Using RELAX, SysML and KAOS for Ambient Systems Requirements Modeling

Manzoor Ahmad, Jean-Michel Bruel

University of Toulouse
CNRS/IRIT
F-31062 Toulouse Université Cedex, France
manzoor.ahmad.bruel@irit.fr

Régine Laleau, Christophe Gnaho

University of Paris-Est Créteil
LACL
94010 Créteil Cedex, France
laleau.gnaho@u-pec.fr

Abstract

Ambient Systems are highly adaptive. They modify their behavior at run-time in response to changing environmental conditions. For these systems, Non Functional Requirements (NFR’s) play an important role, and one has to identify as early as possible the requirements that are adaptable. Because of the inherent uncertainty in these systems, goal based approaches can help in the development of their requirements. RELAX, which is a Requirement Engineering (RE) language for adaptive systems, can introduce flexibility in NFR’s to adapt to any changing environmental conditions. We illustrate our proposal through a case study of an Ambient Assisted Living (AAL) system. We use an existing goal oriented approach, based on KAOS, which extends the SysML 1 meta-model and our proposed Domain Specific Language (DSL) for RELAX; that enables to derive requirements in graphical format from textual requirements in the form of SysML requirements diagrams. In this paper we show how we have integrated these two approaches for a better modeling of these systems.

1. Introduction

Among ambient systems, we consider more specifically in this paper those which are Dynamic Adaptive Systems (DAS). Most of the work in RE for DAS assumes that requirements already exists and the

1http://www.sysml.org/specs/
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. There is much and growing interest in software systems that can adapt to changes in their environment or their requirements in order to continue to fulfill their mandate.

The most important requirements engineering approaches of 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. However, consideration of the environment is critical for DAS. Moreover, DAS is treated as a collection of target systems with varying environmental conditions, so each target systems requirements are modeled, and the adaptive logic that serves for transition between configurations are treated as separate concerns [2]. Therefore, we need an approach that can 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 [3]. Consequently, Goals can be used to systematically model the requirements of a DAS.

There are a number of claims of advantages made from GORE. Goals enable the sufficient completeness and pertinence of a requirements specification. Goals provide the rationale for requirements, any requirement which does not contribute to any goal will not be considered at all. Goal graphs provide traceability link, like from low-level requirements to high level objectives and from organizational to business context. Contributions among goals (positive or negative) can be modeled and managed, in this way conflicts can be identified and resolved. If the current monitoring infrastructure of the DAS does not provide enough information about the environment, then either the monitoring infrastructure should be extended or if resources are constrained, the adaptive capability must be reduced. This kind of trade-off analysis can be effectively modeled using GORE, in particular obstacle analysis in Kaos provides a variant of threat modeling (which is the basis of assessing the monitoring infrastructure).

Our previous work [4] serves as a baseline for using RELAX [5] to model NFR’s. RELAX is an RE language which deals with uncertainty in DAS and 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 other 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. Therefore, we consider each NFR as RELAX-able or variant requirement so that we can introduce the concept of goals in our approach.

Similar work can be found in literature for adaptive systems; starting from defining requirements to operationlisation. Awareness Requirements [6] is one of them; which are characterized syntactically as requirements that refer to other requirements or domain assumptions and their success or failure at runtime. AwReqs are represented in a formal language and can be directly monitored by a requirements monitoring framework. In [7], the authors present FLAGS (Fuzzy Live Adaptation Goals for Self-Adaptive Systems), an innovative goal model that generalizes the Kaos model, adds adaptive goals to embed adaptation countermeasures, and fosters self-adaptation by considering requirements as live, runtime entities. FLAGS also distinguishes between crisp goals, whose satisfaction is boolean, and fuzzy goals, whose satisfaction is represented through fuzzy constraints.

The context of our work is situated in Ambient Systems where we are working on a case study of an Ambient Assisted Living (AAL) house [5]. The case study highlights the need to ensure patient’s health in the AAL house. The objective is therefore to model requirements of AAL through the merging of an existing goal oriented approach that extends the SysML meta-model and find a link with our proposed DSL for RELAX. The rest of the paper is organized as follows: section 2 shows the motivations of our work, background of the concepts and our previous experience, section 3 shows the integration of goals in defining DAS requirements with the help of a case study and the relationship between different concepts section 4 shows the follow up of the case study and the lessons learned, and section 5 concludes the paper and gives an insight about future work.
2. Motivations and context

2.1. Motivations

Our primary motivation is to enforce three aspects of the DAS requirements engineering: To ease the identification of those requirements on which the adaptation is going to apply (this is our primary work around RELAX [4]). 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 (this is the main topic of the paper).

We have focused 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. The next step is now to work on the way we can identify those requirements that can be relaxed (called "RELAX-able") and to integrate goal oriented concepts in modeling these requirements. This is why we are currently investigating the use of goal based approaches.

2.2. SysML

SysML 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.

2.3. DSL for RELAX

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 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. 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 [5] 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 since they represent the main functionality of the system. 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

![Fig. 1. A requirement using RELAX syntax](image-url)
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 helps in bridging the gap between requirements and the overall system model.

Our previous work with RELAX is centered on a Domain Specific Language for self adaptive systems [4]. The RELAX grammar is used as meta-model for our DSL and based on this meta-model we are able to bridge the gap between requirements and the overall system model. Using our DSL, NFR’s in textual format are transformed into graphical format with the help of RELAX grammar in the form of requirements diagram. For the generation of DSL, XText2 is used. XText is a framework for the development of DSL and other textual programming languages. It helps in the development of an Integrated Development Environment (IDE) for the DSL. 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 concrete contribution a tool; COOL RELAX Editor [10] is developed using RELAX grammar as meta-model. The limitations of our DSL for RELAX is overcome by the concepts of SysML/Kaos [11]. In SysML/Kaos, NFR’s are expressed in the form of goals which is much more rich and complete in defining relations between requirements (refinement relations, conflict identification and resolution, positive/negative and direct/indirect impacts). Here, invariant requirements are captured by the concept of Functional Goal whereas relaxed requirements are captured by the concept of Non Functional Goal (NFG).

2.4. SysML/Kaos

The SysML/Kaos model 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], Goal-Based Requirements Analysis Method (GBRAM) [14]. The choice of Kaos [15] is motivated by the following reasons. Firstly, it permits the expression of several models (goal, agent, object, behavioral models) and relationships between them. Secondly, Kaos provides a powerful and extensive set of concepts to specify goal models. This facilitates the design of goal hierarchies with a high level of expressiveness that can be considered at different levels of abstraction. 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 NFR’s [16] to be modeled. This paper focuses on concepts related to NFR’s. For functional requirements concepts, see [17].

2http://www.eclipse.org/Xtext/documentation
Figure 2 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 NFR’s 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. 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 NFG. 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. 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.

3. Modeling the Requirements of AAL Case Study

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: “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”. We start by identifying two high level goals: functional goal Mary should live a healthy life and NFG Reliability [AAL System]. These two goals must be refined progressively using goal models to obtain final requirements of the system. In this paper we limit ourselves to the refinement of NFG Reliability [AAL System].

3.1. High Level Goal Model

From the AAL System problem statement, we have identified the following non functional high level goal: Reliability[AAL system]. In fact, one of the expected qualities of the system is to run reliably. This is very important for several reasons and particularly because frequent visit of technician could be a factor of disturbance for Mary and unfeasible due to the large number of AAL houses across the world. An NFG can be written in the form of: NFGType [Topic] where the attribute NFGType specify the type of NFR and the attribut Topic represent the domain element effected by this type of requirement. The refinement of an NFG can be either refinement by type (NFGType) or refinement by subject (Topic). The high level goal Reliability [AAL System] is AND-refined into four sub goals using refinement by type: Precision [AAL System], Security [AAL System], Robustness [AAL System] and Performance [AAL System]. Each sub goal can be further refined until the refinement stops. The sub goal Precision [AAL System] is AND-refined into two sub goals: Precision [Location Detection] and Precision [Sensors] using refinement by subject. The sub goal Precision [Sensors] is then AND-refined into three sub goals using refinement by subject: Precision [Location Sensors], Precision [Medical Data Sensors] and Precision [Fridge Sensors]. The sub goal Precision [Location Detection] can be satisfied by a positive and direct contribution by one of the following contribution goals: combine data from multiple sensors, combine multiple features and use redundant features. The contribution goal, combine data from multiple sensors, contribute indirectly and negatively to the satisfaction of sub goal Performance [AAL System]. Figure 3 shows the high level goal model of AAL.
3.2. Security Goal Model

In AAL case study, the goal Security [fridge input data] is an abstract NFG that can be AND-refined into three sub-goals which is refinement by type: 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 using refinement by subject: Availability [Storing RFID information] and Availability [Sensors data]. 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]. Figure 4 shows the security goal model of AAL.

4. Lessons Learned from the Case Study

4.1. Uncertainty Factors/Impacts

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 monitors its environment. Having said this, SysML/Kaos can complement Relax by injecting more information in the form of positive/negative and direct/indirect impacts as shown in the high level and security goal models of AAL.

4.2. Verification of Ambient System’s Properties through Formal Methods

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, bridging the two languages is 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 [17] to derive formal B specifications from SysML/Kaos models, that can be extended to consider the combined approach. It is important as the development of an AAL system involves different technologies and formal methods can provide tools to verify the consistency of a specification.

4.3. Relationship b/w SysML/Kaos, SysML and RELAX

In Figure 5, 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. eclipse³, papyrus⁴, topcased⁵ etc and for RELAX we have eclipse based COOL RELAX editor [10].

5. Conclusion and Future Work

Ambient Systems are highly adaptive. They modify their behavior at run-time in response to changing environmental conditions. We consider more specifically those which are DAS, for these systems, NFR’s play an important role, and one has to identify requirements that are concerned with the adaptive features.

---

³http://www.eclipse.org/
⁴http://www.papyrusuml.org
⁵http://www.topcased.org/
To develop requirements for DAS, goal based approaches play an important role. RELAX which is an RE language for self adaptive systems can introduce flexibility in NFR’s to adapt to any changing environmental conditions. This paper is based on requirements modeling of an AAL system using an existing goal oriented approach, based on KAoS, which 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. In this paper we show how we have integrated these two approaches for a better modeling of these systems.

We believe that SysML/KAoS can help RELAX inject additional useful information e.g. in the form of positive/negative and direct/indirect impacts, conflict resolution, the refinement relationships etc.. Both approaches treat requirements at very early stages. Our work on RELAX is part of an integrated work plan [18]. 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 (SysML/Kaos 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 focused on the development of the AAL case study. We are also interested in using formal methods 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