complex requirements specification, including critical non-functional requirements (safety, performance, interoperability, etc.) and requirements for a product line with sophisticated variability concerns. The characteristics of this project were propitious enough to apply an Action Research method [2] allowing us to define and improve our approach iteratively through continuous discussion with the analysts. At a later stage, we developed a more completed specification and tool support for our proposal, which we are currently applying to other projects.

This work is organized as follows. The following section establishes a global view of the required expressiveness associated with requirements specifications. This is achieved by describing the essential concepts included in five relevant RE techniques. Section 3 presents our approach for requirements specification based on metamodeling, which provides integration facilities to include requirements concepts of different RE techniques. Section 4 describes the application of our approach in a case study along with the model support tool (MetaEdit+ [32]) we use for modeling. Finally, related work in section 5 and conclusions in section 6 conclude this paper .

2. APPROACHES TO REQUIREMENTS ENGINEERING

Before discussing about the different approaches to RE is necessary to introduce the role of RE in software development. Zave [51] provides one of the most well known definitions of RE: “Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families.” This definition points out how important the relation between the expectations of the software and the real system to be

developed is. For this reason, this discipline has been recognized as one of the cornerstone for the success of a project [37].

Several challenges have been identified related to research in RE that need to be dealt with in the years ahead. Both Nuseibeh and Easterbrook [37] have pointed out two which are closely related to the intentions of this work. First of all, they describe the need for richer models for capturing and analysing non-functional requirements. This means that a Requirements Model will have to cope with non-functional requirements as an assurance of usefulness. Secondly,, the reuse of requirements models for specific application domains will greatly lighten the load involved in developing them from scratch. It will therefore be a real advantage to provide specific tools and techniques to customize these reused requirement models according to specific needs.

In view of the above, we have selected five approaches to Requirements Engineering focusing mainly on the integration of their expressiveness. They have been selected as being highly representative in the state of the art in this field and are also the subject of a great deal of research.

2.1. Standard IEEE 830-1998 Approach

Standard recommendations are a reference point with regard to the content expected in a Software Requirements Specification (SRS). The IEEE 830-1998 standard [20] proposes a template which identifies several requirements artefacts, the main ones being described in a section called “Specific Requirements”. These artefacts include: artefact“External interfaces” for user, hardware, software and/or communications, “Functions” of the software, “Constraints” to software; “Design Constraints”, “Logical Database requirements” for any managed information, “Standards Compliance”, and “Software System Attributes” in terms of reliability, availability, security, maintainability and portability.

This standard does not propose details with regard to the attributes that each of these artefacts should possess. The standard does include some recommendations to deal with potential organizations of specific software requirements that are based on "operation modes of the system", "user classes", "objects", "characteristics", "stimuli", "responses" or "functional hierarchies". However, in practice, when the number of requirements and/or the level of complexity (due to the high number of relationships between them) are significant, those recommendations are not enough. Finally, IEEE 830-1998 does not provide any assistance to the process of elaboration or analysis of the requirements.

2.2. Use Case Approach

Use Case modeling [8] has been widely embraced by the industrial community because its straightforward notation and application allows stakeholders to easily understand the requirements specification. This facilitates the elicitation and validation of the requirements. model In a Use Case diagram we mainly distinguish the following artefacts:

- **Use Cases** represent an atomic functionality (there is no hierarchical refinement when Uses Cases are specified or identified) which the system offers to the environment for achieving some specific goal. Basically, templates for specifying Use Cases usually include other artefacts such as: preconditions, post-conditions, communication steps between the environment and the system to obtain a desired service, alternative steps or exceptions, non-functional requirements, etc.
- **Actors** represent the environment of the system and can be users, devices or any other system that interacts with the system under development. The name of the *Actor* describes the role that is played in that interaction.
- **Relationships** which include **Communication** (interaction between the Actor and the Use Case); **Generalization** (applicable both to Uses Cases and Actors to establish specialization hierarchies); **Include** and **Extend** (to factorize an original Use Case).

By means of the Include and Extend unidirectional dependencies, the Use Case model offers expressivity for specifying relationships between requirements. Include and Extend are both factorizations of Use Case specifications but with a different purpose. The Includes relationship permits a Use Case to be reused in other Use Case specifications and the Extends relationship simplifies the specification of a complex Use Case. Additionally, as stated above, Uses Cases do not allow hierarchical refinement, thus it is important to decide the proper granularity level of functionality that a Use Case should have and to exploit correctly Extend and Include relationships. Consequently, the simple and easy notation of Use Cases is actually one of their major inconveniences. In situations where modeling has to be rigorous and/or precise, Uses Cases usually exhibit problems with regard to their interpretation because of their overloaded semantics and lack of consensus.

2.3. Goal Oriented Approach

One of the main activities in the RE process is related to requirements analysis (to determine conflicts, dependencies and alternatives to satisfy the construction of software that meets clients' needs). This aim has inspirited Goal-Oriented proposals such as KAOS [9] or the NFR Framework [6]. Although there are a wide number of proposals (see [26] for an exhaustive survey), some concepts are common to all of them:

- **Goal** describes why a system is being developed. In order to identify it, both functional goals (expected services of the system) and non-functional goals related to the quality of services (constraints on the design, quality attributes, etc.) should be determined.
- **Agent** is any active component, either from the system itself or from the environment, whose cooperation is needed to define the operationalization of a goal.
- **Refinement Relationships**: AND/OR/XOR relationships allow the construction of the goal model as an acyclic directed graph. These relationships are applied from generic goals

towards subgoals until they have enough granularity to be assigned to a specific operationalization.

This approach is especially appropriate for analyzing the specification for two reasons . First, its ability to specify and manage positive and negative interactions among goals [6] allows the analyst to reason about different design alternatives and to validate the Goal Model by means of its animation [48]. Secondly, its capabilities regarding traceability from low-level details to high-level goals (**concerns**) [9] makes it especially suitable to bridge the gap between architectural and requirements models. In spite of these advantages, Goal-Oriented approach has not been widely embraced by the industrial community. One reason is the lack of available tools that make its exploitation difficult . In addition, hitherto not much attention has been paid to the so-called "-ilities" in the requirements specification [37], and this is just one of the key factors of the Goal Oriented Approach.

2.4. Aspect Oriented Approach

This approach has emerged from the implementation level [27], [13] as a mechanism to manage code complexity . With this aim, a set of techniques has been provided that allows the analyst to specify each concern of the system separately. Afterwards, these concerns are weaved according to the expected behaviour of the system. The concerns of the system can refer to either its functionality or any other issue such as security or distribution. In this way, the code that would arise tangled in many locations can be managed by modularizing it as an aspect.

There are several recent proposals that promote the detection and description of aspects at early stages of development (such as the design [47] and requirements phases [42]) in order to satisfy the closure property [13]. This is how the term Aspect Oriented Requirements Engineering (AORE) emerged. AORE does not have its own notation or expressiveness and

current proposals have developed their own notations (see [1] for an extensive survey). Nonetheless, we can enumerate a set of concepts common to all of them:

- **Concern** refers to the interests of the system, which can be either functional or non-functional.
- **Crosscutting** is a relationship among concerns that arises whenever a given concern interacts with other concerns (either by constraining, extending, etc.). A more detailed description of potential crosscutting relationships can be found in [41].

The main purpose of AORE is to improve the crosscutting management and to establish the composition relationships between specifications. This is done by encouraging the **separation of concerns** in a similar way to the role played by **weaving** in Aspect-Oriented Programming. One of the main difficulties whenever this approach is applied is related to the lack of consensus about what an early aspect is and, in particular , how to detect such aspects. Several proposals have appeared in this sense [3][34], but there is no consensus on this question.

2.5. Variability Management Approach

This approach helps the analyst to delay the decision of what functionality or quality aspects will be incorporated in the final system for as long as possible. This approach has been successfully applied in two areas: Software Product Lines (SPLs) [7] and Dynamic Software Architectures (DSA) [40]. Traditionally, most of these proposals have dealt with the management of variability at the architectural level by establishing a central architecture, along with a set of components that can be evolved or integrated according to the system needs. When variability identification is delayed at the architectural level a problem related to product lines and dynamic architectures arises because the number of potential systems and the capability of adaptation, respectively, are more limited [17]. Thus, the early identification

of the variability which is performed in the requirements phase (**early variability**) is a great advantage.

As in AORE, there is no widely accepted notation to specify the variability in the requirements phase. Instead of describing the different notations available, we present below the necessary concepts as designated by Maßen and Lichter [33]:

- **Representation of common and variable parts**. The notation should allow the expression of those assets that are shared among different products of a product line, or among different instances of software architecture and those that are specific to a specific product or instance. The notation should be able to represent both variation points and variants. A **variation point** indicates a specific variability in the specification. A **variant** is the specific realization of variability in a specific variation point. Each variation point has a **time link**, i.e., when the variability is removed during the development process: design, analysis, running, etc. It is also important to define the **multiplicity** (both maximum and minimum) in each variation. The multiplicity determines how many variants must exist at the same time in a product or architecture when the variability is removed.
- **Distinction between types of variability**. The notation must allow the different types of variability to be expressed : a) **option -** when some specific variants can be selected in the instantiation process, b) **alternative -** when only a single variant can be selected and c) **optional alternative -** when either zero or one alternative can be selected from those available.
- **Representation of dependencies between variable parts**. Variants frequently have dependencies among one another that have to be represented. Two examples are dependencies **require** (when a variant must be selected if another variant is present) and **exclude** (when the selection of a variant implies another variant can not be selected).. A extended set of relationships in the requirements stage is presented in [5].

3. A META- MODELING PROPOSAL TO REQUIREMENT SPECIFICATION

Taking into account the approaches described in section 2 and their needs of expressiveness the current proposal has been defined. The first obstacle to overcome when integrating these approaches and their notations is the wide diversity of terms and concepts, with many overlaps among them. We have realized that a consensus would have been be very difficult if we had attempted to define a global notation so as to cope with the entire expressiveness included. Therefore, we have decided to organize our work in two parts. The first part, described in section 3.1, defines a metamodel for the essential concepts that allow us to deal with generic expressiveness. By doing so, we could achieve a consensus more easily. The second part (see section 3.2) describes a process which specifies how this core metamodel can be tailored according to the specific needs of expressiveness.

3.1. A Metamodel for Requirement Specification

The core concepts and their relationships, which are taken from the described approaches, are shown in Figure 1. The essential concept in a requirements specification is *artefact*; which represents any kind of specification at a certain level of granularity. An *artefact* can be a complete SRS document, a section in a document, a piece of text representing a requirement, etc. In addition to the artefacts, it is also necessary to establish relationships among artefacts

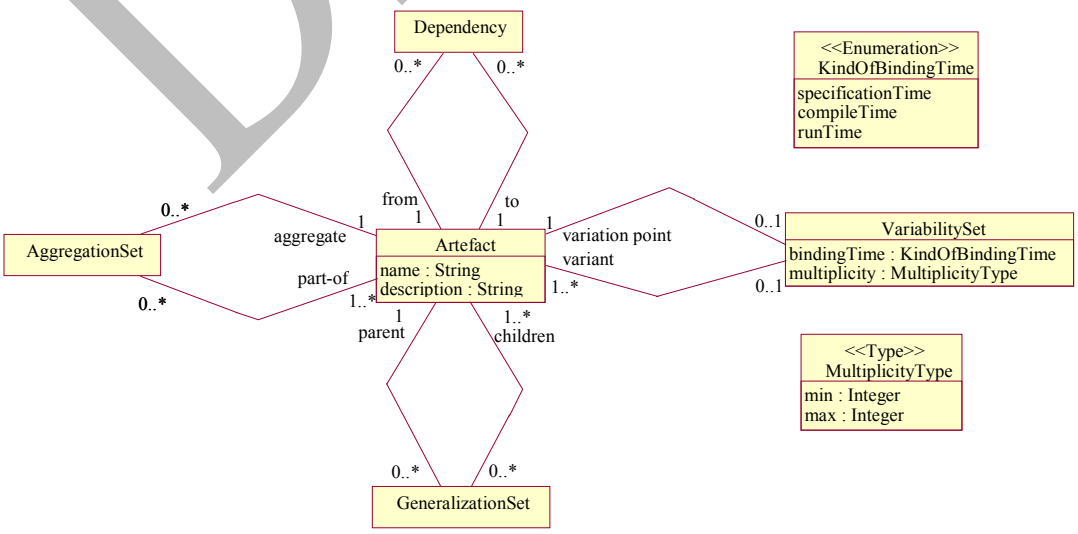

Figure 1. Metamodel for the Core Concepts.

of the SRS. Therefore, we have identified several kinds of relationships which are described below.

An artefact can be refined through other artefacts, forming a hierarchical structure. Three basic kinds of refinements are included: *VariabilitySet*, *GeneralizationSet* and *AggregationSet*. The *VariabilitySet* refinement specifies an artefact as a variation point and establishes its variants. This means that a starting point for a variability hierarchy can be determined from an artefact. During the specification of *VariabilitySet*, the attribute *multiplicity,* which constrains the number of variants to be simultaneously included in the product, must be specified. The attribute *bindingTime* must also be specified in order to determine the maximum delay allowed to decide which variants will be included in the product or in the instanced architecture. Typically, *bindingTime* is assigned as: design time, compile time and runtime. For this reason, they have been included as the enumeration *KindOfBindingTime* and *bindingTime* has been typed with it.

When establishing refinements through *GeneralizationSet* or *AggregationSet*, the same artefact can belong to several hierarchies, either as parent/aggregated or as child/component, respectively. If *GeneralizationSet* is used, it allows the analyst to define hierarchies of alternative specializations from the same parent and to relate one child to more than one parent by multiple inheritance. If *AggregationSet* is employed, it means that an artefact can be part of several aggregate artefacts. Although, these refinements criteria are mutually exclusive, some special cases may arise. For instance, when an established *VariabilitySet* relationship is not only based on a refinement but also on a specialization criterion, this refinement would be considered as a *GeneralizationSet* and a *VariabilitySet* simultaneously.

In addition to these three refinement relationships between artefacts, the dependency relationship has also been included in the metamodel. Perhaps this is the most conflictive relationship for consensus. For this reason, we have preferred to represent it in its most

generic form, i.e., by means of *Dependency* which is applicable to artefacts in the core concepts. Several kinds of dependency relationships are allowed between artefacts of different hierarchies. For example, non-functional requirements constrain functional requirements or goals, data are used for requirements, actors interact with use cases, variants require other variants, etc. Some kinds of dependency between artefacts are bidirectional, such as those used to specify conflicting goals or mutual exclusion between variants. However, others are unidirectional, such as *Extension* dependencies between Use Cases or *Requires* dependencies between variants. Therefore, although in general terms we consider that dependencies are unidirectional, the metamodel also permits the specification of dependencies that in some cases may imply their inverse.

The following table summarizes concepts introduced in section 2 and how the mapping between them and those described in the proposed Metamodel (Figure 1) was established. We have to point out that some concepts do not appear in the core Metamodel but are described by extending it according to the process described in section 3.2.

Table 1 Mapping concepts from concepts of Approaches to RE (rows) to our Metamodel (columns)

Approaches to RE		**Proposed Metamodel**				
	Concept	*artefact*	*Dependency*	*AggregationSet*	*VariabilitySet*	*GeneralizationSet*
Use Case	**Use Case**	Can be described extending				
	Generalization					Realized by
	Communication					
	Include		Extending (*Include*)			
	Extend		Extending (*Extend*)			
Goal-Oriented	**Goal**	Extending				
	Agent	Extending (operationalization)				
	Refinement Rel.					
	AND			Realized by		
	OR/XOR				Realized by	
Variability	**Variant**	An artefact with a Variability or Generalization relationship			Establishing this relation on an artefact	Establishing this relation on an artefact
	Variation point				Described by	Described by
	Time link				Using an attribute	

	Multiplicity				Using an attribute	
	Types of Variability				Using multiplicity	
	Require		Extending (*Require*)			
	Exclude		Extending (*Exclude*)			
AORE	**Concern**	Extending				
	Crosscutting		Extending (*Include, Extend*)			

3.2. A process for customizing the core

Once we have described the metamodel, it has to be tailored according to the specific needs of expressiveness. With this purpose in mind, we suggest the following steps to adapt and/or extend the metamodel which need not be applied sequentially but in accordance with the analyst's preferences:

- To establish the refinement relationships of interest. The essential metamodel provides three forms of refinement: *AggregationSet*, *GeneralizationSet* and *VariabilitySet*. If necessary, an additional refinement type could be added as a specialization of any of the existing ones or as a new and independent one.
- To establish the types of dependency relationships between artefacts. This allows the definition of the relevant relationships between artefacts from the metaclass *Dependency* or any other already defined by means of specialization.
- To include the attributes considered relevant to the types of artefacts, types of refinement and types of dependency. It is necessary to bear in mind that, whenever a new kind of artefact, refinement relationship or dependency relationships is defined, all the attributes defined by the parent are also inherited by the artefact so that their semantic can be considered as avoiding inconsistencies.
- To formalize artefacts and relationships. Those constraints to be applicable on artefacts and relationships have to be described by means of OCL [38] (Object Constraint Language). It

has been selected for this activity because it is a well-known and extended proposal for this purpose. In addition, there are a wide set of tools [39] that allow for simple consistency checks and type checking in terms of defined OCL constraints.

As can be observed in Figure 1, there is no direct connection between the described relationships and those artefacts on which they are to be applied but rather artefacts and relationships are specified independently. This alternative provides us with an improved readability and comprehensibility of the Metamodel. However, this means that whenever a new relationship is defined, the analyst has to constrain the set of artefacts on which the relationship can be applied by means of the following OCL step (v).

4. CASE STUDY: TAILORING THE CORE TO ACTUAL NEEDS

In this section, we will illustrate how the established metamodel can be extended and tailored according to some particular needs of expressiveness. With this purpose, we present a requirements model that we have developed and tested in a real system, which has been carried out within the European project Environmental Friendly and cost-effective Technology for Coating Removal (EFTCoR) [11]. The aim of this project is to design a family of robots capable of performing maintenance operations for ship hulls . The system includes operations such as coating removal, cleaning and re-painting of the hull.

This project exhibits several specific needs in terms of requirements specification, such as, the variability inherent in the family of robots to be handled the high incidence of non-functional requirements (reliability, performance, safety, etc) which crosscut functional ones; the need to evaluate alternative designs to meeting system requirements; and, finally, a large specification where an appropriate organization is unavoidable. In this context of complexity, associated with the requirements specification and the emphasis in achieving reuse through a product family specification, we found favorable conditions for the application of an Action Research allowing us to solve a real problem, by means of continuous refining of our

approach throughout the 3 years of project duration. We play the researchers role, while the research aim is to define a suitable RE approach and produce the requirements specification in this project.The analysts of the project are the critical reference group, and the stakeholders the consortium on behalf of the EFTCoR project.

Bearing in mind these needs, a requirements model was defined using ATRIUM [36], a methodology that guides the analyst through an iterative process, from a set of user/system needs to the instantiation of a Software Architecture. This requirement model is a Goal Model [35] which is based on the Dardenee and Lamsweerde's [9] and Chung's et al. [6] proposals. This Goal Model has been extended [34] by integrating the aspect-oriented approach, in order to achieve both the efficient management of crosscutting and the correct organization of the SRS. In its definition, a strategy for the separation of concerns has also been established based on the standard ISO/IEC 9126 and on the IEEE 830-1998 related to the organization of the SRS. We have also incorporated the variability management in order to apply the model to the definition of product lines and dynamic architectures.

4.1. Goal Model of ATRIUM

Figure 2 shows the metamodel of the ATRIUM Goal Model. The artefacts and the identified relationships shown in the figure 2, correspond to the application of steps described in section 3.2. By applying step (i), three kinds of artefacts were identified to support a Goal-Oriented approach (section 2.3): *Goal*, *Requirement* and *Operationalization*. *Goal* and *Operationalizations* have been defined by inheriting from *artefact* A *Requirement* is defined as a specialization of a *Goal*, but a *Requirement* must also be verifiable and assignable to an agent by means of an operationalization. Step (ii) was not applied because no additional refinement relationship was needed.

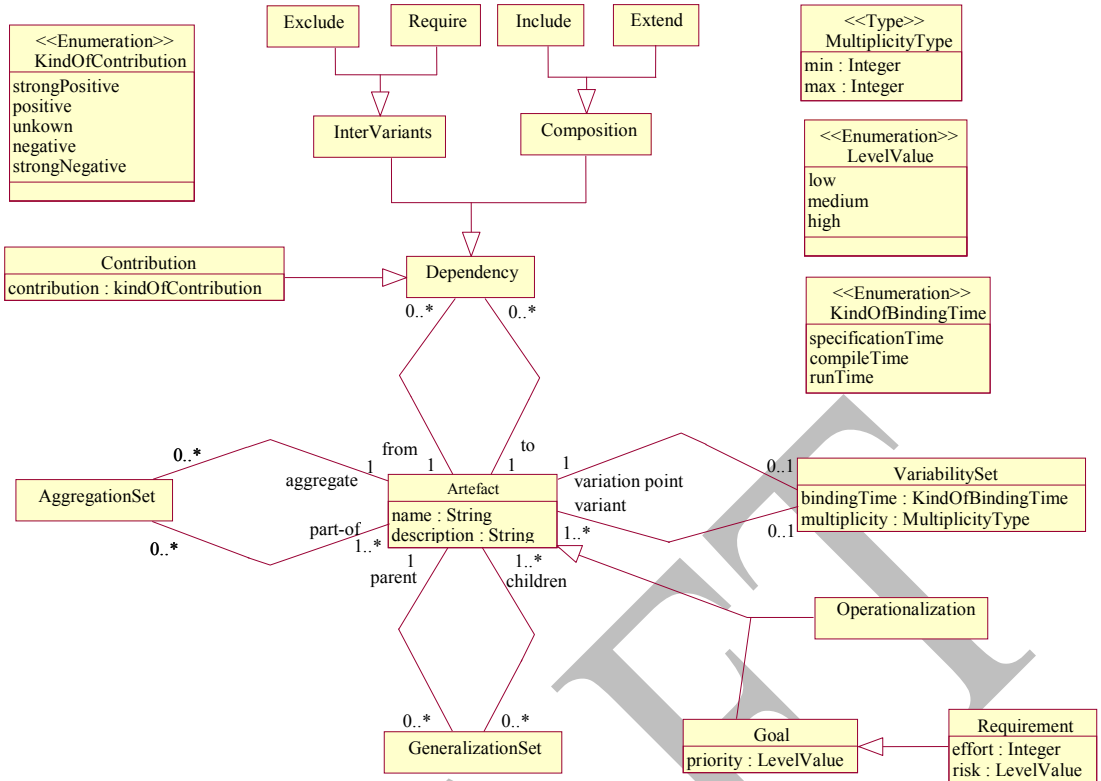

Figure 2. Metamodel of the ATRIUM Goal Model

We have defined a complete and disjoint hierarchy of types of dependencies by applying step (iii). On this hierarchy, *Intervariant*, *Composition* and *Contribution* specializations were identified. *Intervariant* expresses a dependency between variants. This dependency can be either *Require* or *Exclude*, using the semantic described in section 2.5. *Composition* allows the establishment of a composition dependency between artefacts. Following up this idea, the same dependencies as those applied to Use Cases were identified, i.e., *Include* or *Extend*; however, these relationships will model composition dependencies for crosscutting requirements (section 2.4). In addition, we have established a *Contribution* dependency from one artefact to another; it establishes that the accomplishment of one artefact affects negatively or positively the accomplishment of another .

In the application of step (iv), the relevant attributes should be defined. In our case study, we have established the following attributes: *priority* for the metaclass *Goal*; *risk* and *effort* for the metaclass *Requirement*; and *contribution* for the metaclass *Contribution*. Their possible values have been declared at the corresponding enumeration shown in Figure 2.

Due to space limitations, we have not indicated the integrity constraints of the whole model when step (v) is applied; these constraints are applied in addition to those determined by associations and multiplicities defined in the metamodel. In the ATRIUM Goal Model, these constraints refer to the types of artefacts that can be related by means of a specific kind of dependency or refinement. Table 2 shows an example of how we have described which *Contribution* dependencies are allowed between *Operationalizations* and *Requirements* and also which *GeneralizationSet* are possible between *Requirements* or *Goals*.

Table 2 OCL Constraints for ATRIUM Goal Model

Relationship	Constraint
Contribution	**context** Contribution **inv** allowedartefactsContribution: Self.from.oclIsTypeOf(Operationalization)**and** Self.to.oclIsTypeOf (Requirement)
Generalization	**context** GeneralizationSet **inv** allowedartefactsGeneralization: Self.parent.oclIsKindOf(Requirement)**and** Self.children->forAll(a:artefact | a.oclIsKindOf (Requirement))

4.2. MetaEdit+: Tool Support for modeling and Metamodeling

In order to provide tool support for our proposal we have studied three possibilities: (a) to implement a tool specially suited for our approach, (b) to define a UML (Unified modeling Language) profile associated to our metamodel and use a CASE (Computer Aided Software Engineering) tool based on UML for providing visual modeling by means of UML stereotypes, and (c) to use a Meta-CASE tool for defining our metamodel and take advantage of the functionalities offered in the Meta-CASE tool for immediate modeling according to the metamodel. Option (a) is the most flexible one but it requires more effort. We have tested option (b) but we have found some important limitations in achieving full support for UML profiles in a CASE tool. Meta-CASE tools for Domain Specific modeling is a promising emergent technology thrust by current interest into Model Driven Development. Although we are not concerned in generating code from models, we basically need the same facilities

which include: support for metamodel definition, support for modeling according to the metamodel and access to the tool repository in order to generate reports. These reports can be formatted representations of models, new models obtained by means of model transformation, or generated from models following some mappings.

Currently, we are using a Meta-CASE tool namely MetaEdit+ metaCase tool [32]. It is a Domain Specific modeling tool that allows an easy metamodel specification and immediate modeling in accordance with it. It keeps models synchronized with any change performed on their metamodel. With this idea in mind, it integrates two facilities: Method Workbench (Figure 3) which provides the user with a tool suite for designing the metamodel by means of a metamodeling language; and MetaEdit+ which automatically gives the user a full CASE tool functionality according to the metamodel description.

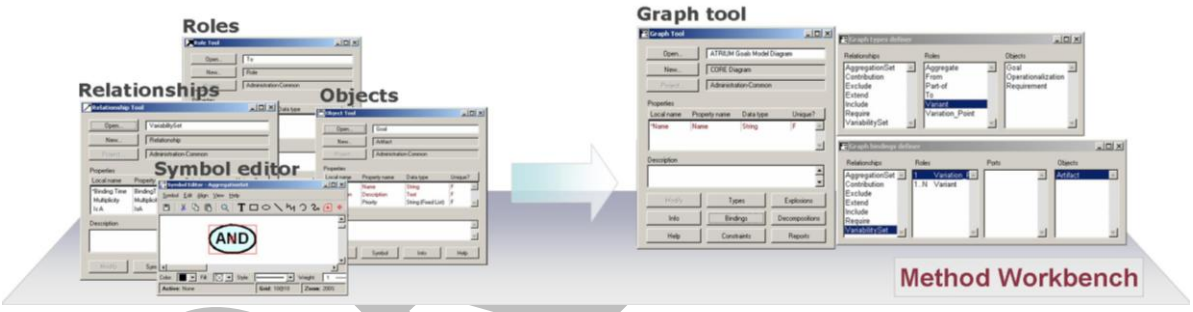

Figure 3 Method Workbench: tool support for metamodeling

Method Workbench is based on the **GOPRR** metamodeling language [32] which defines the following concepts: **Graph**, **Object**, **Property**, **Relationship** and **Role**. Graphs are the graphical representations for a metamodel or part of it. Objects are kinds of entities in the metamodel. Relationships are kinds of n-ary connections among Objects. Relationships have a Role kind for each kind of Object that it connects. Finally, Property allows the specification of attributes for Graphs, Objects and Roles. Therefore, Graphs specifications contain binding information about what Objects can be connected by means of a specific Relationship and by playing a particular Role. The graphical appearance for Objects, Relationships and Roles is

set by means of a Symbol editor, included in the Method Workbench, which allows metamodels to be visually customized.

4.3. Exploiting MetaEdit+ in our case study

We have exploited MetaEdit+ metaCase tool to provide support for our proposal. For this purpose, the first step has been to establish the mapping from the metamodel of the ATRIUM Goal Model to a MetaEdit+ metaCase tool metamodel. We have defined *artefact*, *Goal*, *Requirement* and *Operationalization* as **Objects** with their related properties. Afterwards, *AggregationSet*, *Dependency,* and *VariabilitySet* have been defined as **Relationships**. Finally, we have defined a **Graph** by establishing bindings among such Objects and Relationships.

Once we have stored the whole metamodel in the repository, MetaEdit+ metaCase tool provides us with full Case tool functionality for specifying ATRIUM Goal Models: MetaEdit+ (see Figure 4). In addition, thanks to the report facilities of MetaEdit+ we have defined specific reports for verification and analysis of models. We would like to point out that this facility allows us to generate XMI (XML Metadata Interchange) files for each model. It facilitates its checking by means of any of the existing tools of OCL [39] by and using the OCL constraints described following step (v).

Figure 4 shows an outline of the model for the EFTCoR system. We have started from the quality characteristics of ISO/IEC 9126 standard, using “Portability”, “Adaptability”, “Functionality”, “Suitability”, “Efficiency “and “Time Behaviour”, according to the concerns in our case study. The *AggregationSet* refinements labeled with AND indicate that the satisfaction of an aggregated goal is achieved if every one of its component goals is satisfied. The refinement OR under the goal “Adaptability Working Environment” indicates that it is a variation point, which is defined by means of a refinement *VariabilitySet*. Thus, in our example, either the goal “Adaptability Hull Surfaces” or the goal “Adaptability Working

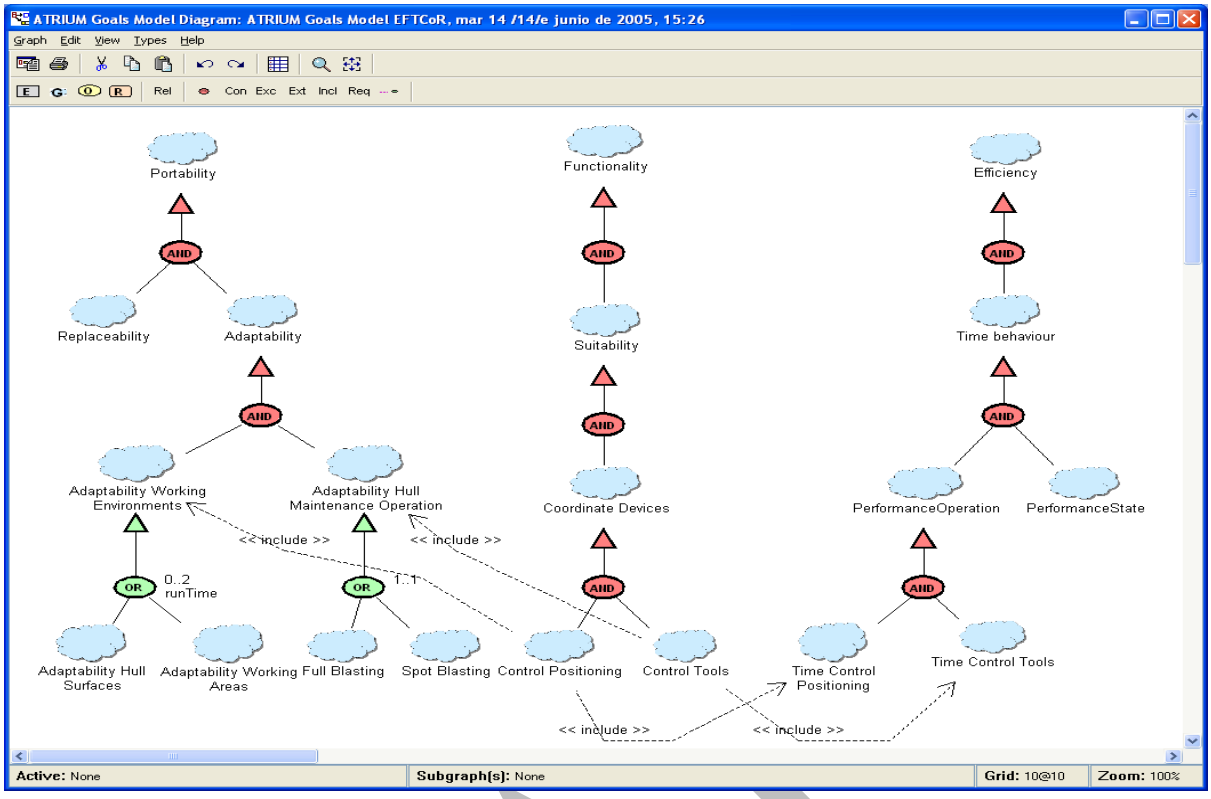

Figure 4 Modelling a part of the Goal Model of the EFTCoR system by means of MetaEdit+.

Areas" had to be satisfied in order to achieve a system that was self-adaptive to different working environments. In this case, both subgoals were satisfied by means of characteristics added at runtime. The 0..2 multiplicity indicates that the goal "Adaptability Working Environments" is optional, it can have zero selected variants. "Adaptability Hull Maintenance-Operation" is also a variation point that should have at least one variant at runtime (minimum multiplicity 1).

The functionality of coordinating devices ("Coordinate Devices") was refined into two subgoals: positioning control ("Control Positioning") and tools control ("Control Tools") on the robot. The specification of these subgoals is composed of other non-functional goals obtained from the refinement of "Adaptability" and "Efficiency". The refinement of the goal "Efficiency" establishes time constraints on the control and tools positioning according to those established by "Time Control Positioning" and "Time Control Tools". This composition of specifications was modeled by means of dependencies that were labelled as *Include*.

4.4. Discussion of results

Several conclusions were achieved after the use of this customized metamodel along with its tool support. First of all, we have to point out its ability to cope with all the specific needs of this project. Previously, Use Cases and IEEE 830-1998 were used for specification purposes. We have to point out that, after an initial elicitation step, more than one hundred requirements were identified. Most of them were non-functional ones with the difficulties inherent to the crosscutting and the large specification. This crosscutting appears in several contexts, mainly whenever non-functional requirements were constraining functional ones, for instance, the performance goal "Time Control Positioning" which restricts the functional goal "Control Positioning".

In this sense, the proposed model was helpful for the analysts in the project because the metamodeling approach provides them with the ability to suitably define and use these kinds of crosscutting as specific dependencies among artefacts. Moreover, this allows them to trace these dependencies to other artefacts throughout the process development in the same framework.

In addition, the availability of a tool with graphical capabilities both for metamodeling and modeling also offered a great advantage over other less flexible alternatives such as UML profiles. One of the key points for this project was to provide a clear notation where every stakeholder could easily understand the specification. We need to bear in mind that a multidisciplinary team is developing this project integrated by computer science masters, industrial masters, telecommunication masters, etc. For this reason, a graphical notation made the specification more readable and comprehensible to the practitioners of the project.

5. RELATED WORKS

In this section, we focus on describing works that include some of the requirements engineering approaches described in section 2. It is important to point out that their purpose is

not the integration of approaches but the extension of an existing notation to achieve some specific expressiveness. In spite of this fact, none of them provides neither guidelines about how its notation can be tailored to specific needs nor any detail about how they extended the initial notation.

Within the scope of the Aspect-Oriented approach, the proposals included in [34] and [49] exploit a Goal Model to identify and specify aspects in early stages. Navarro et al [34] have extended the expressiveness of the Goal Model by incorporating a new type of relationship which allows the specification of the crosscutting between goals. Yu et al [49] identify the crosscutting whenever an operationalization contributes to the satisfaction of several goals.

Brito et al [4] present an extension of the Use Case diagram for its application to the AORE context. They have specified «wrappedBy» and «overlappedBy» relationships by stereotyping a dependency. In this way, if a non-functional requirement implies a constraint on a functional one, this would appear as a Use Case that is stereotyped as a «crosscuttingConcern» and would present a weaving relationship with the other Use Cases. Other proposals, such as [3] and [46], find it more appropriate to employ two different models to specify functional and non-functional requirements, using a Use Case model and a Goal Model, respectively. The main problem with these approaches is that specification can hardly be analyzed, due to the fact that they use two separate models.

In [15], [18] and [33] extensions to Use Case diagrams are used to manage the variability. All of them take advantage of the easy communication with the user. However, there are drawbacks whenever a specific expressiveness is required to specify the variability. Basically, these proposals have increased the expressiveness of the Use Case diagrams by means of the introduction of stereotypes that allow variability concepts to be modeled. Thus, for example, when a Use Cases specifies a functionality that some products of the family have, it is stereotyped as «variant». The proposal presented by Halmans and Pohl [18] is

perhaps the more extensive because it is the only one to incorporate its own graphical notation to specify the variation points and the maximum and minimum multiplicities. However in this case, the formulated extension occasions some drawbacks that are inherent to the Use Case approach, because it does not provide by default any support to analyze or validate the model.

Similarly, the Goal-Oriented Approach has been recently applied to modeling and analyzing variability. Meaningful works in this area are those presented in [16] and [19]. When the Goal-Oriented approach is applied, AND/OR/XOR relationships appear as a consequence of the construction process of the model and are recorded in the refinement graph. Thus, these goals that have been refined by means of OR and XOR relationships are capable of representing variation points in the specification of a product line or a dynamic architecture, in a natural way. However, these proposals have some deficiencies from the point of view of the required expressiveness for variability; for example, they lack the concept of multiplicity or specific dependencies between variants. However, they do provide sophisticated mechanism for analyzing alternatives that have been used to select the optimum variant for each product [16] or for each profile/ability of the user [19].

We have analyzed some current tools for the management of requirements such as DOORS [10], IRqA [21] and RequisitePro [43]. Even though they exhibit a great potential for defining artefacts, they only offer one type of dependency (the traceability specified between artefacts and characterized as *from* or *to*) and only one refinement (using the *AggregationSet* semantic).

6. CONCLUSIONS AND FURTHER WORKS

One of the main challenges for RE is to prove in practice the advantages of the proposed techniques, providing facilities for the integration and adaptation of RE technology to real-life projects. Each project has its specific needs and requires to select, integrate and

customize suitable techniques to define its RE method. In this work, we have presented an approach based on metamodeling to offer such an integration and adaptability.

One important problem when defining highly expressive models (which can handle a wide range of types of artefacts and/or dependencies) is achieving a consensus with regard to their semantic. We have deal with this problem by using metamodeling which allows us to adapt and extend a core set of concepts keeping a suitable level of semantic consistence. In order to establish a global set of RE concepts and the required expressiveness, we have studied five representative techniques for requirements specification. In our approach, according to the project specific needs, it is provided a proper integration as well as scalability from simple up to other more sophisticated RE techniques. The main features of our approach are:

- Definition of a metamodel which includes a core of concepts that corresponds to the essential expressiveness of some of the more popular and/or advanced approaches in requirements engineering.
- Establishment of guidelines for adapting the metamodel to specific needs, according to the expressiveness. We have validated both the core and the presented extension of the metamodel by means of a UML profile and its application to the EFTCoR system.
- This metamodel is the first step towards a tool that supports modeling requirements by integrating expressiveness from different approaches. The demands for scalability and expressiveness, adaptability to specific application domains and the flexible organization of the SRS only can be efficiently satisfied by means of a software tool. The proposed metamodel has been implemented as a UML profile in a CASE tool Rational Rose, obtaining a simple tool support for the metamodel and the associated requirements models. However, due to limitations of UML profiles and incomplete CASE tool support for them, this option does not provide the necessary dynamic configuration of the expressiveness and scalability of the metamodel. Therefore, we have overcome these drawbacks using a Meta-

CASE tool. Using MetaEdit+ metaCASE tool we have validated our approach in the EFTCoR project.

We consider that our proposal constitutes a step forward in achieving a successful application of RE techniques in real-life projects. We have obtained a preliminary validation of our proposal through its application in a medium-size project with satisfactory results.

Two topics constitute our future work. The first of them is related to studying other interesting approaches in Requirements Engineering area in the context of our proposal: *View Points* [14], *Problem Frames* [23] or [29] and *Cognitive Mappings* [45]. Based on the results of our case study, we think it will be easy to include the additional required expressiveness. Our proposal facilitates a deeper analysis of the specification because proper traceabilities could be established, for instances, between conceptual maps described by means of Cognitive Mappings and services of the system-to-be described using goals models.

The second work is focused on providing a formal framework for analyzing and checking models. It is necessary to specify artefacts and to provide a precise semantic for the expressiveness provided by the model. The works by Letier and Lamsweerde's [31] or Katz and Rashid [25] can be useful as a reference to achieve this aim.

7. REFERENCES

[1] AOSD Europe, Survey of Aspect-Oriented Analysis and Design Approaches, http://www.aosd-europe.net/documents/analys.pdf, 2005.

[2] Baskerville, R.L., Wood-Harper, A.T. "A Critical Perspective on Action Research as a Method for Information Systems Research," *Journal of Information Technology*: (11), 1996, pp. 235-246.

[3] Brito, I., Moreira, A. "Integrating the NFR framework in a RE model", Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, collocated to 3rd. Aspect-Oriented Software Development Conference (AOSD), Lancaster, UK: March 22, 2004.

[4] Brito, I., Moreira, A. "Towards a composition process for aspect-oriented requirements", Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design Workshop collocated to 2nd Aspect-Oriented Software Development Conference,, USA: March 17, 2003.

[5] Bühne, S., Halmans, G. and Pohl, K. "Modeling Dependencies between Variation Points in Use Case Diagrams", 9th. International Workshop on Requirements Engineering: Foundation for Software Quality. Collocated to CAiSE'03, Klagenfurt/Velden, Austria: 16 -17 June 2003.

[6] Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J., *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing, Boston Hardbound, 2000, 472.

[7] Clements, P. and Northrop, L. *Software Product Lines - Practices and Patterns*, Addison-Wesley Professional, 2001, pp. 530.

[8] Cockburn, A. *Writing Effective Use Cases,* Addison Wesley Professional, 2000, pp. 304.

[9] Dardenne, A., van Lamsweerde, A. and Fickas, S. "Goal-directed Requirements Acquisition", *Science of Computer Programming*, 20(1-2): 1993, pp. 3-50.

[10] DOORS, http://www.telelogic.com/products/doorsers/doors/, 2005.

[11] EFTCOR: Environmental Friendly and cost-effective Technology for Coating Removal. European Project, 5th Framework Program (GROWTH G3RD-CT-00794), 2003.

[12] Elrad, T., Filman, R. E., Bader, A. "Aspect-oriented programming: Introduction", *Communications of the ACM*, 44(10): 2001, pp. 29 – 32.

[13] Elrad, T., Aksits, M., Kiczales, G., Lieberherr, K. and Ossher, H. "Discussing aspects of AOP", *Communications of the ACM*, 44(10): 2001, pp. 33 – 38.

[14] Finkelstein A. and Sommerville I., "The Viewpoints FAQ", *Software Engineering Journal,* 11:(1), 1996, pp. 2-4.

[15] Gomaa, H. Designing Software Product Lines with UML: From Use Cases to Pattern-based Software Architectures, Addison Wesley Professional, 2004, pp. 736.

[16] Gonzales-Baixauli, B., Prado Leite, J.C.S, Mylopoulos, J. "Visual Variability Analysis with Goals Models", Proceedings of the 12th International Conference on Requirements Engineering, Kyoto, Japan: September 6-10, 2004, pp. 198-207.

[17] van Gurp, J., Bosch, J. and Svahnberg, M. "On the notion of variability in software product lines", Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01), Amsterdam, The Netherlands: August 28 - 31, 2001, pp. 45—54.

[18] Halmans, G., Pohl, K. "Communicating the Variability of a Software Product Family to Customers". *Software and Systems Modeling*, 2(1): 2003, pp. 15-36.

[19] Hui, B., Liaskos, S., Mylopoulos, J. "Requirements Analysis for Customizable Software Goals-Skills-Preferences Framework", Proc. 11th IEEE International Requirements Engineering Conference, Monterey Bay, California, USA: September 08 – 12, 2003, pp. 117-126.

[20] IEEE Std. 830-1998. IEEE Recommended Practice for Software Requirements Specifications, Volume 4: Resource and Technique Standards, IEEE Software Engineering Standards Collection.

[21] IrqA, http://www.irqaonline.com, 2005.

[22] ISO/IEC Standard 9126-1 Software Engineering- Product Quality-Part1: Quality Model, ISO Copyright Office, Geneva, June 2001

[23] Jackson, M. Problem Frames: Analyzing and Structuring Software Development Problems, Addison-Wesley Pub, 2000, pp. 416.

[24] John, I. and Muthig, D. "Tailoring use cases for product line modelling", International Workshop on Requirements Engineering for Product Lines, Essen, Germany: 2002.

[25] Katz, S. and Rashid, A. "From Aspectual Requirements to Proof Obligations for Aspect-Oriented, Systems", Proceedings of 12th IEEE International Requirements Engineering Conference, Kyoto, Japan: September 06 - 10, 2004, pp. 48-57.

[26] Kavakli, E. Loucopoulos, P. "Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods". Information Modeling Methods and Methodologies, IDEA Group, 2005, pp. 102 – 124.

[27] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. Loingtier, J. M. and J. Irwin, "Aspect-Oriented Programming". Proceedings of the European Conference on Object-Oriented Programming, LNCS 1241, Finland Jyväskylä, Finland: June 9-13, 1997, pp. 220--242.

[28] van Lamsweerde, A. "Goal-Oriented Requirements Engineering: A Guided Tour", Invited Paper for 5th IEEE International Symposium on RE, Toronto, Canada: August 2001, pp. 249-263.

[29] Lauesen S., "Task Descriptions as Functional Requirements", *IEEE Software*, 20:(2), 2003, pp. 58-65.

[30] Letelier, P., Navarro, E. and Anaya, V. "Customizing Traceability in a Software Development Process", Information Systems Development Advances in Theory, Practice, and Education, Springer Science+Business Media, Inc., USA, 2005, pp. 137-148.

[31] Letier, E. and van Lamsweerde, A., "Deriving Operational Software Specifications from System Goals", *ACM SIGSOFT Software Engineering Notes*, 27(6): 2002, pp. 119 - 128.

[32] Kelly, S., Lyytinen, K., Rossi, M.: "METAEDIT+ A fully configurable Multi-User and Multi-tool CASE and CAME Environment". Proceedings of 8th International Conference on Advances Information System Engineering, LNCS1080, Springer-Verlag, 1996, pp. 1-21.

[33] von der Maßen, T. and Lichter, H. "Modeling Variability by UML Use Case Diagrams", International Workshop on Requirements Engineering for Product Lines (REPL'02), collocated to the International Conference on Requirements Engineering, Essen, Germany: September 9, 2002.

[34] Navarro, E., Letelier, P. and Ramos, I., "Goals and Quality Characteristics: Separating Concerns", Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, collocated to OOPSLA 2004, Vancouver, Canada: October 25, 2004.

[35] Navarro, E., Ramos, I. and Pérez, J. "Goals Model-Driving Software Architecture", Proceedings of the 2nd International Conference on Software Engineering Research, Management and Applications, Los Angeles, USA: May 5-8, 2004, pp. 205-212.

[36] Navarro, E., Ramos, I. and Pérez, J., "Software Requirements for Architectured Systems", Proceedings of 11th IEEE International Requirements Engineering Conference (RE'03), Monterey, USA: September 8-12, 2003, pp. 365-366.

[37] Nuseibeh, B., Easterbrook. S., "Requirements Engineering: A Roadmap", Proceedings of 22nd International Conference on Software Engineering, Future of Software Engineering Track, Limerick Ireland: June 4-11, 2000, pp. 37-46.

[38] OCL 2.0 Specification, Version 2.0, OMG, http://www.omg.org/docs/ptc/05-06-06.pdf, 2005.

[39] OCL tools & services http://www.klasse.nl/ocl/ocl-services.html, 2005.

[40] Oreizy, P., Medvidovic, N., Taylor, R. N. "Architecture-Based Runtime Software Evolution", Proceedings of the 20th International Conference on Software Engineering (ICSE-98), Kyoto, Japan: April 1998, pp. 117-186.

[41] Rashid, A., Moreira, A., Araújo, J. "Modularization and composition of aspectual requirements", Proceedings of the 2nd international conference on Aspect-oriented Software Development, Boston, USA: March 17 - 21, 2003, pp. 11-20.

[42] Rashid, A., Sawer, P., Moreira, A. and Araújo, J. "Early Aspects: a Model for Aspect-Oriented Requirements Engineering", Proceedings of the International Conference on Requirements Engineering, Essen, Germany: September 9-13, 2002, pp. 199-202.

[43] RequisitePro, http://www.306.ibm.com/software/awdtools/reqpro/, 2005.

[44] Rational Rose, http://www-306.ibm.com/software/awdtools/developer/datamodeler/, 2005.

[45] Siau, K., Tan, X., "Technical Communication in Information Systems Development: The Use of Cognitive Mapping" *IEEE Transactions on Professional Communications*: 48(3): 2005, pp. 269-284.

[46] Sousa, G., Soares, S., Borba, P., J. Castro, "Separation of Crosscutting Concerns from Requirements to Design: Adapting the Use Case Driven Approach", Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, collocated to 3rd. Aspect-Oriented Software Development Conference (AOSD), Lancaster, UK: March 22, 2004.

[47] Suzuki, J. and Yamamoto, Y, "Extending UML with Aspects: Aspect Support in the Design Phase", Aspect Oriented Programming Workshop, collocated to the 13th European Conference on Object Oriented Programming (ECOOP'99) Lisbon, Portugal: June 1999.

[48] Tran Van, H., van Lamsweerde, A., Massonet, P. and Ponsard, C. "Goal-Oriented Requirements Animation", Proceedings of 12th IEEE International Requirements Engineering Conference, Kyoto, Japan: September 06 - 10, 2004, pp. 218-228.

[49] Yijun Y.; Leite, J.C.S.P.; Mylopoulos, J.; "From Goals to Aspects: Discovering Aspects from Requirements Goal Models", Proceedings of 12th IEEE International Requirements Engineering Conference (RE'03), Kyoto, Japan: September 06 - 10, 2004, pp. 38-47.

[50] Yu, E., Liu, L., & Li, Y. "Modelling Strategic Actor Relationships to Support Intellectual Property Management", Proceedings of the 20th International Conference on Conceptual Modelling, ER-2001, LNCS Vol. 2224, Yokohama, Japan: November 27-30, 2001, pp. 164 - 178.

[51] Zave, P. "Classification of Research Efforts in Requirements Engineering". *ACM Computing Surveys*: 29(4), 1997, pp. 315-321.