A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty *

Betty H.C. Cheng[1], Pete Sawyer[2], Nelly Bencomo[2], Jon Whittle[2]
chengb@cse.msu.edu; {sawyer,nelly,whittle}@comp.lancs.ac.uk

[1] Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan 48824, USA
[2] Computing Department, InfoLab21, Lancaster University, LA1 4WA, United Kingdom

Abstract. Dynamically adaptive systems (DASs) are intended to monitor the execution environment and then dynamically adapt their behavior in response to changing environmental conditions. The uncertainty of the execution environment is a major motivation for dynamic adaptation; it is impossible to know at development time all of the possible combinations of environmental conditions that will be encountered. To date, the work performed in requirements engineering for a DAS includes requirements monitoring and reasoning about the correctness of adaptations, where the DAS requirements are assumed to exist. This paper introduces a goal-based modeling approach to develop the requirements for a DAS, while explicitly factoring uncertainty into the process and resulting requirements. We introduce a variation of threat modeling to identify sources of uncertainty and demonstrate how the RELAX specification language can be used to specify more flexible requirements within a goal model to handle the uncertainty.

1 Introduction

Dynamically adaptive systems (DASs) are systems designed to continuously monitor their environment and then adapt their behavior in response to changing environmental conditions. DASs tend to be *cyberphysical systems*, where the physical environment is tightly intertwined with the computing-based system. Example domains where DASs are necessary include power grid management systems, telecommunication systems, and ubiquitous systems. For these systems, the software may need to be reconfigured at run time (e.g., software uploaded or removed) in order to handle new environmental conditions.

Specifying the requirements for DASs is a challenging task because of the inherent uncertainty associated with an unknown environment. This paper

* This work has been supported in part by NSF grants CNS-0551622, CCF-0541131, IIP-0700329, CCF-0750787, Army Research Office grant W911NF-08-1-0495, Ford Motor Company, and a Quality Fund Program grant from Michigan State University.

presents an approach in which goals [1] are used to systematically model the requirements of a DAS. In particular, we use a variation of threat modeling (see, e.g., [2]) to uncover places in the model where the requirements need to be updated to support adaptation. In this case, threats correspond to changes in the environment that may require the software to dynamically adapt at run time. This process results in a goal-based requirements model that explicitly captures where adaptations are needed, documents the level of flexibility supported during adaptation, and takes into account enviromental uncertainty.

This paper builds directly on our previous work. In [3], it was noted that a DAS is a collection of *target systems*, each of which handles a combination of environmental conditions. As such, we can model the requirements of individual target systems and the adaptive logic that transitions between the configurations as separate concerns. The LOREM process [4] describes how to use this idea to develop goal models to represent the individual target systems and the adaptive logic. However, LOREM does not support requirements engineers in identifying the requirements for these target systems. In [5], we described the RELAX language, a textual language for dealing with uncertainty in DAS requirements which allows requirements to be temporarily relaxed if necessary to support adaptation. This flexibility is required, for example, if non-critical requirements must be partially neglected in order to satisfy short-term critical requirements. RELAX, however, was not integrated with modeling approaches used in the requirements engineering community.

This paper, therefore, has three main contributions. Firstly, it gives a process for identifying requirements for target DAS systems that can then be modeled using a process such as LOREM. Secondly, it integrates our previous work on RELAX with goal modeling. Finally, the paper presents a novel application of threat modeling to systematically explore environmental uncertainty factors that may impact the requirements of a DAS.

We illustrate our approach by applying it to Ambient Assisted Living (AAL), an adaptive system providing assistance to elderly or handicapped persons in their homes. The remainder of the paper is organized as follows. Section 2 introduces AAL as our running example and presents our approach, including the stepwise process for creating the goal and uncertainty models. Section 3 describes the details of applying the approach to the AAL system. Section 4 discusses related work. Finally, in Section 5, we present conclusions and discuss future work.

2 Modeling Approach

A key characteristic of a DAS is that that there may be numerous approaches to realizing its high-level objectives, where a specific set of run-time environmental conditions will dictate which particular realization is appropriate at a particular point in time. In order to support this type of variation, this paper uses goal modeling to describe requirements of a DAS, since goal-based modeling offers a means to identify and visualize different alternatives for satisfying the overall objectives of a system [1, 6]. The alternatives may be due to different

tradeoffs between non-functional goals (e.g., performance, reliability, etc.); and, in the case of DASs, different goal paths may be due to uncertainty factors in the environment. As such, goal-based modeling offers a means to explicitly capture the rationale for how and why goals and requirements are decomposed. Furthermore, requirements identified through goal modeling can be used as the basis for model-driven engineering (MDE) [1, 7, 3]. The additional information captured by goal models (i.e., the rationale for a particular path of goal refinement) may be used as constraints and/or guidance during the MDE process [3].

2.1 Running Application

To validate our approach, we conducted a case study provided by Fraunhofer IESE in the form of an existing concept document describing a smart home for assisted living. The concept document was written previously and independently of this research. We present an excerpt of the document here to serve as a running example for introducing our approach. [1]

Mary is a widow. She is 65 years old, overweight and has high blood pressure and cholesterol levels. Mary gets a new intelligent fridge. It comes with 4 temperature and 2 humidity sensors and is able to read, store, and communicate RFID information on food packages. The fridge communicates with the ambient assisted living (AAL) system in the house and integrates itself. In particular, it detects the presence of spoiled food and discovers and receives a diet plan to be monitored based on what food items Mary is consuming.

An important part of Mary's diet is to ensure minimum liquid intake. The intelligent fridge partially contributes to it. To improve the accuracy, special sensor-enabled cups are used: some have sensors that beep when fluid intake is necessary and have a level to monitor the fluid consumed; others additionally have a gyro detecting spillage. They seamlessly coordinate in order to estimate the amount of liquid taken: the latter informs the former about spillages so that it can update the water level. However, Mary sometimes uses the cup to water flowers. Sensors in the faucets and in the toilet also provide a means to monitor this measurement.

Advanced smart homes, such as Mary's AAL, rely on adaptivity to work properly. For example, the sensor-enabled cups may fail, but since maintaining a minimum of liquid intake is a life-critical feature, the AAL should be able to respond by achieving this requirement in some other way.

2.2 Overview of Approach

Our approach follows the principles of the model-based approach described by Zhang and Cheng [3] which considers a DAS to comprise numerous target systems, each of which supports behavior for a different set of environmental conditions (posed by the environmental uncertainty). At run time, the DAS transitions from one target system to another, depending on the environmental conditions.

[1] See www.iese.fraunhofer.de/fhg/iese/projects/med_projects/aal-lab/index.jsp.

While the earlier work emphasized design-phase models, this paper focuses on the identification of the goals and requirements for each of the target systems.
Scope of Uncertainty. Before we start the goal derivation process, we identify the top-level goal for the system; this goal should state the overall objective for a system, while not being prescriptive for how to realize the objective. And we also create a conceptual domain model (as a UML class diagram) that identifies the key physical elements of the system and their relationships (e.g., sensors, user interfaces). These elements contribute to the environmental conditions and the uncertainty that must be handled by the system. In essence, the domain model serves to scope the uncertainty for the system; that is, elements in the domain model are either the sources of uncertainty or they are used to monitor environment conditions that pose uncertainty. Figure 1 gives the conceptual domain model, identifying the physical elements, their relationships to each other and to the AAL. It also includes actors that may be human (e.g. Person) or software-controlled (e.g. iCup, an intelligent cup with sensors).

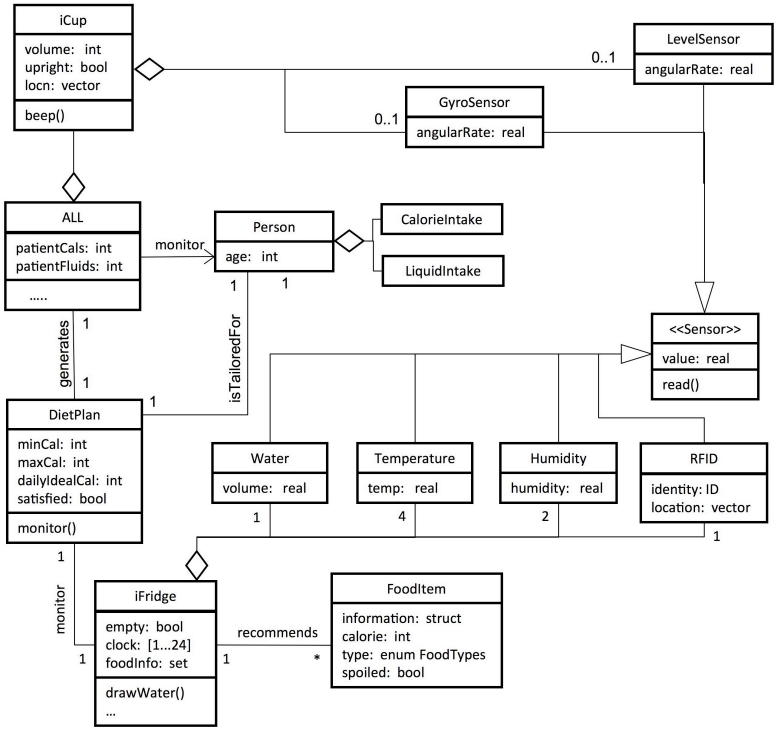

Fig. 1. Conceptual domain model

Target System Modeling. From the top-level goal, we develop a goal lattice using a process of goal refinement, where the first level of subgoals are termed

high-level goals, representing the key services to be provided by the system. This refinement process is informed by the conceptual domain model and any problem descriptions, use-cases or other sources of information elicited about the problem to be tackled by the system under development (herein referred to as system). We use KAOS, a goal-oriented requirements engineering language [1]; one influencing factor for using KAOS is its support for threat modeling. In KAOS, goals describe required properties of a system that are satisfied by different agents such as software components or humans in the system's environment. Goal refinement in KAOS stops when responsibility for a goal's satisfaction can be assigned to a single agent. KAOS defines such a goal as a *requirement* if satisfied by a software agent or an *expectation* if satisfied by a human agent. Requirements and expectations form leaves of the goal lattice. It should be noted that the KAOS definition of requirement is specific to KAOS but, for consistency sake, we shall use the KAOS convention in the remainder of this paper.

Figure 2 gives a goal model for the AAL system, where the top-level goal is to keep Mary healthy (i.e., Maintain[Health]). The right leaning parallelograms represent goals, while the left leaning parallelograms represent KAOS *obstacles* that act to confound goal satisfaction. Considering the goals first, requirements and expectations are denoted as goals with embolded outlining. The hollow circles represent goal refinement junctures, where multiple edges represent AND goals (all subgoals must be satisfied in order to satisfy a parent goal). Goals can also be OR-ed, denoted by multiple arrows directly attached to a parent goal; an example appears in Figure 5. Goals can be elaborated to provide a number of attributes including a definition. The dashed box attached to the Maintain[Health] goal shows its definition formulated as a conventional SHALL statement.[2] Finally, agents are represented by hexagons. The network of goal-related elements form a *goal lattice*.

Identifying Uncertainty. We assess the goal lattice in a bottom-up fashion, looking for sources of uncertainty (i.e., elements in the domain model) that might affect the satisfaction of the goals. Previously, threat modeling has been used to identify threats that might exploit (security) vulnerabilities of system assets [8, 9]. In this current work, we introduce a variation of threat modeling to identify uncertainty. More specifically, in the case of DASs, the "threats" are the various environmental conditions (or the impact of environmental conditions) that pose uncertainty at development time and thus may warrant dynamic adaptation at run time to ensure acceptable behavior. The obstacles in Figure 2 represent uncertainty factors impacting the goals which, like the goals, form a lattice, termed *uncertainty lattice*, in which obstacles can be AND-ed and OR-ed to combine their effects and propagate uncertainty upwards towards the top-level goal. The lower uncertainty nodes represent the sources of uncertainty. The barred arrows indicate the goals that they affect. The upper uncertainty nodes and the barred, broken arrows that lead from them represent the impact of the uncertainty.

[2] *SHALL* statements are commonly used to specify requirements, indicating a contractual relationship between the customer and the developer as to what functionality should be included in the system.

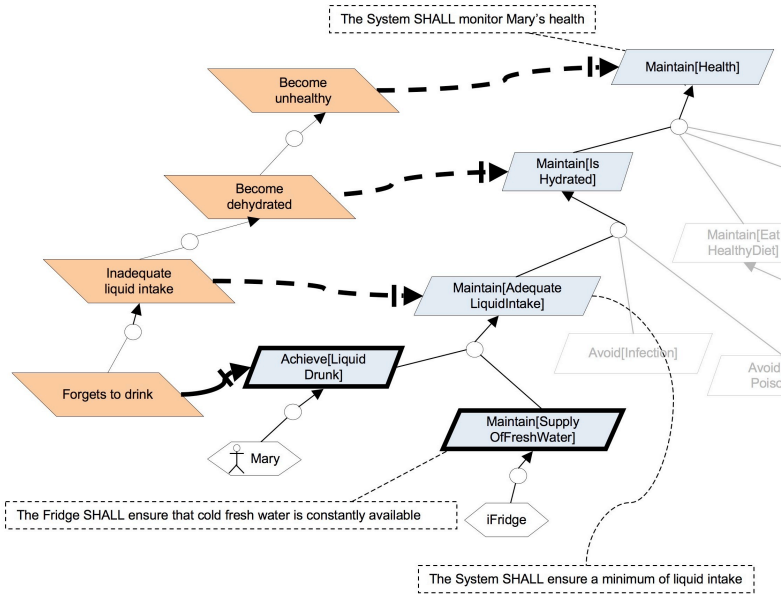

Fig. 2. Initial refinement of the goals to keep Mary hydrated

Mitigating Uncertainty. The impact of the uncertainty is assessed to determine what type of mitigation, if any, is needed. Three possible tactics can be used to mitigate the offending uncertainty factors, with each requiring different levels of effort to realize. For a goal affected by uncertainty, the least costly mitigation tactic is to define new behavior in the form of a further subgoal to handle the condition; this step equates to adding incremental functionality to a target system. If the subgoal refinement is not sufficient to mitigate the uncertainty, but partial satisfaction of the goal is tolerable, then we attempt to add flexibility to the goal to account for the uncertainty. For this tactic, we use the RELAX specification language [5] to add flexibility to the goal specification by specifying requirements declaratively, rather than by enumeration. Briefly, RELAX can be used to specify several dimensions of uncertainty, including duration and frequency of system states; possible states of a system; and configurations for a system. While the RELAX specifications are in the form of structured natural language with Boolean expressions, the semantics for RELAX have been defined in terms of temporal fuzzy logic [5]. Due to space constraints, we can only briefly overview the RELAX language here; details may be found in [5].

To illustrate the use of RELAX to mitigate uncertainty, consider the following goal that may not be satisfiable all the time.

"The System SHALL ensure that cold fresh water is constantly available."

If we fail to take into account the uncertainty surrounding water supply and design the system as if interruptions in water supply will never occur, then the system may be too brittle and fail when an interruption does occur. However,

if the recipient of the system's services can tolerate short disruptions in supply, then we might RELAX the goal using a temporal RELAX operator as follows:

> *"The System SHALL ensure that cold fresh water is AS CLOSE AS POSSIBLE to constantly available."*

The RELAXed goals can be realized by implementations that have built-in flexibility (e.g., through parameter definitions or alternate branches of functionality). Note that goals for which partial satisfaction is not tolerable are considered to be *invariants*. Invariants represent system requirements that must always be satisfied even during adaptation.

If the adverse impact of the uncertainty cannot be mitigated by formulating new subgoals or by RELAX-ation, then we have to consider the given goal as *failed*. As such, we need to create a new high-level goal that captures the objective of correcting the failure. This uncertainty-mitigation tactic is the most costly since the new high-level goal and its subsequent refinement correspond to the goal lattice for a new target system. Examples of each uncertainty-mitigation tactic are described in Section 3.

Not shown in the text or the figures above are two key non-functional requirements that guided the goal refinement process: the solutions offered by the AAL should, as far as practicable, be *non-invasive* and of *low cost*. Since the focus of this paper is on detecting and modeling uncertainty in the context of DASs, we only consider the non-functional requirements implicitly in this discussion. In the LOREM work [4], we described how to use goal modeling of non-functional requirements as the sole basis for dynamic adaptation, where the different combinations of environmental conditions were explicitly enumerated. In contrast, this paper describes a technique for identifying the environmental conditions warranting dynamic adaptation.

2.3 Process Overview

The analysis steps described above can be applied systematically using the following stepwise process: Figure 3 gives the data flow diagram for the process. Ovals represent processes, arrows represent data flows, and parallel lines represent data stores.

Step 0: Identify Top-level goal and Environment: Identify the top-level goal for system. Create a conceptual domain model that identifies the environmental elements relevant to the system; these elements are potential sources of uncertainty for the system).

Step 1: Derive the goal models: Perform goal refinement until we derive leaf requirements/expectations and their respective agents.

Step 2: Identify Uncertainty Factors: Starting from the leaf requirements/expectations identify the uncertainty factors that might prevent their satisfaction. These uncertainty factors represent environmental conditions that potentially affect the behavior of the system. The uncertainty and/or the impact of the uncertainty factors propagate up the goal lattice if not adequately mitigated.

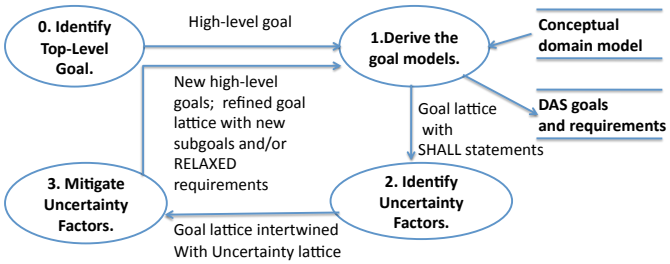

Fig. 3. Process for Goal-Based Modeling of Adaptive Systems

Step 3: Mitigate Uncertainty Factors:
Below are the mitigation tactics, presented in order of increasing cost (i.e. effort to realize).

i. No refinement: If the uncertainty factors do not prevent satisfaction of the goals then do not modify the respective goal.

ii. Add low-level subgoals: If the uncertainty can be mitigated by introducing new low-level goals, then refine with new subgoals.

iii. RELAX goals: If the uncertainty prevents high-level goals from being completely satisfied but we can accept their partial satisfaction, then RELAX the highest level goal impacted by the corresponding uncertainty.

iv. Add high-level goal: If the effect of uncertainty on a high-level goal is unacceptably severe, then identify a new (high-level) goal to mitigate the uncertainty. This new goal represents a new target system and the closer to the top-level goal it is, the greater the implied cost of implementation. Such a goal will have to be refined by executing Steps 1 - 3 for the new portion of the goal lattice.

3 Application of Goal Modeling for the AAL System

This section describes the results of applying our modeling approach to the AAL system. Due to space constraints, we can only present excerpted goal models of each of the types of uncertainty mitigation.

Step 0: Identify Top-level goal and Environment. Recall that Figure 1 gives the conceptual domain model for the AAL, which serves to scope the environment and uncertainty factors for the AAL. Step 0 of our analysis identified the top-level goal of the AAL house as keeping Mary healthy (i.e., Maintain[Health]), as shown in Figure 2. The 'Maintain' predicate of the label denotes the goal as a behavioural goal specifying a property that should always hold. The inverse of a 'Maintain' goal is an 'Avoid' goal. Hence the top-level goal could be denoted by the goal Avoid[BadHealth]. A third class of behavioural goals is denoted by an 'Achieve' predicate, indicating a property that should eventually hold.

Step 1: Derive the goal models. Figure 2 shows Step 1 of our process to refine the top-level goal as a lattice of subgoals. We elide all but one branch of the lattice to illustrate the refinement of the goals concerned with ensuring that her liquid intake is sufficient. The branch has been refined to a single expectation that Mary drinks and a single requirement that the iFridge supplies cold drinking water. These are AND-ed to indicate that both need to be satisfied in order to satisfy the goal of maintaining adequate liquid intake.

Step2: Identify Uncertainty Factors. Following identification of the goals, Step 2 analyses the extent to which they are satisfiable by developing the uncertainty model using KAOS obstacles. The key uncertainty factor in Figure 2 is represented by the obstacle Forgets to drink. It is uncertain whether Mary will drink enough liquid; she could forget to drink and the effect of this would mean that she gets too little liquid, becomes dehydrated, and ultimately, unhealthy.

Step 3(ii): Mitigate Uncertainty Factors. Completion of the uncertainty model triggers Step 3 whose purpose is to evaluate the uncertainty factors and decide whether to try to mitigate them. Assuming that the uncertainty is sufficiently serious that some mitigation is needed, we start by attempting to apply 3(ii), adding a new subgoal to mitigate the obstacle. Uncertainty about whether Mary will drink enough, which is represented by the Forgets to drink obstacle in Figure 4, has been mitigated by adding a new goal Achieve[ReminderToDrinkIssued], highlighted by the block arrow 3(ii). This new goal is AND-ed with the expectation that Mary drinks and the requirement that the iFridge supplies cold drinking water. In other words, we can reduce the likelihood of Mary forgetting to drink by giving her a reminder by exploiting the iCups' capability to beep; this new goal mitigates the obstacle Forgets to drink, denoted by a solid bold arrow from goal to obstacle. An implication of the new goal, however, is that we need to estimate how much Mary drinks over time and issue reminders if her liquid intake falls below some ideal level. Hence, identification of the Achieve[ReminderToDrinkIssued] goal triggers a repeat of Step 1 to refine it down to the level of requirements, followed by Step 2 to build an uncertainty model for these new requirements. This mitigation tactic is illustrated in Figure 4; the goal lattice is extended with the goal Achieve[RemindertoDrinkIssued] and its refinements, and the corresponding uncertainty lattice is extended with the nodes Doesn't act on prompt and Calculated liquid intake shortfall inaccurate, along with their respective refinements. The extended goal lattice also includes a domain *assumption*, denoted by the trapezoid labelled Most drinking vessels are iCups, which we use here to record an assumption upon which the correctness of our analysis depends; that Mary will drink most of her water from iCups.

Step 3(iii): Mitigate Uncertainty by RELAXation. Performance of Step 3 on the new goals and uncertainty factors is interesting because it reveals that the uncertainty can be mitigated but not be entirely eliminated. In this case, the mitigation tactic is to add flexibility that accounts for the uncertainty directly into the goal specification, assuming that the goal is not an invariant. Hence, for example, the amount of liquid being taken from an iCup can be sensed, but it cannot be guaranteed that the liquid taken is being consumed by Mary. Mary might be using it to water her potted plants or simply spilling

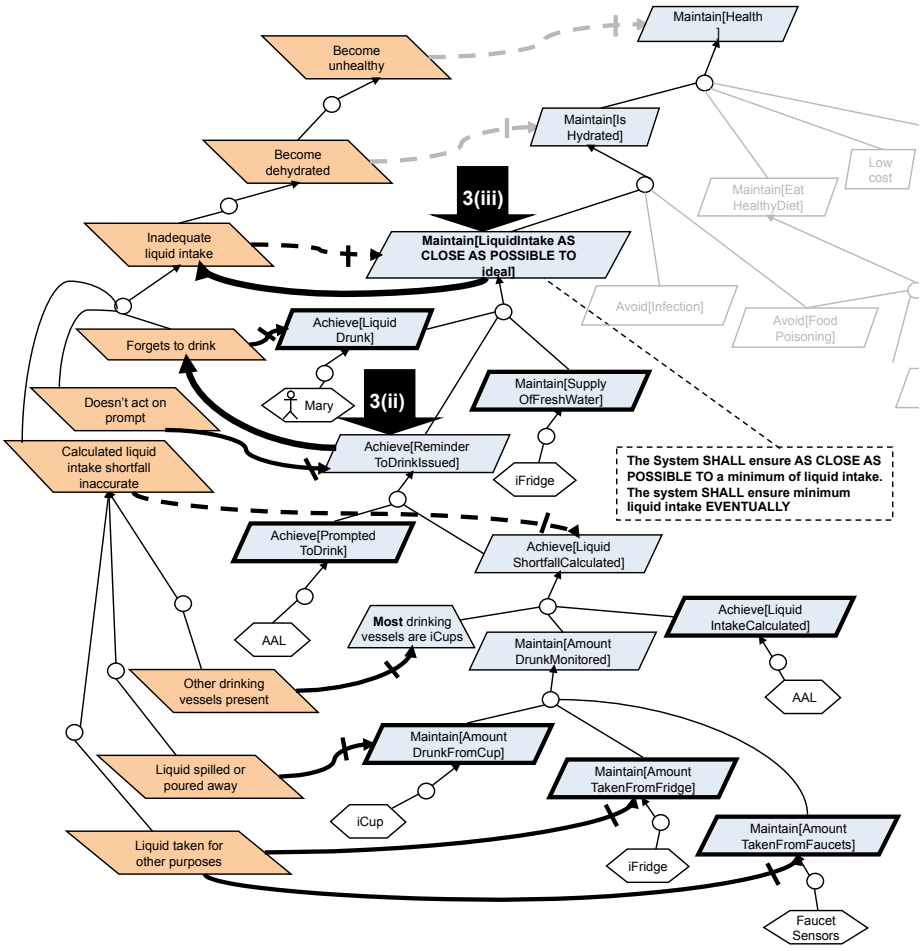

Fig. 4. Two types of uncertainty mitigation: by adding a new subgoal and RELAXing subgoal, denoted by block arrows 3(ii) and 3(iii), respectively

it. As a consequence, the Maintain[AdequateLiquidIntake] goal from Figure 2 cannot be guaranteed to be satisfiable under all circumstances. This uncertainty poses a problem; there does not appear to be a technological solution that can guarantee to accurately measure Mary's liquid intake, or one that will guarantee that Mary will act on reminders that she should drink. On the other hand, a temporary shortfall in the ideal liquid consumption may:

- Be normal - the temperature may be low, causing Mary to lose less liquid through perspiration;
- Be recouped later - it may lead to a mild headache (which may in turn prompt Mary to drink) rather than immediate organ failure;

Step 3 reveals that there is uncertainty about the environment (Mary's behaviour), yet, rather than calling into question the viability of the AAL, the uncertainty can be tolerated. Figure 4 shows the result of applying RELAX to Maintain[AdequateLiquidIntake], which has been reformulated as the goal (indicated by the block arrow 3(iii))

> *The System SHALL ensure AS CLOSE AS POSSIBLE TO a minimum of liquid intake. The system SHALL ensure minimum liquid intake EVENTUALLY.*

The arc leading from the goal and pointing to the Inadequate liquid intake obstacle indicates partial mitigation of the uncertainty over Mary's liquid intake. The goal is a composite comprising two clauses. The first mandates that although Mary's liquid intake cannot be measured with complete accuracy, the system should be designed to exploit the capabilities of the resources identified in the domain model to provide a best effort at liquid intake estimation. The second clause mandates that although under-consumption of liquid may occur, whenever this happens, the AAL must ensure that Mary's liquid intake recovers to acceptable levels at some point in the future. How to achieve eventual intake of the minimal level of liquid, and how soon is left to the AAL system's designers to determine.

Step 3(iv): Mitigate Uncertainty by adding a High-Level Goal. As implied above, Mary's liquid intake may fall below minimal levels so specification and RELAX-ation of goals aimed at getting Mary to drink cannot guarantee that she will not become dehydrated at some point. Mary might still forget to drink enough, or she could become dehydrated as a side-effect of acquiring an infection. If we are to prevent Mary from becoming unhealthy due to dehydration, we need to mitigate the uncertainty represented by the Become dehydrated obstacle in Figure 4. Mitigation of this uncertainty requires recourse to the most costly of our tactics, which is represented as Step 3(iv) of our process. Step 3(iv) triggers the search for a new goal, higher in the goal lattice than our RELAX-ed goal to maintain Mary's liquid intake, concerned with rehydrating Mary.

This mitigation approach is shown as the goal Achieve[ReHydration], indicated by the block arrow 3(iv) in Figure 5. Rehydrating Mary represents a radical change in the system behaviour. Instead of merely getting her to drink enough, we now need to cope with the emergency situation of getting her rehydrated before organ damage occurs. So urgent is this condition, that the new goal is OR-ed with the other high-level goals. In other words, the AAL suspends its goal of maintaining a healthy diet along with all the other goals that need to be satisfied if Mary is to lead a normal life, and divert resources into getting her rehydrated. This high-level goal represents a new target system, specified by refining the Achieve[ReHydration] goal and, of course, applying uncertainty modeling to ensure this new goal's refined sub-lattice is robust too. The arc leading from the new goal Achieve[ReHydration] to the obstacle Become dehydrated indicates the mitigation of the associated uncertainty.

Discussion. In summary, this example illustrated three different mitigation strategies for handling uncertainty in the environment. At the end of this process for addressing the Maintain[IsHydrated] goal, we included functionality in

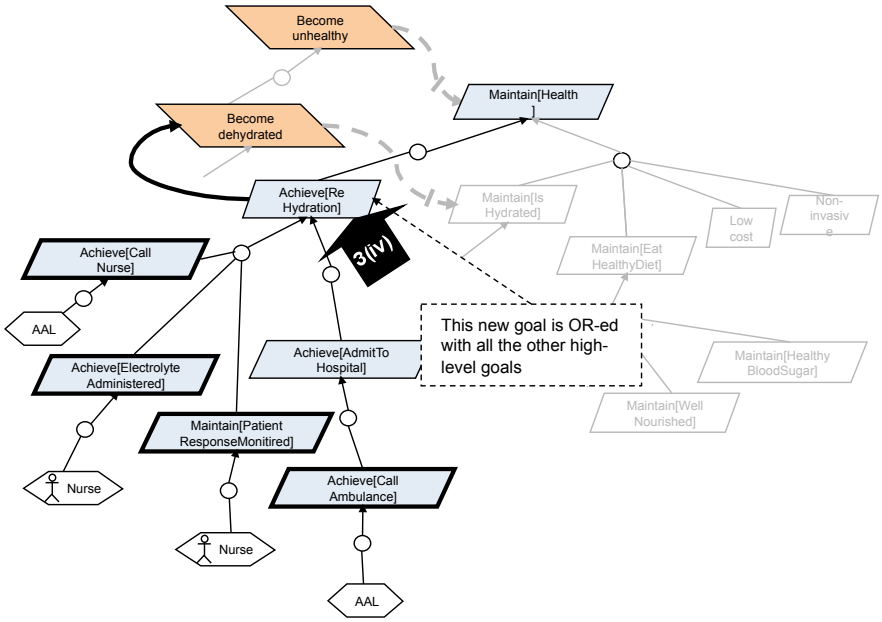

Fig. 5. Uncertainty mitigation with new high-level goal for new target system

the requirements for the original target system to support a reminder to drink feature in the iCups to account for the uncertainty with Mary's behavior. In order to make the target system more flexible with respect to the uncertainty associated with the water supply provided by the iFridge and take into account the lack of accuracy in the sensors measuring the liquid intake, we RELAXed the goal Maintain[AdequateLiquidIntake] to introduce flexibility in the quantity of liquid consumed and the time frame in which it can be consumed. Finally, to handle the uncertainty associated with severely adverse conditions with Mary (either her unwillingness to respond to the reminders or illness) and/or adverse conditions with the water supply, we introduced a new high-level goal Achieve[Rehydration] to account for the situation where Mary has become dehydrated and the system must provide new behavior to correct the situation. Dynamic adaptation is required to realize the third mitigation tactic since it requires a different target system to handle Mary's dysfunctional state, with the objective of bringing her and the system back to the point where the goal Maintain[IsHydrated] is satisfiable again. The other two mitigation strategies may be implemented statically with different branches of alternative behavior or realized by run-time adaptation, depending on the available technology to support the run-time flexibility.

4 Related Work

The increasing demand for self-adaptation has led to a surge of interest in software engineering for self-adaptive systems – see [10] for a recently compiled summary. Most of this work has been in the design of software architectures that enable flexible adaptations [11]. In general, such architectures share common characteristics that enable them to monitor and respond to environmental changes. Much less work has been carried out on how to explicitly incorporate the inherent uncertainty associated with adaptive systems into existing modeling languages. UML profiles exist that provide stereotypes for marking model elements that are in some way uncertain – e.g., an uncertainty profile [12] for capturing uncertainty in process modeling and fuzzy UML [13] for representing imperfect information in databases. Uchitel *et al.* [14] have also dealt with uncertainty using partial labelled transition systems (PLTS) to model aspects of the system behaviour that are unknown and remain undefined.

Limited work has also been performed in modeling and monitoring requirements for adaptive systems. Goal-based modeling notations, such as i* [15] and KAOS [1], have been applied to the specification of requirements of self-adaptive systems. Specifically, goal-based models are well suited to exploring alternative requirements and it is natural to use goal models to represent alternative behaviors that are possible when the environment changes [4, 16–18]. Furthermore, goal models can effectively be used to specify the requirements for transition between adaptive behaviours [4, 19]. With these approaches, however, the modeler must explicitly enumerate all possible alternative behaviours. In contrast, RELAX [5] supports a declarative approach for specifying requirements for a DAS, thus accounting for more flexibility in the system behavior.

Run-time monitoring of requirements dynamically assesses the conformance of run-time behaviour to the specified requirements [20]. This capability is a crucial enabler for self-adaptive systems as non-conformance to requirements may trigger an adaptation. Requirements monitoring approaches often rely on *ad-hoc* run-time representations of the requirements [21]. A more promising approach is to monitor goal models at run time as described in [22], where failed goals are diagnosed and fixed at runtime using AI theories of diagnosis. More generally, in the context of self-adaptive systems, it may only be possible to *partially* satisfy runtime goals – that is, goal satisfaction is not a "yes" or "no" decision. Adaptation decisions, therefore, may have to be made probabilistically. Letier and van Lamsweerde [23] have proposed a technique to quantify degrees of satisfaction in goal models but the work has not yet been applied to adaptive systems.

5 Conclusions and Future Work

Goals are objectives or statements of intent that the system should accomplish. For the case of adaptive systems, different environmental uncertainty factors may put at risk the accomplishment of such goals. In this paper, we have presented a goal-based modeling approach to specify the requirements of a DAS,

where environmental uncertainty associated with the goal specifications are explicitly integrated. The approach offers a systematic use of a range of tactics for adaptation to deal with uncertainty on a rising scale of costs. The tactics include adding low-level goals (the least costly approach), RELAXing requirements to express bounded uncertainty to accomplish a partial but still suitable satisfaction of the goals, and the identification of a new (high-level) goal to mitigate the uncertainty that leads to the identification of a new target system.

The general objective of goal modeling is to refine goals so that the set of subgoals that satisfy their parent goal is necessary and sufficient. One key lesson from reasoning with uncertainty is that, where uncertainty exists, the most we can hope for is that the subgoals are necessary. They will never be sufficient. Uncertainty must be handled, therefore, by assigning responsibility to a human agent or by introducing some intelligent or adaptive behavior into the software.

Several avenues for future research are possible. Estimation of the risk posed by uncertainty is implicit in the application of our process; i.e., our work requires risk to be inferred from the goal and uncertainty models. Further work is required towards systematic techniques to quantify the risk as a complement to threat modelling, understanding what we can RELAX (i.e. what is variant vs. what is invariant), and the extent to which we can RELAX requirements. We speculate that risk could be made explicit by quantifying it in the manner of attack trees [2]. The systematic approach for identifying target systems makes it possible to extend existing MDE-based approaches to DAS development (e.g., [3, 24]) to start at a higher-level of abstraction. That is, with the results from this work, we can start with a conceptual domain model of a DAS and systematically progress from goals and requirements to their designs and implementation.

References

1. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. John Wiley & Sons (2009)
2. Schneier, B.: Attack Trees - Modeling security threats. Dr. Dobb's Journal (Dezember 1999)
3. Zhang, J., Cheng, B.H.C.: Model-based development of dynamically adaptive software. In: ICSE '06: Proceedings of the 28th international conference on Software engineering, New York, NY, USA, ACM (2006) 371–380
4. Goldsby, H., Sawyer, P., Bencomo, N., Hughes, D., Cheng, B.H.C.: Goal-based modeling of dynamically adaptive system requirements. In: 15th Annual IEEE Int. Conf. on the Engineering of Computer Based Systems (ECBS). (2008)
5. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.M.: Relax: Incorporating uncertainty into the specication of self-adaptive systems. In: Proceedings of IEEE International Requirements Engineering Conference (RE09), Atlanta, Georgia (2009) (to appear in).
6. Yijun, Y., Lapouchnian, A., Liaskos, S., Mylopoulos, J., Leite, J.: From Goals to High-Variability Software Design. Volume 4994. Springer Berlin / Heidelberg (2008)
7. Mylopoulos, J., Chung, L., Yu, E.: From object-oriented to goal-oriented requirements analysis. Commun. ACM **42**(1) (1999) 31–37
8. Mead, N.: Identifying Security Requirements using the SQUARE Method. In: Integrating Security and Software Engineering: Advances and Future Visions. Idea Group (2006) 44–69

9. den Braber, F., Dimitrakos, T., Gran, B.A., Lund, M.S., Stölen, K., Aagedal, J.O.: The coras methodology: model-based risk assessment using uml and up. In: UML and the unified process. IGI Publishing, Hershey, PA, USA (2003) 332–357
10. Cheng, B.H.C., et al: 08031 – software engineering for self-adaptive systems: A research road map. In Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J., eds.: Software Engineering for Self-Adaptive Systems. Number 08031 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2008)
11. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In Briand, L.C., Wolf, A.L., eds.: FOSE. (2007) 259–268
12. Jing, X., Pinel, P., Pi, L., Aranega, V., Baron, C.: Modeling uncertain and imprecise information in process modeling with uml. In Das, G., Sarda, N.L., Reddy, P.K., eds.: COMAD, Computer Society of India / Allied Publishers (2008) 237–240
13. Ma, Z.M., Yan, L.: Fuzzy XML data modeling with the UML and relational data models. Data Knowl. Eng. **63**(3) (2007) 972–996
14. Uchitel, S., Kramer, J., Magee, J.: Behaviour model elaboration using partial labelled transition systems. In: ESEC/FSE-11: Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering, New York, NY, USA, ACM (2003) 19–27
15. Yu, E.S.K.: Towards modeling and reasoning support for early-phase requirements engineering. In: RE 97: Proc. of 3rd IEEE International Symposium on Requirements Engineering (RE97), Washington, DC, USA (1997)
16. Lapouchnian, A., Liaskos, S., Mylopoulos, J., Yu, Y.: Towards requirements-driven autonomic systems design. In: Workshop on the Design and Evolution of Autonomic Application Software (DEAS 2005). (2005)
17. Lapouchnian, A., Yu, Y., Liaskos, S., Mylopoulos, J.: Requirements-driven design of autonomic application software. In: Proceedings of CASCON 2006. (2006)
18. Yijun, Y., Lapouchnian, A., Liaskos, S., Mylopoulos, J., Leite, J.: From Goals to High-Variability Software Design. Volume 4994. Springer Berlin / Heidelberg (2008)
19. Morandini, M., Penserini, L., Perini, A.: Modelling self-adaptivity: A goal-oriented approach. In: SASO '08: Proc. of 2008 Second IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems. (2008) 469–470
20. Fickas, S., Feather, M.: Requirements monitoring in dynamic environments. In: Second IEEE International Symposium on Requirements Engineering (RE'95). (1995)
21. Dingwall-Smith, A., Finkelstein, A.: Checking complex compositions of web services against policy constraints. In Augusto, J.C., Barjis, J., Ultes-Nitsche, U., eds.: MSVVEIS, INSTICC PRESS (2007) 94–103
22. Wang, Y., McIlraith, S.A., Yu, Y., Mylopoulos, J.: An automated approach to monitoring and diagnosing requirements. In Stirewalt, R.E.K., Egyed, A., Fischer, B., eds.: ASE, ACM (2007) 293–302
23. Letier, E., van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. In: Proc. of 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering. (2004) 53–62
24. Morin, B., Fleurey, F., Bencomo, N., Jézéquel, J.M., Solberg, A., Dehlen, V., Blair, G.: An aspect-oriented and model-driven approach for managing dynamic variability. In: MoDELS '08: Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems, Berlin, Heidelberg, Springer-Verlag (2008) 782–796