UML Visualization for an Aspect and Goal-Oriented Approach

Elena Navarro
Department of Computer Science, UCLM
Avda. España S/N
Albacete, Spain
+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
+34 96 387 7007 ext. 73589
letelier@dsic.upv.es

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

ABSTRACT

The Goal-Oriented requirement engineering approach offers important advantages for a deeper study of software requirements. Some of them are supported for reasoning about design alternatives and traceability between requirements and software architecture. However, in complex systems, requirements specifications suffer from crosscutting, which affects elaboration, readability and maintainability of the specification, even when using a Goal-Oriented approach. Separation of concerns, included in Aspect-Oriented Requirement Engineering provides an elegant and effective solution to cope with this problem. In this work we present a model for requirement specification which integrates Goal-Oriented and Aspect-Oriented approaches. This model is included in ATRIUM, a methodology for concurrent definition of requirements and software architecture. Using a UML profile we give graphical notation to our model allowing its support in most CASE tools based on UML.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: *Requirements Specification – Elicitation methods, Languages and Methodologies.*

General Terms

Algorithms, Documentation, Design, Reliability, Standardization, Languages, Verification.

Keywords

Aspect Oriented, Goal Oriented, Software Requirements, Software Architecture.

1. INTRODUCTION

The requirements specification involves a number of challenges related to quality characteristics that must be achieved, like those described in IEEE 830-1998 [10] standard as: correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, and traceable. The Goal-Oriented approach [12] has aroused interest in researchers because of its facilities for elaboration and deeper analysis of requirements specification. In this approach, detailed requirements are obtained by stepwise refinement starting from general system goals (or concerns). This refinement continues until requirements are assigned to system agents which are able to operationalize them. Thus, the Goal-Oriented paradigm has two advantages that make it especially suitable to guide the selection among several architectural design alternatives:

- Its ability to specify and manage positive and negative interactions between goals [6] allows the analyst to reason about design alternatives.
- Its capability to trace low-level details back to high-level concerns [7] is very appropriate to bridge the gap between architectural models and requirements.

However, like in other approaches for requirements specification, when dealing with complex and/or large systems, crosscutting of elements usually appears in the specification. This crosscutting manifests itself by affecting negatively readability and maintainability of the specification. The Aspect-Oriented Requirement Engineering (AORE) [20] identifies and manages the crosscutting in an elegant and effective way, based on separation of concerns.

ATRIUM [17] (Architecture generated from RequIrements applying a Unified Methodology) is a methodology that emphasizes the concurrent definition of requirements and software architecture. In the ATRIUM context, the Goals Model plays an essential role conducting the software architecture generation and validation process. This Goals Model integrates Goal-Oriented and Aspect-Oriented approaches, offering the advantages of each of them.

On the other hand, one of the main concerns in any modern modelling approach is to include a notation that allows a visual representation of the produced specifications. UML seems to be the appropriated candidate to be used as a language for software modelling. Thanks to its extension mechanisms around the concept of UML profile, UML can be extended and adjusted to the particular needs of our Goals Model. Furthermore, most CASE tools provide support for UML and allow us, in different

5th Aspect-Oriented Modeling Workshop'04, October 11, 2004, Lisbon, Portugal.

levels, to work with these kinds of extensions based on UML profiles.

In Requirements Engineering, there are some works defining notations based on UML in the Goal-Oriented [9] approach as well as in the Aspect-Oriented one [4]. However, none of them integrates both perspectives in the way required by our Goal Model.

The aim of this work is to present the Goal Model of ATRIUM and the definition of a corresponding UML profile. This work is structured as follow: next section is giving a brief review of ATRIUM, its intention and activities. Section 3 shows how the profile has been described for our proposal. Eventually, section 4 and 5 describe the related works and the reached conclusions.

2. ATRIUM: REQUIREMENTS AND SOFTWARE ARCHITECTURES

ATRIUM is a methodology oriented to the concurrent definition of Software Architecture (SA) and Requirements. In ATRIUM, decisions at architectural level are made to satisfy specific software requirements. With this aim, ATRIUM provides the analyst with guidance, along an iterative process, from an initial set of user/system needs until the instantiation of the architecture, specified by means of PRISMA model [19]. PRISMA is an architecture description language that allows us to define dynamic architectures.

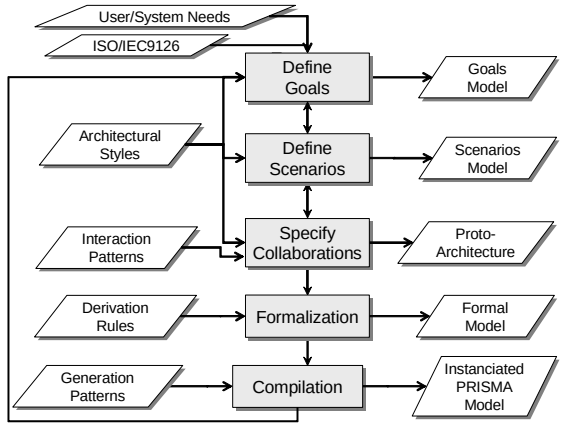

Figure 1 ATRIUM: activities and artefacts

ATRIUM entails five activities (Figure 1) to be iterated on in order to define and refine the different artefacts and allow the analyst to reason about partial views, both of requirements and of architecture. The *Define Goal* activity allows one to identify the different concerns of the software as well as the crosscutting between them. The main idea is to determine which concerns are candidate to be classified as aspects, in the PRISMA specification, and realize them through aspects integrated into components and/or connectors.

The Goals Model, one of the artefacts generated by applying ATRIUM and previously presented at [16], was inspired by the NFR Framework [6] and KAOS [7]. Goals, requirements, contribution, etc, are also elements in its construction. Nevertheless, the main difference, that this proposal exhibits, is that only one model is introduced for functional and non-functional requirements definition. It is due to both types of requirements are highly relevant in the architectural definition.

Finally, our proposal introduces concepts from the Aspect Oriented Software Development (AOSD [3]) approach. Terms as *weaving relationships*, *crosscutting*, etc, have been brought in our approach, as we already stated in a previous work [18]. In our approach, the Goals Model allows us to identify and manage the involved concerns, in such a way that every identified goal, along the process, is considered initially as a concern.

2.1 Building blocks

The Goals Model provides a number of abstractions in terms of which constraints on the software system have to be defined. A key element introduced in its construction is a *goal*. It is defined as an objective that the system-to-be should achieve [13], i.e., a constraint or obligation that the system should meet. In its definition it is characterized as *Functional* or *Non-Functional*, according to the type of need or expectation it refers to:

- *Functional goals or requirements* describe services that the software provides, i.e., the transformations the system performs on the inputs.
- *Non-Functional goals or requirements* describe conditions or constraints that the software must satisfy; they refer to how the services are provided, for instance, in terms of performance, adaptation, security, etc. We are highlighting them because they are especially meaningful in terms of the architecture of the end system.

Additionally, other characteristics have to be stated when a goal is defined. For instance, each goal has to be classified according to its *priority*, from *very high* to *very low*, for the system-to-be. This classification helps the analyst to focus on the important issues. These priorities can arise from several factors: organizational ones when they are critical to the success of the development, constraints on the development resources, etc.

Moreover, a set of *preconditions* and *postconditions* should be identified. Preconditions establish which situations must hold before some operation is performed. Postconditions define the conditions that have to be satisfied after some operation is performed. Their evaluations help us to determine the best design alternatives among those that satisfy the postconditions for the established goals. For their description a variant of dynamic logic [14], which includes deontic operators for expressing permission and obligation, is used but to go into more details on this topic is out of the scope of this paper.

Similarly to goals, another used element in the Goals Model construction is known as *requirement*. They also specify a need or constraint on the end system, although its main difference regarding goals is its capability to be assigned to and realized by a set of agents. Additionally, also due to its capability it can be verified at the end system. Its textual notation is similar to that defined for goals. However, in this case, both postconditions and preconditions should be defined for each requirement.

Aside from goals and requirements, another building block for the Goals Model is the *operationalization*. When an analyst has refined the initial set of goals, he/she must offer a set of solutions that allow the system to achieve the established goals. An operationalization is a solution that provides the target system with architectural design choices which meet the users' needs and

expectations. They are called *operationalizations* because they describe the operation of the system, i.e., the system behaviour, to meet functional and non-functional requirements.

It can be noticed that the alternative solutions to satisfy a given requirement are not described on each operationalization. On the contrary, it is in the Scenarios Model where these solutions are expressed. However, operationalizations are introduced in the Goals Model to conceptually represent each solution so that relationships among the different alternatives can be established within the Goals Model. Operationalizations establish a coupling between the Scenarios Model and the Goals Model, establishing the traceability between operationalizations and a specific view of the Scenarios Model.

Additionally, we want to notice that operationalizations are not functionally or non-functionally characterized like goals and requirements. This is because the same solution can be associated to different goals, both functional and non-functional.

2.2 Relationships for the Refinement Process

The stated building blocks, goals, requirements and operationalizations, are inter-related by means of a set of *relationships*. They are in charge of gluing the different elements to complete the model and enhance its cohesion. Moreover, their relevance is not only restricted to this gluing but also they allow the analyst to introduce the rationale for the system design. The decomposition of goals or how an operationalization positively or negatively contributes to a goal can be defined via relationships.

They are applied via a stepwise refinement process which takes an informal set of user/system needs, usually stated in natural language, as well as the framework for an initial selection of concerns provided by the ISO/IEC 9126 [11]. Both act as inputs to begin with the model definition, and using some of the proposals to identify goals [2]. In such a way, every refinement step generates new goals/requirements and/or operationalizations in the model. Therefore, the analyst has to deal with a reduced set of building blocks at each step.

There are two types of refinements that can be applied: intentional and operational. The former describes how a goal can be reduced into a set of subgoals/requirements via *AND/OR/XOR* relationships. The latter depicts how a set of solutions address a requirement by means of *AND OPERATIONALIZE* / *OR OPERATIONALIZE* / *XOR OPERATIONALIZE* relationships. Both building blocks and relationships are structured as an acyclic graph, where the refinement is achieved along the structure, from the higher to the lower level, by applying intentional and operational refinements.

Every goal, which is too coarse-grained, is refined in a set of subgoals which are a decomposition of the original one. An *AND* relationship between a goal *GoalX* and a set of sub-goals G_1, …, G_N or requirements R_1, …, R_N is established if the whole set of sub-goals and/or requirements has to be satisfied in order to satisfy *GoalX*. An OR relationship is established if *GoalX* is satisfied if at least a sub-goal or requirement is satisfied. Finally, a XOR relationship is introduced if *GoalX* is satisfied when only a sub-goal or a requirement from this set can be satisfied.

An operational refinement deals with requirements and operationalizations. The alternative solutions for each goal are established by means of this decomposition. There can be a large number of valid operationalization methods that are applicable to a requirement. In such a case, it is up to the analyst to examine the impact of such methods on other requirements and decide on what and how many operationalizing methods must be applied via *AND OPERATIONALIZE/ OR OPERATIONALIZE/ XOR OPERATIONALIZE* relationships. An *AND OPERATIONALIZE* relationship relates the set of mandatory solutions for a requirement. On the other hand, whenever several alternative solutions can be provided for a requirement, the analyst can introduce them in terms of the *OR OPERATIONALIZE* relationship. Finally, whenever several alternative solutions can be provided for a requirement, but only one can be selected for the end system, the analyst can introduce them by using *XOR OPERATIONALIZE* relationship. OPERATIONALIZE relationships do not only relate operationalizations to requirements but also to other solutions. It is used to refine the operationalizations down to other simpler ones, i.e., to describe how a solution can be expressed in terms of a set of simpler solutions.

A set of symbols [++|+|#|-|--] are used to characterize the way an operationalization contributes to achieve a requirement. Symbols ++ and + describe a strong positive and positive contribution, i.e., it provides a sufficient or partially sufficient solution, respectively, to satisfy the related requirement. On the other hand, symbols -- and - describe a strong negative or negative contribution, i.e., the operationalization prevents or partially prevents, respectively, the satisfaction of the related requirement. The # symbol is introduced to specify operationalizations whose impact (positive or negative) is neutral at the moment. This is the default value.

The intentional refinement is iteratively applied to the set of goals ending up when every sub-goal can be operationalized, i.e., when a requirement can be defined. Similarly, the operational refinement is iteratively applied to operationalizations until the corresponding scenarios are simple enough.

Another type of relationship that can be introduced is *conflict*. It can be set up among two goals/requirements if an incompatibility appears between them, in other words, whenever the satisfaction of a goal/requirement prevents the satisfaction of another goal/requirement.

Finally, other relationship that can appear between goals/requirements is called *weaving*. When a goal/requirement crosscut other goals/requirements, a *weaving* relationship is established. These relationships can be charecterized according to some of the traditional AOSD weaving mechanisms, like *before* and *after*. This allows us to express how a piece of goal/requirement specification (from the aspect point of view) is incorporated inside some other goal/requirement specification. Other more specific weaving relationships could be used (like in [19]), but we suggest to do this refinement in the specific domain context of the system.

3. DEFINING A UML PROFILE

Along the process of definition of the ATRIUM profile, we were faced with several challenges. One of them was related to the satisfaction of the requirements [1] that Aldawud et al stated for defining a UML profile for AOSD:

(1) *The Profile shall enable specifying, visualizing, and documenting the artifacts of software systems based on*

Aspect-Orientation. This requirement has been satisfied by means of the stereotypes and their visual representation as it is described below.

(2) *The Profile shall be supported by UML (avoid "Heavy-weight" extension mechanisms), this allows a smooth integrating of existing CASE tools that support UML*. This requirement is also satisfied due to our UML profile has been defined according to the established construction rules in the UML specification [22].

(3) *The Profile shall support the modular representation of crosscutting concern.* The separation of concerns provided by the goal oriented approach, along with the defined weaving relationship, allows us to identify and manage crosscutting in an early stage.

(4) *The Profile shall not impose any behavioural implementation for AOSD, however it shall provide a complete set of model elements (or Stereotypes) that enable representing the semantics of the system based on Aspect-Orientation*. No constraint has been defined about the implementation, only a proper semantic related to the way we use the ATRIUM elements at the requirements stage.

With the aim of describing this UML support for the ATRIUM elements, two tasks have been carried out: the Metamodel description of the Goals Model (section 3.1) and the UML profile associtated to the metamodel (section 3.2 and 3.3).

3.1 Metamodel Description

The metamodel, shown in Figure 2, defines the abstract syntax for specifying Goals Model in ATRIUM. We use the prefix "A-" (from **A**TRIUM) to name metaclasses. One of the model elements is *AGoal* which allow us to describe every concern of the Goals Model, as we described above. *AGoal* has a meta-attribute called *type* which describe the type of the goal, i.e., functional or non-functional. By means of the generalization, *ARequirement* inherits this meta-attribute, i.e., it can be also typed as functional and non-functional.

Several relationships are described in the model. One of them is *AGoal/RequirementGroup* which is used for representing the refinement of a *AGoal* element into a set of goals/requirements. As shown, the meta-attribute *joinType* is defined that allow us to describe the type of refinement (AND, OR or XOR) by means of the enumeration *JoinType*. In addition, *AWeaving* relation is introduced with a meta-attribute called *matchPoinType* which allow us to specify the way the weaving is applied. Its type is *MatchPointType* that has been defined as an *Enumeration*. Eventually, a *conflict* relationship is established to describe the relation between conflicting goals/requirements. Every one of the previous relationships is inherited by *ARequirement*, that is, they can be used for it with the same meaning.

AOperationalizationGroup allow us to represent how an element *ARequirement* has associated a set of operationalizations, and the way each element *AOperationalization* contributes to its realization. The allowed contributions are described with the enumeration *ContributionType.*

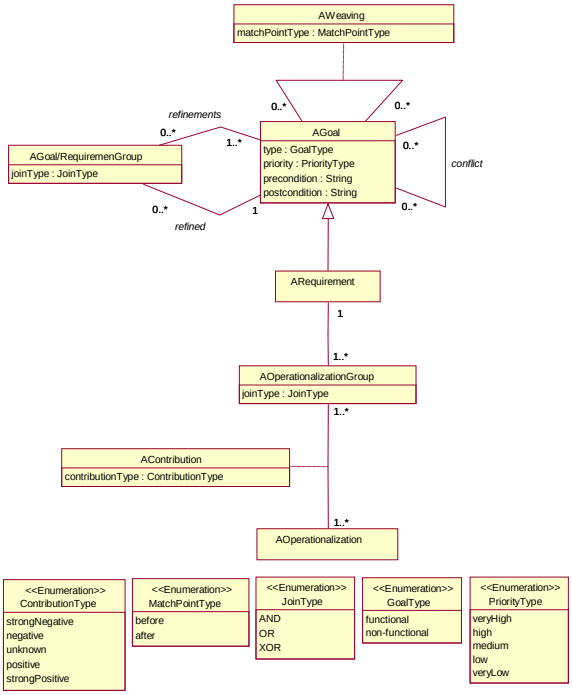

Figure 2 Metamodel of the Goals Model

3.2 UML Profile

This profile defines an extension to the reference UML 2.0 metamodel with the purpose of tailoring it to our Goals Model but keeping its semantics. Metaclasses of the UML metamodel are extended by means of a mechanism called *extension* that is represented with an arrow with filled end, as can be observed in Figure 3. For instance, *AContribution* is defined as a stereotype which extends the UML metaclass *Generalization*. Additionally, those specific features of the ATRIUM elements are defined by means of meta-attributes (tagged-values in 1.5) of the stereotypes to be described. That is the case of the meta-attribute *joinType* in *AContribution*. In a similar way, each element, which appears in the Goals Model Metamodel, is mapped to a stereotype and avoiding "Heavy-weight" extension mechanisms, as can be observed in the Figure 3.

Additionally, a set of well-formed rules, in the context of a profile, are defined to introduce the specific needed semantic. Several structural constraints can be easily extracted from the Metamodel in Figure 2. For instance, a *conflict* can be described among two Goals/Requirements, and similarly for the *weaving* relationship. However, the more relevant rules are those related to the domain semantic. These rules express constraints such as: a *ARequirement* always has to be defined by refining one or more *AGoal*.

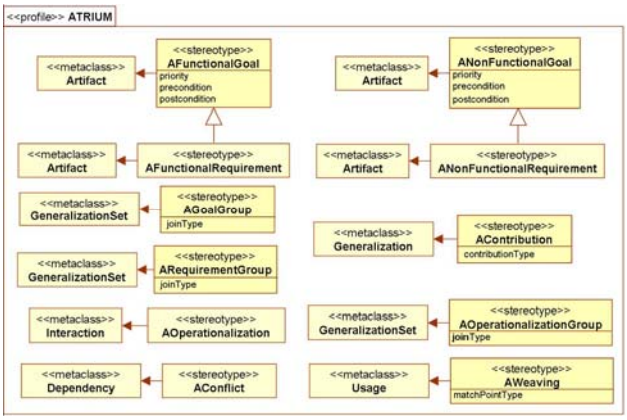

Figure 3 Goals Model Profile

3.3 Graphical Notation

This section outlines the graphic elements that may be shown in a Goals Model. Along with the visual notation for every described stereotype, a brief description is also provided. All of them are described in Table 1.

Table 1 Graphics nodes included in Goals Models

Node Type	Notation	Description
AFuntionalGoal		Indicates a software functional goal that the system must satisfy
ANonFuntionalGoal		Indicates a software non-functional goal that the system must satisfy
AGoalGroup	<<GoalGroup>>	Indicates a refinement of one goal in subgoals or requirements
AFunctional-Requirement		Indicates a functional goal which is verifiable and assignable to an agent
ANonFunctional-Requirement		Indicates a non-functional goal which is verifiable and assignable to an agent
AOperationalization		Indicates a scenario describing how agents collaborate to support a requirement
AOperationalization-Group	<<Operationalization Group>>	Indicates a set of possible operationalizations for a requirement . Each operationalization is modeled as a AContribution.
AContribution		Indicates how one operationalization contributes to one requirement
AConflict	<<Conflict>>	
AWeaving	<<weaving>>	Indicates a weaving relationship between two goals or requirements

4. AN EXAMPLE

This section illustrates how we have applied our proposal in the context of the European Project Environmental Friendly and cost-effective Technology for Coating Removal (EFTCoR) [8]. The scenario of this project is the hull maintenance operations of ships. Mainly, it addresses operations of coating removal, washing and re-painting of hull of ships by using a family of robots, that either perform different operations or the same operation but in a different way. The identified robotic teleoperation platform is integrated by the next subsystems:

(1) *Monitoring System*: encompasses the functionality concerning to the informational and managerial needs related to ship maintenance operation that is going to be accomplished.

(2) *Vision System*: allows the hull inspection of the working areas and provides information for automatically moving the robotic devices along the hull.

(3) *Recycling System*: retrieves the residues from the working areas and recycles them.

(4) *Robotic Devices Control Unit*: interacts with the other robotic devices with the aim of getting the needed information to control the different devices (positioning systems and cleaning tools) to be used in the maintenance tasks. It is accomplished according to the commands introduced by the operator.

Our case study focuses on the Robotic Devices Control Unit. Its architectural definition is highly relevant due to the fact that several constraints have to be satisfied in order to allow a dynamic behaviour of the system. This dynamism allows the EFTCoR to replace, at run time, each cleaning tools and positioning devices. Either change or operation has to be secure, providing a mean to stop it if any damage can be produced to the equipment, the environment or the operator. Moreover, every operation has to be scheduled to accomplish hard deadlines.

The graph on Figure 4 shows (part of) the Goals Model where the refinement of goals is reviewed. In this way, we observe how *Portability*, *Functionality* and *Efficiency* are some of the selected characteristics to become concerns for the EFTCoR system. Furthermore, this figure shows us some of the relationships of refinement that were established. For instance, the AND relationship for the goals *AdaptabilityWorkingEnvironment* and *AdaptabilityHullMaintenaceOperation* that was introduced to satisfy *Adaptability*.

On the other hand, crosscutting also appears in the specification. For instance the goals related to *Efficiency* and *Adaptability*. Both goals are applied to other goals such as *ControlPositioning* or *ControlTools*. Furthermore, conflict relationship can be described in this model, when *Security* and *Performance* are demanded goals for the system.

5. RELATED WORKS

Most works on profiles for AOSD has focused their efforts on the design stage, such as [1, 21]. However, there are no many proposals for the requirements stage. One of them was presented by Araujo et al [4]. They have described an extension for UML that provide support to a previous approach [15]. This one establishes the way non-functional requirements constrain functional requirements. In this way, functional requirements are described as Uses Cases and non-functional requirements as stereotyped Use Cases. The main problem, which this proposal shows, arises from a metamodel in permanent evolution, i.e., whenever a new non-functional requirement has to be specified, a new stereotype, with Use Case as base class, has to be defined which is a heavier mechanism than our proposal.

More related to the Goals Model, and the way it is used for eliciting requirements, is the proposal presented by Heaven and Finkelstein [9]. They have defined a UML profile (based on UML

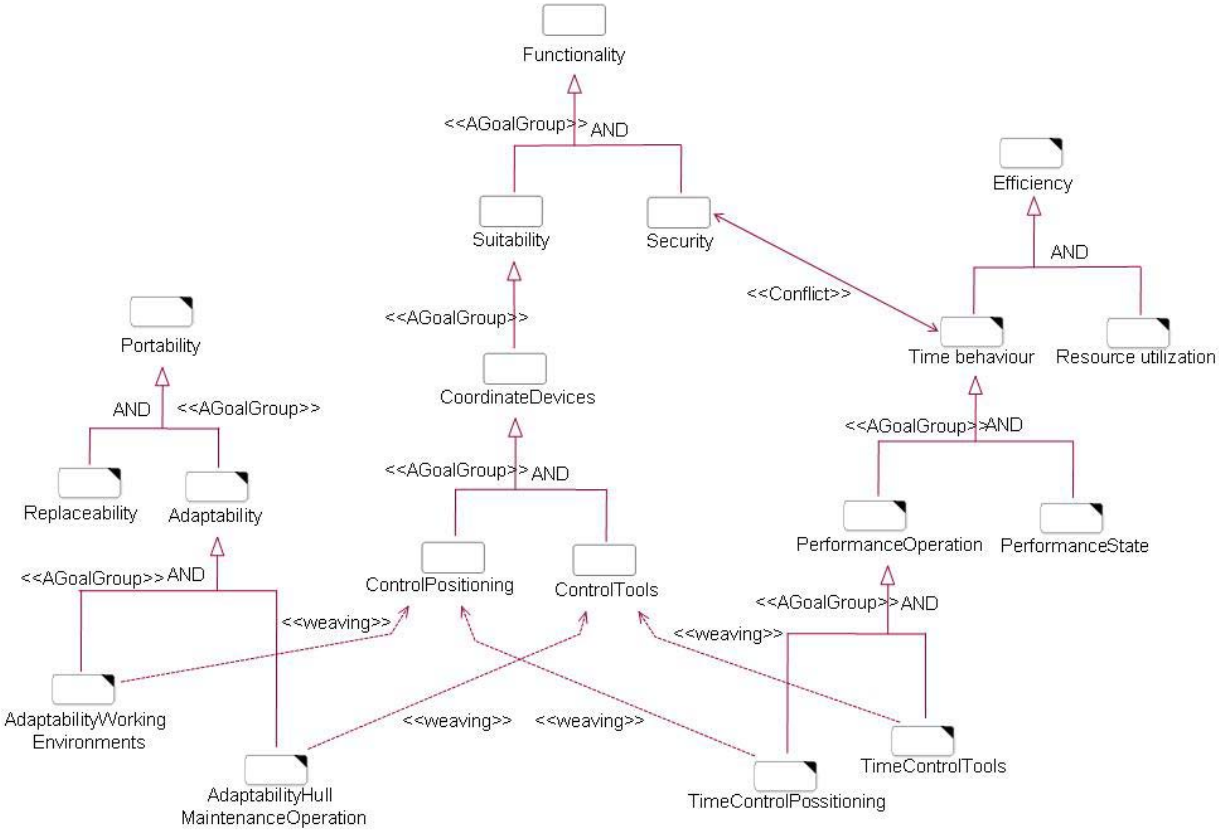

Figure 4 Partial view of the Goals Model for EFTCoR

1.4) for the Goals Model of KAOS. Several stereotypes are included that describe some common concepts with our proposal, for instance, *goals* and *AND/OR/XOR* refinement relationship. In this work, the metaclass Abstraction is used as base class for the stereotype <<reduces>>. This stereotype represents a refinement relationship between a goal and a subgoal. This alternative presents an important drawback when attempting to establish a characterization AND/OR/XOR for the whole refinement hierarchy, instead of establishing it for each subgoal refinement. In our approach, we use metaclass *GeneralizationSet* (incorporated in UML 2.0) for this purpose. It allows us to express the idea of hierarchy with the corresponding characterizations. Additionally, they do not provide the analyst with specific elements to describe the relation between functional and non-functional goals, as we have stated above, that is, the weaving relationship does not exist in their proposal. Additionally, they also introduce operationalizations and an associated model for its description. However, they do not use any interaction diagram even though they are defining the system behaviour. On the contrary, they define a set of new stereotypes to describe concepts such as *actions* or *events* instead of using the provided UML support by means of *activity or sequence diagrams*.

Another difference regards to dynamic logic has been selected as formalism instead of temporal logic that KAOS uses. This election is due to our previous experience with industrial case studies. The analyst comprehensibility and usability was greater when dynamic logic was introduced than temporal logic.

Brito and Moreira [5] have also introduced a Goal Oriented proposal, concretely, the NFR Framework [6]. Every identified concern is specified by using the best approach. In its case study Use Cases are used for functional requirements and SoftGoals Interdependency Graph (SGI) for non-functional requirement. Additionally, a template has to be fulfilled for every stated concern, where both the contributions and the required concerns are drawn. These provide information to detect the crosscutting concern whose specification is accomplished by using a table. This means that a diversity of notations is used in their approach. SGI does not play an important role along the process, only as a mean to describe non-functional requirements. This involves a greater effort in order to have a full comprehension of the system due to the specification is scattered over several artefacts.

The main advantage that our proposal offers is twofold. On one hand, only one single model is introduced for the integrated description of both functional and non-functional elements. On the other hand, our UML profile does not have to be modified to accommodate new non-functional requirements

6. CONCLUSIONS AND FUTURE WORKS

In this work we have presented the Goals Model included in ATRIUM. This Goal Model offers a relevant improvement to specification of requirements that takes the advantages of two prominent and modern approaches: Goal-Oriented and Aspect-Oriented Requirements Engineering. Furthermore, we have defined a UML profile to facilitate the practical use of our proposal in most current CASE tools. Using an example we have

illustrated the application of the Goal Model and the corresponding UML profile.

In our proposal the *aspect* concept does not explicitly appear as a constructor, as other works do. Instead, the candidate aspects implicitly arise on those goals/requirements with weaving relationships. This is due to the concept of aspect is specified in other models of the ATRIUM approach. For instance, when shallow components and connectors are identified and specified in the Scenarios Model, weaving relationships are taken into account for defining possible aspects. We have to consider that both architectural elements are defined by a gluing of aspects.

Several works are in progress related to the stated here and, mainly, to ATRIUM. They are related to the definition of the other involved models in ATRIUM and their corresponding UML visualization. We foresight the common modelling framework of UML will facilitate the management of traceability between involved models.

Also, a deep inspection about the associated semantic of weaving relationships remains as a current challenge. Until now, the traditional ones have been defined but those introduced by Rashid et al [20] can suggests new alternatives for our approach. In this sense, we think the equilibrium between the readability/simplicity of the specification and the versatility of the weaving relation should be achieved.

Another key topic is related to identification of interaction patterns for several crosscutting concerns in order to manage the possible interference among them. In the developed case studies, we observed that several concerns crosscut another one. The associated semantic of this composition and how the tradeoffs between them have to be faced must be solved in the next future.

7. ACKNOWLEDGMENTS

This work has been funded by the Spanish CICYT project DYNAMICA TIC2003-07776-C02-02.

8. REFERENCES

[1] Aldawud, O. Elrad, T. and Bader, A. UML profile for Aspect Oriented Software Development. In the *Aspect-Oriented UML Workshop*, Collocated to the Aspect Oriented Software Development Conference, (Boston, USA March 18, 2003).

[2] Anton, A. I. Goal-Based Requirements Analysis. In Proceedings of the *2nd International Conference on Requirements Engineering*, (Colorado Springs, CO, April 15 - 18, 1996).

[3] Aspect Oriented Software Development, http://www.aosd.net

[4] Araújo, J., Moreira, A., Brito, I. and Rashid, A. Aspect-Oriented Requirements with UML, In the *Aspect-Oriented UML Workshop*. Collocated to the 5fth International Conference on the UML, (Dresden, Germany, October 4, 2002).

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

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

[7] Dardenne, A., van Lamsweerde, A. and Fickas, S. Goal-directed Requirements Acquisition. In *Science of Computer Programming*, 20, (1993), 3-50.

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

[9] Heaven, W. and Finkelstein, A. A UML Profile to Support Requirements Engineering with KAOS, *IEE Proceedings - Software*, 151, 1 (Feb. 2004), 10- 27.

[10] IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications, In Volume 4: Resource and Technique Standards, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection.

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

[12] Lamsweerde, A. van. From System Goals to Software Architecture, In *Formal Methods for Software Architecture*, LNCS 2804, Springer-Verlag, (2003), 25-43.

[13] Lamsweerde, A. van. Goal-Oriented Requirements Engineering: A Guided Tour. In Proceedings of *5th IEEE International Symposium Requirements Engineering*, (Toronto, Canada, August 27-31, 2001), 249-263

[14] Meyer, J.J-Ch. Approach to Deontic logic: Deontic Logic viewed as Variant of Dynamic Logic. In *Notre Dame Journal of Formal Logic*, 29 (1988), 109-136.

[15] Moreira, A., Araújo, J., Brito, I.. A Requirements Model for Quality Attributes. In the *Workshop on "Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design"*. Collocated to the 1st International Conference on Aspect-Oriented Software Development, (Twente, Enschede, Holland, April 22-26, 2002).

[16] Navarro, E., Ramos, I. and Pérez, J. Goals Model-Driving Software Architecture, In Proceedings of the *2nd International Conference on Software Engineering Research, Management and Applications*, (Los Angeles, CA, USA, May 5-8, 2004), 205-212.

[17] Navarro, E., Ramos, I. and Pérez, J. Software Requirements for Architectured Systems. In Proceedings of the *11th IEEE International Conference Requirements Engineering,* (Monterey, CA, September 8-12, 2003), 365-366.

[18] Navarro, E. and Ramos, I. Requirements and Architecture: a marriage for Quality Assurance. In *Proceedings of the 8º Jornadas de Ingeniería del Software y Bases de Datos*. (Alicante, Spain, November 12-14, 2003), 69-78.

[19] Pérez, J., Ramos, I., Jaén, J., Letelier, P., Navarro. E. PRISMA: Towards Quality, Aspect Oriented and Dynamic Software Architectures. In *Proceedings of the 3rd IEEE International Conference on Quality Software*, (Dallas, Texas, USA, November 6 - 7, 2003), 59-66.

[20] Rashid, A., Moreira, A., Araújo, J. Modularisation and composition of aspectual requirements. In Proceedings of the *2nd International Conference on Aspect-Oriented Software Development*, (Boston, Massachusetts, USA, March 17 - 21, 2003), 11-20.

[21] Suzuki, J. and Yamamoto, Y. Extending UML with Aspects: Aspect Support in the Design Phase. In the *Aspect Oriented Programming Workshop*. Collocated to the 13th European Conference on Object Oriented Programming (ECOOP'99) Springer LNCS 1743, (Lisbon, Portugal, June, 1999).

[22] UML 2.0: Superstructure Specification, OMG Adopted Specification. Ptc/ 03 August 2003.