Journal of Computer Information Systems

ISSN: 0887-4417 (Print) 2380-2057 (Online) Journal homepage: http://www.tandfonline.com/loi/ucis20

# A Metamodeling Approach for Requirements Specification

Elena Navarro, Patricio Letelier, Jose Antonio Mocholi & Isidro Ramos

# A METAMODELING APPROACH FOR REQUIREMENTS SPECIFICATION

**ELENA NAVARRO**
**UCLM**
**Albacete, Spain**

**JOSE ANTONIO MOCHOLI**
**UPV**
**Valencia, Spain**

**PATRICIO LETELIER**
**UPV**
**Valencia, Spain**

**ISIDRO RAMOS**
**UPV**
**Valencia Spain**



## ABSTRACT

There are many Requirements Engineering approaches and techniques that help to specify, analyze and validate requirements. However, they are neither widely accepted nor widely used by the industrial software community. One of the main problems faced when applying them is to what extent they can be easily adapted to the specific needs of the project. This has often led to unsatisfactory requirements management in industrial software development. In this work, a proposal for requirements modeling is presented that allows the integration of the expressiveness of some of the more relevant Requirements Engineering techniques by taking advantage of metamodeling. This proposal focuses on scalability with respect to expressiveness and adaptability to the application domain to establish some basic guidelines and extension mechanisms that lend coherence and semantic precision. A case study and 4 tool support are presented to describe the application of this proposal in a real-life project.

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

## 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; and, (c) the increasing importance of the definition of domain specific languages (and approaches) in a development environment which tends toward being model driven.

A point has been reached where it is absolutely necessary to integrate knowledge and techniques that have already been 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 make their application in specific domains smoother.

This work aims at elaborating a proposal for integrating different RE techniques in a common framework and providing 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. In this sense, we provide some guidelines to extend a core set of concepts, in such a way that a proper semantic coherence can be maintained. 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 (18)), Use Cases (7), Goal-Oriented (27), Aspect-Oriented (39) and variability management (15). 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.

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. These characteristics were propitious enough to apply an Action-Research method, allowing us to define and improve our approach iteratively through continuous discussion with the analysts.

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 proposal in a case study along with the support tool. Finally, related work in section 5 and conclusions in section 6 conclude this paper.

## APPROACHES TO REQUIREMENTS ENGINEERING

RE has been recognized as one of the cornerstones for the success of a project (34). Zave (49) provides one of the best 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 behavior, and to their evolution over time and across software families." This definition emphasizes

how important the relation between the expectations of the software and the real system-to-be developed is.

Several challenges have been identified related to research in RE that need to be dealt with in the years ahead. Nuseibeh and Easterbrook (34) 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 requirements models according to specific needs.

Consequently, we have selected five approaches to RE focusing mainly on the integration of their expressiveness. They have been selected as being highly representative of the state-of-the-art in this field and, moreover, are the subject of a great deal of research too.

### 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 (18) proposes a template which identifies several requirements artifacts, the main ones being described in a section called "Specific Requirements." These artifacts include: "External interfaces" for user, hardware, software and/or communications, "Functions" of the software, "Constraints" to software; "Design Constraints," etc.

This standard does not propose details with regard to the attributes that each of these artifacts should possess. However, it 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." Nevertheless, 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.

### Use Case Approach

Use Case modeling (7) 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. In a Use Case diagram, we mainly distinguish the following artifacts:

**Use Cases** (UC) represent an atomic functionality which the system offers to the environment for achieving some specific goal. Basically, templates for specifying UC usually include other artifacts such as: preconditions, post-conditions, communication steps between the environment and the system to obtain a desired service, 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 Use Cases and Actors to establish specialization hierarchies); **Include** and **Extend** (to factorize an original UC).

By means of the Include and Extend unidirectional dependencies, the relationships between requirements are specified. Both of them are factorizations of UC specifications but with a different purpose. The Includes relationship permits a UC to be reused in other UC specifications; and the Extends relationship simplifies the specification of a complex UC. Additionally, UCs do not allow hierarchical refinement, making it important to decide the proper granularity level of functionality that a UC should have and to correctly exploit Extends and Includes relationships (see (45) for more details about analysis of UCs). Consequently, the simple and easy notation of UCs is actually one of their major inconveniences, in situations where modeling has to be rigorous and/or precise.

### 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). This aim has inspired Goal-Oriented proposals such as KAOS (8) or the NFR Framework (5). Although there are a wide number of proposals (25), 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 toward subgoals until they have enough granularity to be operationalized.

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 (5) allows the analyst to weigh up different design alternatives. Secondly, its capabilities regarding traceability from low-level details to high-level goals (**concerns**) (8) 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, mainly because hitherto not much attention has been paid to the so-called "-ilities" in the requirements specification (34).

### Aspect-Oriented Approach

This approach has emerged from the implementation level (26) as a mechanism to manage code complexity. With this aim in mind, 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 behavior 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 (46) and requirements phases

(39)) in order to satisfy the closure property (12). 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. 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 (38).

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.

### 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) (6) and Dynamic Software Architectures (DSA) (37). 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 PL and DSA arises because the number of potential systems and the capability of adaptation, respectively, are more limited (15). 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 (32):

**Representation of common and variable parts**. The notation should allow the expression of those assets that are shared among different products of a PL, or among different instances of DSA, 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 resolved during the development process: design, analysis, running, etc. It is also important to define the **multiplicity** (both maximum and minimum) in each variation, which 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 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 (4). 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 that another variant cannot be selected).

## A META-MODELING PROPOSAL FOR REQUIREMENTS SPECIFICATION

The current proposal has been defined taking into account the approaches described in section 2 and their need for expressiveness. 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 realized that a consensus would have been 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.

### 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 piece of text representing a requirement, etc.

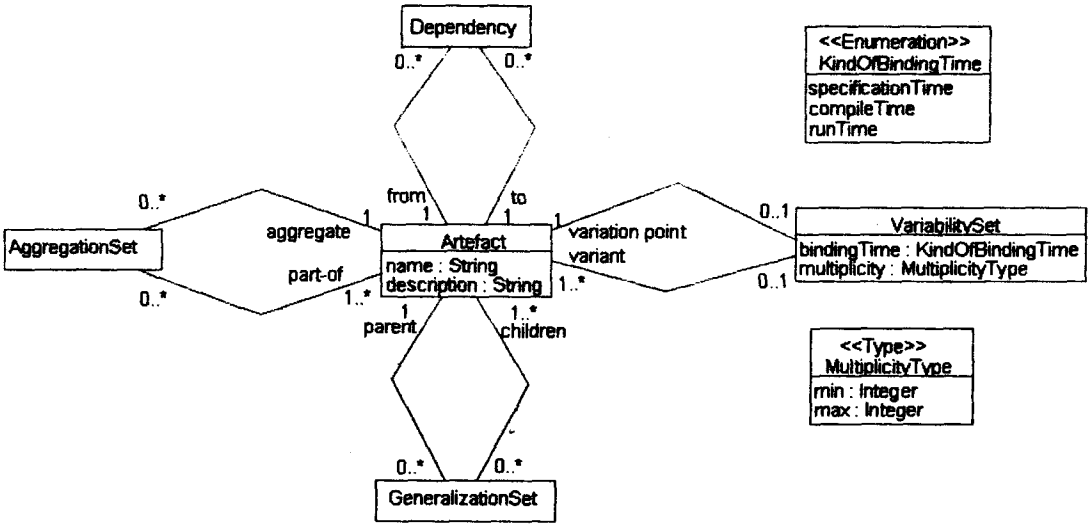
In addition to the artefacts, it is also necessary to establish relationships among artefacts of the SRS. Therefore, we have identified several kinds of relationships which are described below. An artefact can be refined forming a hierarchical structure. Three basic kinds of refinements are included: *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* and *bindingTime* must be specified (see section 2.5). 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.

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, 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 *Require* 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.

**FIGURE 1**
**Metamodel for the Core Concepts**




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 indicate 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.

**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:

(i) To define the types of artefacts that are relevant to the model. This permits the inclusion of the needed artefacts by means of specialization. Whenever a new artefact has to be defined a specialization is specified using *Artefact* or another artefact already defined as a base class. This child artefact inherits any attribute that has been defined by its parent.

(ii) To establish the refinement relationships of interest. If necessary, any additional refinement type could be added as a specialization of any of the existing ones (*AggregationSet*, *GeneralizationSet* and *VariabilitySet*) or as a new and independent one.

(iii) 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.

(iv) 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 relationship is defined, all the attributes defined in the parent are also inherited.

(v) To formalize artefacts and relationships. Those constraints to be applicable on artefacts and relationships have to be described. OCL (35) (Object Constraint Language) has been selected for this activity because it is a well-known and extended proposal for this purpose. In addition, there is a wide set of tools (36) 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 relationships described 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 OCL step (v).

## 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.

**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 | |
| | **Variant** | An artefact with a Variability or Generalization Rel. | | | Establishing this relation on an artefact | Establishing this relation on an artefact |
| Variability Management | **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*) | | | |

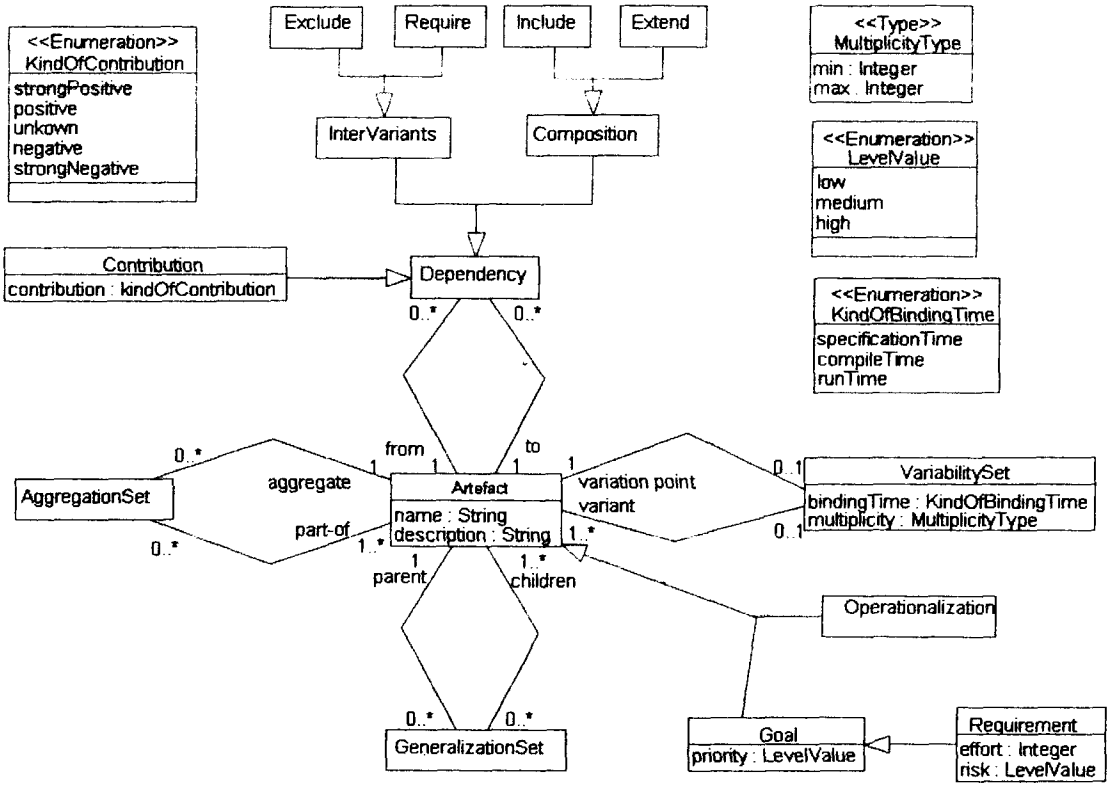
This project exhibits several specific needs in terms of requirements specification, such as, the variability inherent in the family of robots to be handled; the high incidence of non-functional requirements (reliability, performance, safety, etc) which crosscut functional ones; the need to evaluate alternative designs 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 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 (33), 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 which is based on KAOS and the NFR Framework proposals. This Goal Model has been extended (32) 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. We have also incorporated the variability management in order to apply the model to the definition of product lines and dynamic architectures.

## Goal Model of ATRIUM

Figure 2 shows the metamodel of the ATRIUM Goal Model. The artefacts and the identified relationships shown in Figure 2 correspond to the application of steps described in section 3.2. By applying step (i), three kinds of artefacts are identified to support a Goal-Oriented approach (section 2.3): *Goal*, *Requirement* and *Operationalization*. *Goal* and *Operationalizations* are defined by inheriting from *artifact*. 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) is not applied because no additional refinement relationship is 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 are identified. *Intervariant* expresses a dependency between variants. This dependency can be either *Require* or *Exclude*, using the semantics 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 are identified, i.e., *Include* or *Extend*; however, these relationships will model composition dependencies for crosscutting requirements (section 2.4). In addition, a *Contribution* dependency from one artefact to another is included to establish 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 *GeneralizationSets* are possible between *Requirements* or *Goals*.

## MetaEdit+: Tool Support for Modeling and Metamodeling

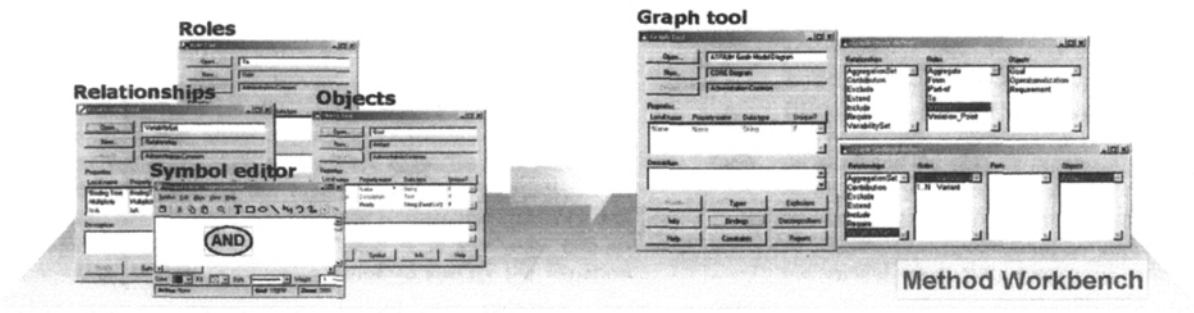
In order to provide tool support for our proposal we have studied three possibilities: (a) to implement a tool specially suited for our approach; (b) to define a UML Profile associated to our metamodel and use a CASE tool based on UML for providing visual modeling by means of UML stereotypes; and

(c) to use a Meta-CASE tool for defining directly our metamodel, attaining immediate modeling support. Option (a) is the most flexible but it requires more effort. We have tested option (b) (30) 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 with 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 generated by means of model transformation or following some mappings.

**TABLE 2**
**OCL Constraints for ATRIUM Goal Model**

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

**FIGURE 3**
**Method Workbench: Tool Support for Metamodeling**




Currently, we are using a Meta-CASE tool namely MetaEdit+ tool (29). It is a Domain Specific modeling tool that allows an easy metamodel specification and its exploitation by keeping models synchronized with any change performed on its 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; and MetaEdit+, which automatically gives the user full CASE tool functionality according to the metamodel description.

Method Workbench is based on the **GOPRR** metamodeling language (29) 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 they connect. 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.

**Exploiting MetaEdit+ in our Case Study**

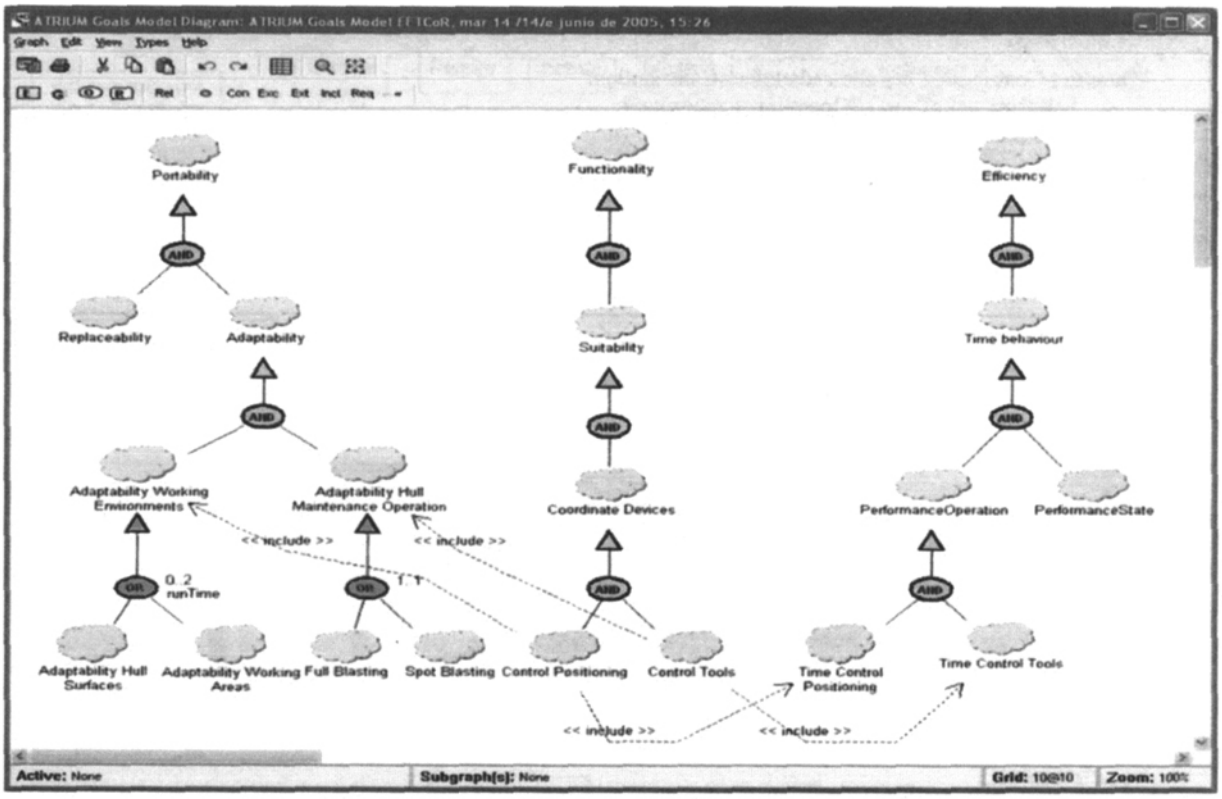
The first step has been to establish the mapping from the metamodel of the ATRIUM Goal Model to a MetaEdit+ metamodel. We have defined *artefact*, *Goal*, *Requirement* and *Operationalization* as **Objects** with their related properties. Afterwards, *AggregationSet*, *Dependency*, and *VariabilitySet* 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+ Meta-CASE provides us with full CASE tool functionality for specifying ATRIUM Goal Models: MetaEdit+ (Figure 4). In addition, thanks to the report facilities of MetaEdit+ we have defined specific reports for verification and analysis of models. We should point out that this facility allows us to generate XMI files for each model. It facilitates its checking by means of any of the existing tools of OCL (36) by 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 our case study concerns. 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*. In our example, either the goal "Adaptability Hull Surfaces" or the goal "Adaptability Working Areas" had to be satisfied in order to achieve a system that was self-adaptive to different working environments. Both subgoals were satisfied by means of characteristics added at runtime. The 0..2 multiplicity indicates that the goal "Adaptability Working Environments" is optional.

**FIGURE 4**
**Modeling a Part of the Goal Model of the EFTCoR System by Means of MetaEdit+**




The functionality "Coordinate Devices" was refined into two subgoals: "Control Positioning" and "Control Tools." 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 labeled as *Include*.

**Discussion of Results**

Several conclusions were achieved after the use of this customized metamodel along with its tool support. First of all, it is important to highlight its ability to cope with all the specific needs of this project. Previously, Use Cases and IEEE 830-1998 were used for specification purposes. 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 artifacts throughout the process development.

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. One of the key points for this project was to provide a clear notation where every stakeholder could easily understand the specification, due to the fact 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.

## RELATED WORKS

This section focuses on describing works that include some of the RE approaches described in section 2. It is worthy of note 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, none of them provides 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 (32) and (47) exploit a Goal Model to identify and specify aspects in early stages. Navarro et al (32) 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 (47) identify the crosscutting whenever an operationalization contributes to the satisfaction of several goals.

Brito et al (3) present an extension of the Use Case diagram for its application to AORE. 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 (2) 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 (16) and (31) 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 UC diagrams by means of the introduction of stereotypes that allow variability concepts to be modeled. Thus, for example, when a Use Case specifies a functionality that some products of the family have, it is stereotyped as «variant». The proposal presented by Halmans and Pohl (16) is perhaps the more extensive because it is the only one to incorporate its own graphical notation to specify the variation points and the 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 (14) and (17). 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.

We have analyzed some current tools for the management of requirements such as DOORS (9), IRqA (19) and RequisitePro (40). Even though they exhibit a great potential for defining artefacts, they only offer one type of dependency (the traceability specified between artefacts characterized as *from* or *to*) and only one refinement (using the *AggregationSet* semantic).

## 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 the selection, integration and customization of 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 semantics. We have dealt 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, five representative techniques were studied. In this proposal, according to the project's specific needs, a proper integration is provided as well as scalability from simple up to other more sophisticated RE techniques. The main features of this proposal are:

Definition of a metamodel which includes a core of concepts corresponding to the essential expressiveness of some of the more popular and/or advanced approaches in RE.

Establishment of guidelines for adapting the metamodel to specific needs, according to the expressiveness. We have validated both the core and the extension presented of the metamodel by means of their application to the EFTCoR system. Currently, we are concerned with evaluating the stated method by using criteria like those proposed by (44).

This metamodel is the first step toward 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 can only be efficiently satisfied by means of a software tool. The proposed metamodel was implemented as a UML profile in 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 MetaEdit+ Meta-CASE tool which has allowed us to validate our proposal 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 RE approaches in the context of our proposal: *View Points* (13), *Problem Frames* (21) and *Cognitive Mappings* (42, 43). 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 instance, between conceptual maps described by means of Cognitive Mappings and services of the system-to-be described using goal 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 (28) or Katz and Rashid (22) can be useful as a reference to achieve this aim.

## REFERENCES


1. Baskerville, R.L. and A.T. Wood-Harper. "A Critical Perspective on Action Research as a Method for Information Systems Research," **Journal of Information Technology**, 11:4, 1996, pp. 235-246.
2. Brito, I. and A. Moreira. "Integrating the NFR Framework in a RE model," Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, Lancaster, UK, March 22, 2004.
3. Brito, I. and A. Moreira. "Towards a Composition Process for Aspect-oriented Requirements," Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, Boston, MA, March 17, 2003.
4. Bühne, S., G. Halmans, and K. Pohl. "Modeling Dependencies between Variation Points in Use Case Diagrams," 9th International Workshop on Requirements Engineering: Foundation for Software Quality, Klagenfurt/Velden, Austria, June 16-17, 2003.
5. Chung, L., B.A. Nixon, E. Yu, and J. Mylopoulos. **Non-Functional Requirements in Software Engineering.** Boston: Kluwer, 2000, p. 472.
6. Clements, P. and L. Northrop. **Software Product Lines - Practices and Patterns.** Boston: Addison-Wesley, 2001, p. 530.
7. Cockburn, A. **Writing Effective Use Cases.** Boston: Addison Wesley, 2000, p. 304.
8. Dardenne, A., A. van Lamsweerde, and S. Fickas. "Goal-directed Requirements Acquisition," **Science of Computer Programming,** 20:1-2, 1993, pp. 3-50.
9. DOORS. http://www.telelogic.com/products/doorsers/doors/, 2005.
10. EFTCOR: Environmental Friendly and Cost-effective Technology for Coating Removal. European Project, 5th Framework Program (GROWTH G3RD-CT-00794), 2003.
11. Elrad, T., M. Aksits, G. Kiczales, K. Lieberherr, and H. Ossher. "Discussing Aspects of AOP," **Communications of the ACM,** 44:10: 2001, pp. 33–38.
12. Finkelstein A. and I. Sommerville. "The Viewpoints FAQ," **Software Engineering Journal**, 11:1, 1996, pp. 2-4.
13. Gonzales-Baixauli, B., J.C.S. Prado Leite, and J. Mylopoulos. "Visual Variability Analysis with Goals Models," 12th Int. Conf. on Requirements Engineering, Kyoto, Japan, September 6-10, 2004, pp. 198-207.
14. van Gurp, J., J. Bosch, and M. Svahnberg. "On the Notion of Variability in Software Product Lines," Working IEEE/IFIP Conference on Software Architecture, Amsterdam, The Netherlands, August 28-31, 2001, pp. 45-54.
15. Halmans, G. and K. Pohl. "Communicating the Variability of a Software Product Family to Customers," **Software and Systems Modeling**. 2:1, 2003, pp. 15-36.
16. Hui, B., S. Liaskos, and J. Mylopoulos. "Requirements Analysis for Customizable Software Goals-Skills-Preferences Framework," 11th IEEE Int. Requirements Engineering Conference, Monterey Bay, California, USA, September 8-12, 2003, pp. 117-126.
17. IEEE Std. 830-1998. IEEE Recommended Practice for Software Requirements Specifications, Volume 4: Resource and Technique Standards, IEEE Software Engineering Standards Collection.
18. IrqA, http://www.irqaonline.com, 2005.
19. ISO/IEC Standard 9126-1 Software Engineering- Product Quality-Part 1: Quality Model, ISO Copyright Office, Geneva, June 2001
20. Jackson, M. **Problem Frames: Analyzing and Structuring Software Development Problems.** Boston: Addison-Wesley, 2000, p. 416.
21. John, I. and D. Muthig. "Tailoring Use Cases for Product Line Modelling," Int. Workshop on Requirements Engineering for Product Lines, Essen, Germany, September 10, 2002.
22. Katz, S. and A. Rashid. "From Aspectual Requirements to Proof Obligations for Aspect-Oriented, Systems," 12th IEEE Int. Requirements Engineering Conference, Kyoto, Japan, September 06-10, 2004, pp. 48-57.
23. Kavakli, E. and P. Loucopoulos. "Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods," Information Modeling Methods and Methodologies, IDEA Group, Hershey, USA, 2005, pp. 102 – 124.
24. Kelly, S., K. Lyytinen, and M. Rossi. "METAEDIT+ A Fully Configurable Multi-User and Multi-tool CASE and CAME Environment," 8th Conference on Advanced Information Systems Engineering, Heraklion, Creta, May 1996, pp. 1-21.
25. Kiczales, G., J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.M. Loingtier, and J. Irwin, "Aspect-Oriented Programming," 11th European Conference on Object-Oriented Programming, LNCS 1241, Finland Jyväskylä, Finland, June 9-13, 1997, pp. 220-242.
26. van Lamsweerde, A. "Goal-Oriented Requirements Engineering: A Guided Tour," 5th IEEE International Symposium on RE, Toronto, Canada, August 2001, pp. 249-263.
27. Letelier, P., E. Navarro, and V. Anaya. "Customizing Traceability in a Software Development Process," Information Systems Development Advances in Theory, Practice, and Education, Springer Science+Business Media, Inc., 2005, pp. 137-148.
28. Letier, E. and A. van Lamsweerde. "Deriving Operational Software Specifications from System Goals," **ACM SIGSOFT Software Engineering Notes**, 27:6, 2002, pp. 119-128.
29. von der Maßen, T. and H. Lichter. "Modeling Variability by UML Use Case Diagrams," Int. Workshop on RE for Product Lines, Essen, Germany, Sept. 9, 2002.
30. Navarro, E., I. Ramos, and J. Pérez. "Software Requirements for Architectured Systems," 11th IEEE Int. Requirements Engineering Conference, Monterey, September 8-12, 2003, pp. 365-366.
31. Navarro, E., P. Letelier, and I. Ramos. "Goals and Quality Characteristics: Separating Concerns," Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, Vancouver, Canada, October 25, 2004.
32. Navarro, E., P. Letelier, and I. Ramos. "UML Visualization for an Aspect and Goal-Oriented Approach," 5th Aspect-Oriented Modeling Workshop, Lisbon, Portugal, October 11, 2004.
33. Nuseibeh, B. and S. Easterbrook. "Requirements Engineering: A Roadmap," 22nd Int. Con. on Software Engineering, Future SE Track, Limerick Ireland, June 4-11, 2000, pp. 37-46.
34. OCL 2.0 Specification, OMG, http://www.omg.org/docs/ptc/05-06-06.pdf, 2005.
35. OCL tools & services http://www.klasse.nl/ocl/ocl-services.html, 2005.
36. Oreizy, P., N. Medvidovic, and R.N. Taylor. "Architecture-

Based Runtime Software Evolution," 20th Int. Conference on Software Engineering, Kyoto, Japan, April 1998, pp. 117-186.
37. Rashid, A., A. Moreira, and J. Araújo. "Modularization and Composition of Aspectual Requirements," 2nd Int. Conf. Aspect-Oriented Software Development, Boston, March 17 -21, 2003, pp. 11-20.
38. Rashid, A., P. Sawer, A. Moreira, and J. Araújo. "Early Aspects: A Model for Aspect-Oriented Requirements Engineering," 10th IEEE Joint International Conference on Requirements Engineering, Essen, Germany, Sept. 9-13, 2002, pp. 199-202.
39. RequisitePro, http://www.306.ibm.com/software/awdtools/reqpro/, 2005.
40. Rational Rose, http://www-306.ibm.com/software/awdtools/architect/swarchitect/, 2005.
41. Siau, K. and L. Lee. "Are Use Case and Class Diagrams Complementary in Requirements Analysis? An Experimental Study on Use Case and Class Diagrams in UML," **Requirements Engineering**, 9:4, 2004, pp. 229-237.
42. Siau, K. and X. Tan. "Evaluation Criteria for Information Systems Development Methodologies," **Communications of the AIS**, 16, 2005, pp. 856-872.
43. Siau, K. and X. Tan. "Improving the Quality of Conceptual Modeling Using Cognitive Mapping Techniques," **Data and Knowledge Engineering**, 55:3, 2005, pp. 343-365.
44. Siau, K. and X. Tan. "Technical Communication in Information Systems Development: The Use of Cognitive Mapping," **IEEE Trans. on Professional Communications**, 48:3, 2005, pp. 269-284.
45. Suzuki, J. and Y. Yamamoto. "Extending UML with Aspects: Aspect Support in the Design Phase," Aspect Oriented Programming Workshop, Lisbon, Portugal, June 1999.
46. Yijun Y., J.C.S.P. Leite, and J. Mylopoulos. "From Goals to Aspects: Discovering Aspects from Requirements Goal Models," 12th IEEE Int. Requirements Engineering Conference, Kyoto, Japan, Sept. 6-10, 2004, pp. 38-47.
47. Yu, E., L. Liu, and Y. Li. "Modeling Strategic Actor Relationships to Support Intellectual Property Management," 20th Int. Conference on Conceptual Modeling, LNCS Vol. 2224, Yokohama, Japan, November 27-30, 2001, pp. 164 - 178.
48. Zave, P. "Classification of Research Efforts in Requirements Engineering," **ACM Computing Surveys**, 29:4, 1997, pp. 315-321.