Using RELAX, SysML and KAOS for Adaptive Systems Requirements Modeling

Manzoor Ahmad and Jean-Michel Bruel
University of Toulouse
CNRS/IRIT
F-31062 Toulouse Université Cedex, France
Email: manzoorcapk@yahoo.com,bruel@irit.fr

Régine Laleau and Christophe Gnaho
University of Paris-Est Créteil
LACL
94010 Créteil Cedex, France
Email: laleau@u-pec.fr,gnaho@u-pec.fr

***Abstract*—Dynamic Adaptive Systems (DAS) modify their behavior at run-time in response to changing environmental conditions. For these systems, non-functional requirements play an important role, and one has to identify requirements that are adaptable. Goal based approaches can help in the development of requirements for DAS, keeping in view the inherent uncertainty in these systems. RELAX, which is a requirement engineering language, can introduce flexibility in non-functional requirements to adapt to any changing environmental conditions. This paper shows how to model adaptive systems requirements through an existing goal oriented approach,based on KAOS, that extends the SYSML meta-model and our proposed domain specific language for RELAX; that enables to derive requirements in graphical format from textual requirements in the form of SYSML requirements diagrams. Moreover, a merge of the two approaches will serve the development of DAS.**

***Keywords*-Domain Specific Language; RELAX; Integrated Development Environment; Dynamic Adaptive Systems; Requirements Engineering;**

I. INTRODUCTION

Most of the work in Requirements Engineering (RE) for Dynamic Adaptive Systems (DAS) assumes that requirements already exists and the main focus is on requirements monitoring and reasoning about the correctness of adaptations [1]. The environmental conditions for these systems tend to change so they have to adapt to these changing conditions. In other words, we can say that DAS tend to be cyber-physical systems where the physical environment is tightly intertwined with the computing based system.

Goals can be used to systematically model the requirements of a DAS. They are treated as a collection of target systems with varying environmental conditions, so each target system's requirements are modeled, and the adaptive logic that serves for transition between configurations are treated as separate concerns [2]. To develop goal models, a process can be used called LOREM (Levels Of Requirement Engineering for Modeling) [3], to represent the individual target system and the adaptive logic. Goal oriented techniques have been proved very useful for requirements engineering in general [4] [5].

Our previous work [6] with requirements serve as a baseline for using RELAX [7] to model non-functional requirements. RELAX is a textual programming language which deals with uncertainty in DAS requirements that allows requirements to be temporarily relaxed to adapt to changing environmental conditions. This relaxation is offered in case non-critical requirements have to be partially neglected in order to satisfy short-term critical requirements. The distributed nature of DAS and changing environmental factors makes it difficult to anticipate all the explicit states in which the system will be during its lifetime. As such a DAS needs to be able to tolerate a range of environmental conditions and contexts, but the exact nature of these contexts remains imperfectly understood. One overarching challenge in developing DAS, therefore, is how to handle uncertainty posed by the respective application domains.

The context of our work is situated in Autonomic Computing where we are working on a case study of an AAL (Ambient Assisted Living) house [7]. The case study highlights the need to ensure patient's health in the AAL house. This house is equiped with an intelligent fridge that communicates with the AAL and is capable of reading, storing RFID information on food items. It also comes with 4 temperature and 2 humidity sensors. The fridge also detects the presence of spoiled food and discovers and receives a diet plan to be monitored based on what food items the patient is consuming.

The objective is therefore to model DAS requirements through an existing goal oriented approach that extends the SYSML meta-model and our proposed DSL for RELAX. The rest of the paper is organized as follows: section II shows the motivations of our work, background of the concepts, our previous experience, why using goals for defining DAS requirements and their benefits; section III shows some challenges we are facing, the integration of goals in defining DAS requirements, and the relationship between different concepts and section IV concludes the paper and gives an insight about future work.

II. MOTIVATIONS AND CONTEXT

A. Motivations

Our primary motivation is to enforce three aspects of the DAS requirements engineering:

R: Mary *SHALL* maintain liquid intake *AS CLOSE AS POSSIBLE To* ideal.
ENV: Fluid intake that is necessary
MON: Sensor enabled cups, fluid monitoring cups and faucet sensors
REL: Sensor enabled cups, fluid monitoring cups and faucet sensors all interact to collaboratively determine Mary's daily liquid intake.

Figure 1. A requirement using the RELAX syntax [7]

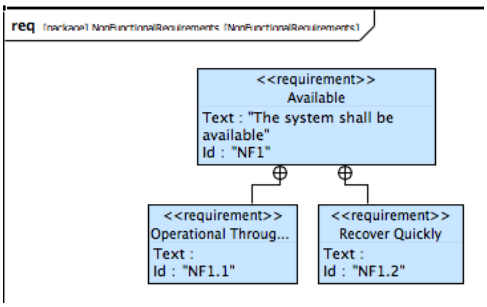

Figure 2. Generated Requirements Diagram

- To ease the identification of those requirements on which the adaptation is going to apply (this is our primary work around RELAX [7])
- To consider their traceability through the development life cycle (this is our primary work around SYSML [8])
- To integrate goal oriented concepts in defining requirements for DAS

We have focussed so far on the requirements themselves (individually), the way we can write them in a more useful and precise way (see Figure 1), and the way we can automatically inject them in a system model (see Figure 2). The next step is now to work on the way we can identify the requirements that can be relaxed (called "relaxable") or not. This is why we are currently investigating the use of goal based approaches.

B. SysML

SYSML[1] is a general purpose modeling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities. It includes a graphical construct to represent text based requirements and relate them to other model elements. The requirements diagram captures requirements hierarchies and requirements derivation, and the `<<satisfy>>` and `<<verify>>` relationships allow a modeler to relate a requirement to a model element, e.g. `<<block>>`, that satisfies or verifies the requirements. The requirement diagram provides a bridge between typical requirements management tools and system models.

C. RELAX

Typical textual requirements use modal verb SHALL that defines the functionality that a software system must always provide. RELAX takes the form of structured natural language, including operators designed specifically to capture uncertainty [9], their semantics is also defined. Uncertainty can be environmental and behavioral; environmental uncertainty is due to the changing environmental conditions such as sensor failure, noisy networks, malicious threats and unexpected human input, here uncertainty refers to maintaining the same requirements in unknown contexts whereas behavioral uncertainty refers to situations where requirements themselves need to change.

The RELAX vocabulary helps in relaxing requirements when environment changes so it enables the analysts to identify the point of flexibility in their requirements. For this purpose RELAX process [7] is used which divides requirements into two types: variant or relaxed requirements that can be relaxed when the environment changes and invariant requirements that are fixed and cannot be changed keeping in view that it is the main functionality of the system. The relaxation also implies that a trade-off be made between critical and non-critical requirements when the environment changes, or resources are constrained, in such a case, we can compromise non-critical requirements for critical requirements.

In RELAX the conventional modal verb SHALL is retained and RELAX operators are introduced to provide more flexibility in how and when that functionality may be delivered. More specifically, for requirements that are left partially unsatisfied, the introduction of an alternative, temporal or ordinal RELAX-ation modifier will define the requirement as RELAX-able. These operators define constraints on how a requirement can be relaxed at run-time. In addition, it is important to indicate what uncertainty factors warrant a relaxation of these requirements, thereby requiring adaptive behavior. This information is specified using the MON (monitor), ENV (environment), REL (relationship) and DEP (dependency) keywords.

SYSML incorporates requirements through requirements diagram so a link between SYSML and RELAX would help in modeling the requirements efficiently. SYSML provides a development environment and a graphical support for expressing all the variables of RELAX and also it helps in bridging the gap between requirements and the overall system model as it is becoming an industry standard. This is why we have developed a DSL for RELAX.

[1] http://www.sysml.org/specs/

D. DSL for RELAX

Our previous work with RELAX is centered on a Domain Specific Language for self adaptive systems [6]. RELAX grammar is used as a meta-model for our DSL and based on this meta-model we would be able to bridge the gap between requirements and the overall system model.

Using our DSL, non-functional requirements in textual format are transformed into graphical format with the help of RELAX grammar in the form of requirements diagram. Figure 2 shows a snapshot of the generated requirements diagram.

For the generation of DSL, XText[2] is used. XText is a development framework for the development of DSL and other textual programming languages and helps in the development of an Integrated Development Environment (IDE) for the DSL. Some of the IDE features that are either derived from the grammar or easily implementable are: syntax coloring, outline view, model navigation, code completion and code templates. Among the benefits of XText, we have benefited from the code generation framework that is automatically generated from the grammar. A code generator has been written that is capable of processing models created with the DSL editor [8].

As a more perspective contribution a tool, COOL RELAX Editor [10] is developed using RELAX grammar as meta-model. It is developed with all the RELAX variables embedded in it and with the help of this tool we can do M2T (Model 2 Text) and M2M (Model 2 Model) transformation.

E. Why Using Goal Oriented Techniques for Requirements Engineering

The most important requirements engineering approaches of the recent years are goal oriented. The main reason is the inadequacy of traditional approaches when dealing with more and more complex systems. These traditional approaches focus on the specification of the system-to-be alone and do not consider its environment. Moreover, they do not provide support for reasoning about alternative system configurations where different solutions can be explored and compared. Goal Oriented Requirements Engineering (GORE) approaches try to solve these issues. They take into account stakeholders' intentions and make use of goal models for specifying these intentions [4]. Thus, there are a number of positive effects if GORE approaches are used, among them:

- Goals provide the rationale for defining the requirements and developing the system-to-be. They facilitate the understanding of the objectives of the system.
- Goals drive and guide the identification of requirements, they can be organized into hierarchies. This feature provides a way to define relationships between goals, from high-level strategic objectives to low-level requirements.
- Goal requirements model support the detection and resolution of conflicts among requirements.
- A goal based requirements model can capture variability in the problem domain through the use of alternative goal refinements and alternative assignments of responsibility. Goals are well suited to evaluate alternative configurations and choose the preferred one.

[2]http://www.eclipse.org/Xtext/documentation

F. SysML/KAOS

The SYSML/KAOS model [11] is an extension of the SYSML requirements model with concepts of the KAOS goal model [12]. Several models exist to represent goal oriented requirements such as i* [13], GBRAM [14]. The choice of KAOS [15] is motivated by the following reasons. Firstly, it permits the expression of several models (goal, agent, object, behavioural models) and relationships between them. Secondly, KAOS provides a powerful and extensive set of concepts to specify goal models. This allows the design of goal hierarchies with a high level of expressiveness that can be considered at different levels of abstraction.

Indeed, as SYSML is an extension of UML, it provides concepts to represent requirements and to relate them to model elements, allowing the definition of traceability links between requirements and system models. However the set of SYSML concepts for requirements modeling is not as extensive as in goal models.

The SYSML/KAOS model allows both functional requirements and non-functional requirements [16] to be modeled. This paper focuses on the concepts related to non-functional requirements. For functional requirements concepts, see [17].

Figure 3 shows non-functional concepts as yellow boxes, the gray boxes represent the SYSML concepts. The instantiation of the meta-model allows us to obtain a hierarchy of non-functional requirements in the form of goals. Non-Functional Goals (NFG) are organised in refinement hierarchies. An NFG is either an abstract NFG or an elementary NFG. A goal that cannot be further refined is an elementary goal. The refinement of an abstract goal by either abstract or elementary goals is represented by the association class Refinement. An abstract NFG may contain several combinations of sub-goals (abstract or elementary). The relationship Refinement becomes an association class between an abstract NFG and its sub-goals. It can be specialised to represent And/Or goal refinements. In AAL case study, the goal Security [fridge input data] is an abstract NFG that can be AND-refined into three sub-goals: Confidentiality [fridge input data], Integrity [fridge input data] and Availability [fridge input data]. Similarly, the sub-goal Availability [fridge input data] can be refined into two sub-goals: Availability [Storing RFID information] and Availability [Sensors data].

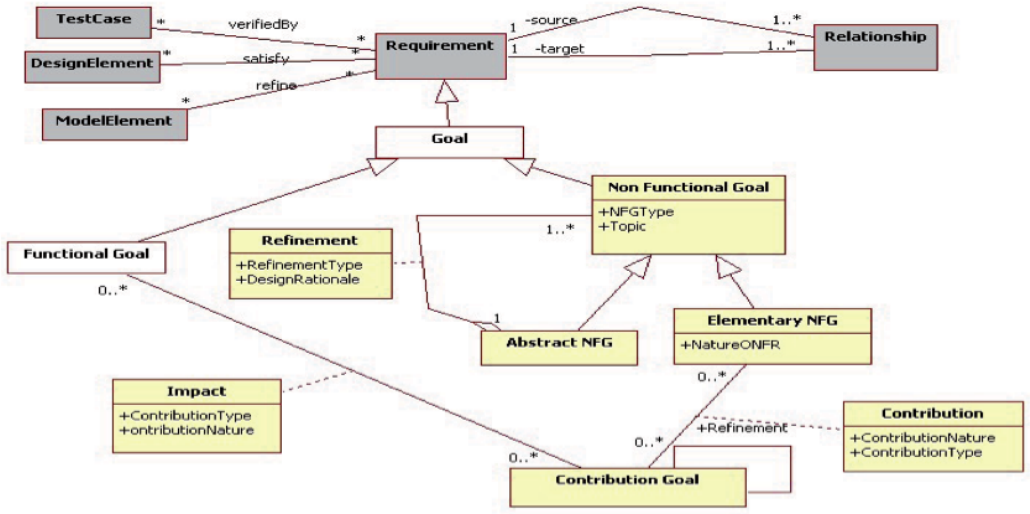

Figure 3. Extended SysML Meta-Model [11]

At the end of the refinement process, it is necessary to identify and express the various alternative ways to satisfy the elementary goals. For that, we consider the concept of contribution goal (Meta-Class Contribution Goal). A contribution goal captures a possible way to satisfy an elementary goal. The association class Contribution describes the characteristics of the contribution. It provides two properties: ContributionNature and ContributionType. The first one specifies whether the contribution is positive or negative, whereas the second one specifies whether the contribution is direct or indirect. A positive (or negative) contribution helps positively (or negatively) to the satisfaction of an elementary goal. A direct contribution describes an explicit contribution to the elementary non-functional goal. An indirect contribution describes a kind of contribution that is a direct contribution to a given goal but induces an unexpected contribution to another goal. Consider for example the elementary goal Confidentiality [fridge input data], a possible solution to meet this goal is to use a code 'PIN'; another solution is to require an additional identifier. These two solutions represent thus direct and positive contribution to this goal. Similarly, having high-end sensors contributes directly and positively to the goal Availability [Sensors data], and may contributes indirectly and positively to Integrity [fridge input data] Finally, the concept of Impact is used to connect non-functional goals to functional goals. It captures the fact that a contribution goal has an effect on functional goals.

III. Our Contribution in Requirements Modeling

Our contribution is to merge the techniques and approaches previously described in order to obtain a detailed and strong requirements description of the system and its context. In order to illustrate our proposal, we are going to use some excerpts of an AAL case study, given below: "Mary is a widow. She is 65 years old, overweight and has high blood pressure and cholesterol levels. Following her doctor's instructions, she is considering to lose weight. The doctor has recommended a hypo caloric diet with low levels of salt. She lives by herself in an AAL house".

A. Some Propositions

Based on the fact that goal oriented techniques can help in defining requirements for DAS, we give some propositions.

Proposition 1: To merge the two approaches, we start by finding the correspondence between requirement and goal. In SysML, a `<<Block>>` satisfy `<<Requirement>>` and in KAOS, an `<<Agent>>` satisfy `<<Goal>>`, so we can say that a `<<Requirement>>` corresponds to a `<<Goal>>` and a `<<Block>>` corresponds to an `<<Agent>>`. This can be verified by the work in [13] and [18] which shows that non functional requirements can be formulated in the form of goals.

Based on our experience with the AAL case study, we can prove this correspondence. Figure 4 shows the SysML requirement diagram. Here the main requirement is `<<Mary should live a healthy life>>`; this requirement is composed of two requirements i.e. `<<Minimum liquid intake>>` which is satisfied by the block `<<Sensor enabled cups>>` and `<<Hypo caloric diet>>` which is satisfied by the block `<<Fridge display>>`. Figure 5 shows the KAOS responsibility model. The main goal is `<<Mary should live a healthy life>>` which is decomposed into two sub-goals. The first sub-goal is the `<<Minimum liquid intake>>` which is achieved by the agent `<<Sensor enabled cups>>` and the other sub-goal is `<<Hypo caloric diet>>` which is achieved by

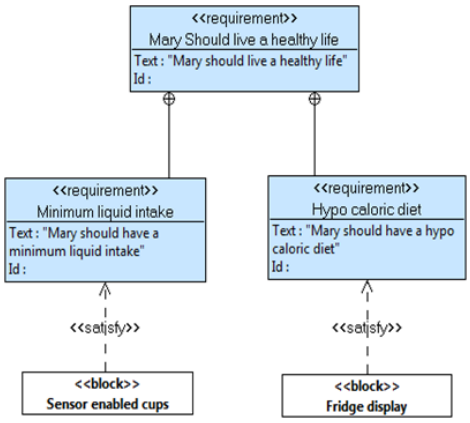

Figure 4. SysML Requirement Diagram

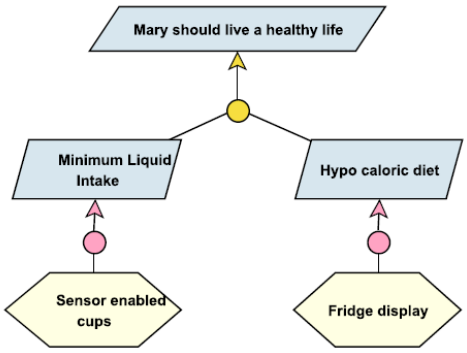

Figure 5. KAOS Responsibility Model

the agent <<Fridge display>>. By comparing the two diagrams, we can say that a <<Requirement>> corresponds to a <<Goal>> and a <<Block>> corresponds to an <<Agent>>.

Proposition 2: Uncertainty factors especially ENV and MON attributes are particularly important for documenting whether the system has means for monitoring the important aspects of environment. By collecting these ENV and MON attributes, we can build up a model of the environment in which the system will operate, as well as a model of how the system monitor its environment. Having said this, SysML/KAOS can complement RELAX by injecting more information in the form of positive/negative and direct/indirect impacts.

Proposition 3: The grammar of RELAX is acting as a meta-model for our DSL, while SysML/KAOS has extended the meta-model of SysML with goal concept. As both meta-models are close to the SysML meta-model, bridge between the two languages are going to be straightforward. We are hence confident in the fact that tooling our combined approach will not be a problem. In addition we will provide a strong consistency between the models. This can be ensured thanks to the use of formal methods that provide verification tools. We have already developed a method to derive formal B specifications from SysML/KAOS models that can be extended to consider the combined approach.

In the next section we illustrate how each important concept is handled by the approaches.

B. Relationship b/w SysML/KAOS, SysML and RELAX

In Figure 6, we have shown how several key concepts are taken into account in the selected models. Most of the time, the concepts are not fully covered (e.g. <<satisfy>> for monitoring in SysML, this stereotype is used between a block and a requirement), but we have indicated in the table the closest mechanism that supports the concepts. In SysML/KAOS, requirements are described in the form of goals, SysML describes requirements in textual form while RELAX requirements are also in textual form with an enhanced version i.e. requirements divided into invariant and RELAX-ed requirements with uncertainty factors added to it. SysML/KAOS has AND/OR refinement relationships, SysML has <<verify>> and <<refine>> relationships while for RELAX, we have REL variable which identifies the relationship between ENV and MON. For Dependency/Impact, SysML/KAOS describes it as an impact of non-functional goal on functional goal; this impact can be positive or negative and direct or indirect while for SysML, we have the concept of <<derive>> which shows the dependency between requirements, RELAX has positive and negative dependency. To deal with monitoring, SysML/KAOS has the <<contribution goal>> concept which is used to satisfy a non-functional goal, SysML has <<satisfy>> which is used when a <<block>> satisfies a <<requirement>> while for RELAX, we have the concept of MON which is used to measure the environment i.e. ENV. SysML/KAOS has a tool called SysML/KAOS editor, SysML has a number of tools e.g. eclipse[3], papyrus[4], topcased[5] etc and for RELAX we have eclipse based COOL RELAX editor [10].

IV. Conclusion and Future Work

Dynamic adaptive systems modify their behavior at runtime in response to changing environmental conditions. For these systems, non-functional requirements play an important role, and one has to identify requirements that are concerned with the adaptive features. To develop requirements for DAS, goal based approaches play an important role keeping in view the inherent uncertainty in these systems. RELAX which is an RE language for self-adaptive systems can introduce flexibility in non-functional requirements; to adapt to any changing environmental conditions. This paper is based on requirements modeling using two different

[3]http://www.eclipse.org/
[4]http://www.papyrusuml.org
[5]http://www.topcased.org/

Concepts	SysML/KAOS	SysML	RELAX
Requirements Description	Goals	Textual Requirements	Enhanced Version of Textual Requirements
Relationship	AND, OR	<<verify>> <<refine>>	REL
Dependency/Impact	Contribution Nature: Positive Negative Contribution Type: Direct (Explicit) Indirect (Implicit) b/w NFG and FG	<<derive>> <<contain>>	DEP: Positive Negative
Monitoring	<<contribution goal>>	<<satisfy>>	MON
Tools	Eclipse based SysML/KAOS Editor	Eclipse/Papyrus/Topcased/	Eclipse based COOL RELAX editor

Figure 6. Relationship b/w Different Concepts

approaches: one through the use of existing goal oriented approach, for which SysML meta-model is extended with goal concepts and the other proposed approach uses RELAX which takes requirements in textual format and transform it into graphical format using RELAX grammar that acts as a meta-model.

We believe that SysML/KAOS can help RELAX to inject additional useful information (e.g., in the form of positive/negative and direct/indirect impacts). Both approaches treat requirements at very early stages. Our work on RELAX is part of an integrated work plan [19]. This work is a baseline for more concrete work aiming at exploring the full merging of the two approaches. In itself, integrating many modeling languages (KAOS, SysML and RELAX) is probably a good idea, as each of these languages brings its own analysis power and has its own benefits. Another important aspect of RELAX is that the ENV, MON and REL attributes will be particularly interesting in building the SysML parametric diagrams so we can for example use mathematical equations to implement these attributes in the parametric diagram. Future work is focussed on the development of the AAL case study. We are also interested in using formal methods [17] to prove some of the properties of the system before the development even starts. It is particularly important as the development of an AAL house involves different technologies ranging from medical services to surveillance cameras to intelligent devices.

REFERENCES

[1] B. H. C. Cheng, P. Sawyer, N. Bencomo, J. Whittle. A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty, MODELS '09, Springer-Verlag Berlin, Heidelberg 2009.

[2] J. Zhang, B. H. C. Cheng, Model-based development of dynamically adaptive software, ICSE 06: Proceedings of the 28th international conference on Software engineering, New York USA, ACM (2006) 371 - 380.

[3] H. Goldsby, P. Sawyer, N. Bencomo, D. Hughes, B. H. C. Cheng, Goal-based modeling of dynamically adaptive system requirements, 15th Annual IEEE Int. Conf. on the Engineering of Computer Based Systems (ECBS) 2008.

[4] Axel Van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", Fifth IEEE International Symposium on Requirements Engineering (RE'01) Toronto, 249-263.

[5] Shahzad Anwer, Naveed Ikram, "Goal Oriented Requirement Engineering: A Critical Study of Techniques," 13th Asia Pacific Software Engineering Conference (APSEC'06), pp.121-130.

[6] M. Ahmad, First Step towards a Domain Specific Language for Self Adaptive Systems, NOTERE, May 31- June 2 2010 Tozeur Tunisia, P. 285 - 290 ISBN: 978-1-4244-7067-9 DOI :10.1109/NOTERE.2010.5536629.

[7] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, J. M. Bruel, RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive systems, Proceedings of the 2009, 17th IEEE International Requirements Engineering Conference, Pages: 79-88.

[8] M. Ahmad, J. M. Bruel, Requirement Language Tooling with Xtext, UBIMOB 2011 7es Journées francophones Mobilité et Ubiquité Toulouse, Museum d'Histoire Naturelle 2011.

[9] J. Whittle, P. Sawyer, N. Bencomo, and B. H. C. Cheng. Reassessing languages for requirements engineering of self-adaptive systems, In RE Workshop for SOCCER08, Technical Report, 2008.

[10] S. Gatti, J. Geisel, S. Labondance, J. Pages, COOL RELAX Editor, Internal Report M2ICE, U. Toulouse II Le Mirail 2011.

[11] Y. Belkaid, C. Gnaho, F. Semmak, Création de l'éditeur KAOS-SysML sous Topcased, Rapport Tacos, Juillet 2010.

[12] A KAOS Tutorial, V1.0 Respect-IT, 2007

[13] Chung et al., Non-functional Requirements in Software Engineering. Kluwer, 1999.

[14] Anton AI, Goal based requirements analysis. In Proceedings of Int. Conf. on Requirements Engineering, 1996.

[15] A. V. Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, 2009.

[16] Christophe Gnaho, Farida Semmak. Une extension SYSML pour l'ingénierie des exigences non-fonctionnelles orientée but. Revue Ingénierie des Systèmes d'Information, Vol 16/1, 23 pages, 2011.

[17] Régine LALEAU, Farida SEMMAK, Abderrahman MATOUSSI, Dorian PETIT, Ahmed HAMMAD, Bruno TATIBOUET: "A first attempt to combine SysML requirements diagrams and B." Innovations in Systems and Software Engineering, Springer, Springer, 6(1-2): 47-54, 2010.

[18] Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos (festschrift), LNCS volume 5600. Springer, 2009. 530 pp. ISBN 978-3-642-02462-7.

[19] J. M. Bruel, N. Belloir, M. Ahmad, SPAS: un profil SysML pour les systèmes auto-adaptatifs, 15ème Colloque National de la Recherche en IUT (CNRIUT), Lille, 2009.