Goal-Based Modeling of Dynamically Adaptive System Requirements*

Heather J. Goldsby[1], Pete Sawyer[2], Nelly Bencomo[2], Betty H.C. Cheng[1], Danny Hughes[2]
[1] Department of Computer Science and Engineering, Michigan State University,
East Lansing, MI 48824, USA
[2] Computing department, InfoLab21, Lancaster University, LA1 4WA, United Kingdom
{hjg, chengb}@msu.edu; {sawyer, nelly, danny}@comp.lancs.ac.uk

Abstract

Self-adaptation is emerging as an increasingly important capability for many applications, particularly those deployed in dynamically changing environments, such as ecosystem monitoring and disaster management. One key challenge posed by Dynamically Adaptive Systems (DASs) is the need to handle changes to the requirements and corresponding behavior of a DAS in response to varying environmental conditions. Berry et al. previously identified four levels of RE that should be performed for a DAS. In this paper, we propose the Levels of RE for Modeling *that reify the original levels to describe RE modeling work done by DAS developers. Specifically, we identify four types of developers: the system developer, the adaptation scenario developer, the adaptation infrastructure developer, and the DAS research community. Each level corresponds to the work of a different type of developer to construct goal model(s) specifying their requirements. We then leverage the Levels of RE for Modeling to propose two complementary processes for performing RE for a DAS. We describe our experiences with applying this approach to GridStix, an adaptive flood warning system, deployed to monitor the River Ribble in Yorkshire, England.*

1. Introduction

Increasingly, Dynamically Adaptive Systems (DASs) are addressing complex problems that require a high degree of assurance [17, 20]. Studies have shown that errors introduced in the requirements are the most costly to fix for traditional (i.e., non-adaptive) systems [15]. Given the complexity of DASs, the need to apply rigorous requirements engineering (RE) is further heightened. Berry *et al.* [2] previously identified four levels of RE that should be performed for a DAS. In this paper, we propose a modeling version of the Levels of RE that reifies the original levels to describe RE work done by DAS developers to create goal models specifying their requirements for a DAS. Specifically, we identify four types of developers: the system developer, the adaptation scenario developer, the adaptation infrastructure developer, and the DAS research community. Each level corresponds to the work of a different type of developer to construct goal model(s) specifying their requirements. We then leverage the Levels of RE for Modeling (LoREM) to propose two complementary processes for performing RE for a DAS.

To date, several notable goal-oriented approaches have been proposed for modeling DAS requirements [7, 6, 22, 14, 23]. A DAS has three main types of RE concerns: what are the *conditions to monitor* for adaptation, what *adaptations* are needed to achieve a desired new behavior, and what *decision-making procedure* should be used to associate the monitored conditions to the appropriate adaptations. In general, the current approaches model the concerns of one developer, namely the system developer, and focus on specifying the conditions to monitor for adaptation, but do not explicitly model the other two RE concerns (adaptations and decision making). As such, there is a need for additional approaches to modeling the RE concerns of all DAS developers.

In this paper, we propose the LoREM as an approach to modeling the requirements of a DAS using *i** goal models [21]. Our approach reifies each of the original levels to describe work performed by a specific type of DAS developer to produce *i** goal models that are intended to be integrated with those produced by the other developers. We use the *i** goal models to represent the stakeholder objectives, non-adaptive system behavior (business logic), adaptive behavior, and adaptation mechanism needs of a DAS. Each of these *i** goal models addresses the three RE concerns (conditions to monitor, decision-making procedure, and possible adaptations) from a specific developer's perspective.

*This work has been supported in part by NSF grants EIA-0000433, EIA-0130724, CDA-9700732, CCR-9901017, Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, Eaton Corporation, Siemens Corporate Research, and a grant from Michigan State University's Quality Fund and the EPSRC project EP/C010345/1 The Divergent Grid.

We leverage the LoREM to propose two complementary processes for performing RE for a DAS, where each process step corresponds to performing the activities described by one of the levels. Specifically, we offer an *application-driven process* that assumes a mature set of adaptation mechanisms that enable a DAS to dynamically adapt. And we present a *technology-driven process* that assumes a less mature set of adaptation mechanisms, thus constraining the range of possible adaptive behavior.

We illustrate the technology-driven process and the LoREM in the context of GridStix, an adaptive flood warning system, deployed to monitor the River Ribble in Yorkshire, England. We demonstrate how the process assists GridStix developers in understanding how the DAS needs to adapt in response to changing river conditions. The remainder of the paper is organized as follows. Section 2 presents the original Levels of RE. Section 3 presents the LoREM. Section 4 offers two process models for specifying the requirements of a DAS. Section 5 describes our case study. Section 6 discusses related work. Finally, in Section 7, we present conclusions and discuss future work.

2. Levels of RE for a DAS

In this section, we introduce the Levels of RE. For clarity, a DAS is assumed to be a collection of steady-state programs, called *steady-state systems* one of which is executing at a given point in time. We consider an adaptation to be the dynamic transition from executing one steady-state system, the *source system* to running another steady-state system, the *target system*. A DAS is supported by an *adaptation infrastructure* comprising a set of mechanisms that enable adaptation to occur. The adaptation infrastructure must include three key types of mechanisms. First, a *monitoring mechanism* is hardware or software that is responsible for detecting adaptation conditions at run time. Second, a *decision-making mechanism* is software that is responsible for selecting a target system to adapt to based on input from the monitoring mechanisms at run time. Third, an *adaptation mechanism* is software or hardware that executes adaptive steps at run time, where an *adaptive step* is an adaptive action, e.g., adding or swapping a component. The adaptation infrastructure may have been developed as a single, comprehensive, and integrated collection of elements that support the three types of mechanisms. Alternatively, it may have been constructed from a collection of disparate mechanisms from different sources, thus requiring "glue" code to make the mechanisms compatible.

The four Levels of RE done for a DAS specified by Berry et al.[2] are as follows:

Level 1 is the traditional RE work done for a system. Specifically, it deals with the application domain of a DAS and identifies all possible steady-state systems that can be executed by the DAS after adaptation.

Level 2 is the RE work done by the DAS itself at run time to detect the need to adapt and to select the appropriate target system to adopt.

Level 3 is the RE work done to select and configure the DAS adaptation infrastructure for a specific DAS application (e.g., what kinds of monitoring options exist to support a given monitoring need?). A given adaptation mechanism may be used for multiple adaptation needs within a given DAS, and there may be many different adaptation mechanisms from which to choose.

Level 4 is the RE research into adaptation to identify the adaptation infrastructure needs. For example, what type of monitoring support (e.g., software sensors, hardware sensors) is needed? What is the granularity of the monitoring data types? What type of monitoring (e.g., centralized, distributed real-time) needs to be performed?

3. Levels of RE for Modeling a DAS

The LoREM reify the levels in four key ways: First, Level 2 describes the work performed by a developer. In the original Levels of RE, Level 2 referred to the RE work done by a DAS at run time. Feedback from DAS developers indicated that while this is true for idealistic (and futuristic) DASs, it is not realizable with current technology. Second, we identify four types of developers, where each level corresponds to the work of a different developer to construct goal model(s) describing their requirements for a DAS. Third, each level describes the modeling work performed by the specific developer, models constructed, and how to integrate the models constructed at the other levels. We identified the tasks of each developer by refining the work description in the original levels to describe modeling activities present in the model-driven development (MDD) of a DAS. Fourth, to provide context for the LoREM and guidance for integrating the RE work into an overall development process for a DAS, each level is annotated with the MDD phase(s) in which its activities occurs.

The levels are not ordered according to levels of abstraction, or order for performing RE tasks, but instead, according to the level of "meta-ness" [2]. Level 2 artifacts describe how Level 1 artifacts are to be composed; Level 3 identifies the set of adaptation mechanisms to be used by Level 2; and Level 4 models the collection of adaptation mechanisms available for selection in Level 3, for a specific DAS. Thus, the LoREM do not have a one-to-one mapping to development phases. Moreover, any given level may only describe a portion of the work done during any development phase. Other MDD artifacts for non-adaptive system elements may be integrated with the LoREM artifacts.

Briefly, we overview the phases of MDD of a DAS (depicted in Figure 1) prior to describing how the work performed for the LoREM produces several of these artifacts. Figure 1 depicts the phases of MDD of a DAS and the artifacts created at each phase. At the Goal phase, the functional goals (Goal) and non-functional, or soft, goals (Softgoal1 and Softgoal2) of the DAS are identified. These are represented as roundtangles and clouds, respectively. At the

Requirements phase, the domains (D_i and D_j), i.e., environmental conditions, of the DAS are identified and the requirements for realizing the goals in each domain are captured in a requirements model (e.g., R_i and R_j). The requirements models are represented as parallelograms. The effect of these requirements models on the softgoals are captured as dotted arrows from the requirements models to the soft goals labeled with help/hurt. Additionally, adaptations among these requirements models are captured as solid line arrows between the requirements models. At the Design phase, design models (e.g., M_i, M_j, $M_{i,j}$) are constructed, where $M_{i,j}$ is a design model capturing the behavior of the system during adaptation. At the Implementation phase, code is created. The circles in the Implementation phase represent adaptation infrastructure mechanisms. At the Mechanism Selection phase, adaptation infrastructure for a specific DAS is selected. RE is used to identify what specific adaptation mechanisms are needed to realize the adaptations among requirement models. Lastly, at the Development of Adaptation Infrastructure Mechanism phase, adaptation infrastructure mechanisms, e.g., monitors, decision-makers, and adaptation mechanisms, are created. These are depicted as circles. RE activities in this development phase need to identify the general adaptation needs for different domains. The LoREM describe work performed for the Goal, Requirements, Mechanism Selection, and Development of Adaptation Infrastructure Mechanisms phases. Next we elaborate each of the levels.

3.1. Level 1 RE

Developer: System developer
Development Phase: Goal and Requirements phases
Tasks: In the Goal phase, the system developer first identifies the essential goals and softgoals (e.g., performance, reliability) of the DAS. Second, the system developer identifies the goals and softgoals of the DAS adaptation infrastructure. These goals and softgoals are captured in a goal model, which we refer to as the *objectives model*.

In the Requirements phase, the system developer first works closely with a domain expert to identify the unique domains in which the DAS will operate. Second, based on the domains, the goals, and the softgoals of the DAS, the system developer identifies a set of steady-state systems, such that each steady-state system is suitable for at least one domain and satisfies the goals of the DAS. For each steady-state system, the system developer creates a requirements model (e.g., R_i) by describing specific requirements that the steady-state system should satisfy to achieve the primary goals and softgoals of the DAS. These requirements include *local properties*, which are the local requirements of a steady-state system in a specific domain. We refer to these models as *steady-state system behavior models* or *behavior models* for brevity.
Models: The system developer produces an objectives

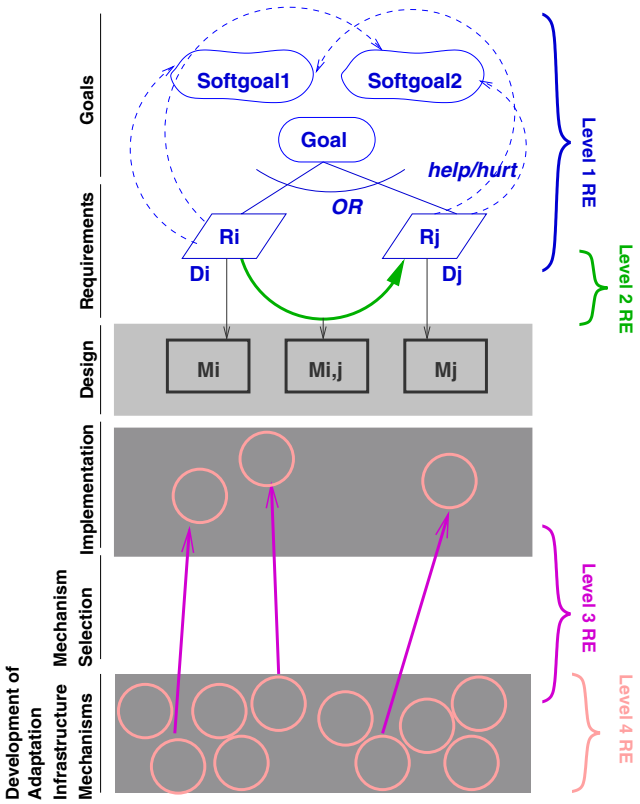

Figure 1. MDD Phases

model and a behavior model for each steady-state system. These requirements models can be refined into design models during the Design phase, from which code can be generated.

3.2. Level 2 RE

Developer: Adaptation scenario developer
Development Phase: Requirements phase
Tasks: I is the RE work done to specify possible DAS adaptations by identifying source system, target system, and adaptation infrastructure requirements for supporting adaptation from a source system to a target system (e.g., what has to be monitored, what subset of target systems from Level 1 RE are to be used in specifying adaptation scenarios). In the original Levels of RE described by Berry et al., Level 2 referred to the RE work done by a DAS at run time. Feedback from DAS developers indicated that while this is true for idealistic (and futuristic) DASs, it is not realizable with current technology. Thus, for this paper, we modified Level 2 to describe the work of a developer.

n the Requirements phase, the adaptation scenario developer is responsible for creating a set of adaptation scenarios for the DAS, where an *adaptation scenario* is an acceptable adaptation transition at run time between the source system and the target. These adaptation scenarios should enable the DAS to continually meet its goals and softgoals (iden-

tified as part of Level 1 RE) despite a continually changing environment. Then for each adaptation scenario, the adaptation scenario developer creates an adaptation model. An *adaptation model* specifies requirements of the monitoring mechanism, decision-making mechanism, and the adaptation mechanism necessary to accomplish this adaptation. These requirements include specific conditions that should be monitored and a high-level description of what functional or non-functional behavior should change during adaptation. This description can be created by the adaptation scenario developer by comparing the behavior of the source and target systems.

Models: The adaptation scenario developer constructs an adaptation model for each possible adaptation scenario. For each adaptation model, both the source and target systems must have previously been specified as a part of Level 1 RE. These adaptation scenario models can be refined into design models for adaptation during the Design phase.

3.3. Level 3 RE

Developer: Adaptation infrastructure developer

Development Phase: Mechanism Selection phase

Tasks: The adaptation infrastructure developer performs RE work to identify what adaptation infrastructure capabilities are needed to support the adaptation scenarios developed for Level 2 and the adaptation infrastructure goals identified at Level 1. Specifically, the adaptation infrastructure developer is responsible for identifying: What types of monitoring, decision-making, and adaptation support is needed? What mechanisms satisfy the goals and softgoals of the adaptation infrastructure of the adaptive infrastructure (e.g., maintainability, reliability) specified at Level 1? The adaptation infrastructure developer then identifies monitoring mechanisms, decision making mechanisms, and adaptation mechanisms that meet the identified criteria. These selections are documented in an *adaptation infrastructure model*. The model specifies the goals and softgoals of the adaptation infrastructure and the mechanisms that satisfy the goals and satisfice the softgoals. This model will later serve as documentation for why specific mechanisms were selected to support the DAS.

Models: The adaptation infrastructure developer constructs an adaptation infrastructure model for the DAS.

3.4. Level 4 RE

Developer: DAS Research Community

Development Phase: Development of Adaptation Infrastructure Mechanisms phase

Tasks: The DAS Research Community performs RE work to determine the requirements for the three major types of adaptation infrastructure elements, i.e., monitoring, decision-making, and adaptation mechanisms.

Models: A model depicting the various types of adaptation infrastructure elements and the requirements they respectively satisfy is created. Because Level 4 RE is not performed by the developers for a specific DAS, it is assumed to have occurred prior to the development of a specific DAS [10, 13, 16, 18] and thus is not included in our RE process models.

4. RE Process Models for a DAS

The order in which the LoREM activities are performed depends upon the maturity of the supporting adaptation infrastructure. To that end, we offer two complementary processes for creating models of the requirements of a DAS, where the steps for both processes draw from the activities in the LoREM. An application-driven process, which is primarily top-down, assumes a rich set of adaptation infrastructure components. A technology-driven process, which is primarily bottom-up, assumes a more sparse set of adaptation infrastructure components. In the following, we describe each process in greater detail.

4.1. Application-Driven RE Process

In the application-driven process, the RE activities drawn from the LoREM are ordered: Level 1, Level 2, and then Level 3. Essentially, the application-driven process model represents the ideal RE process in which the system functionality is determined at Level 1, the adaptation scenarios are determined at Level 2, and then the adaptation infrastructure is selected and configured to support the desired system functionality and adaptation scenarios. This process model relies upon a mature set of adaptation infrastructure components that meet the requirements of a wide variety of adaptation needs and behavior. As such, the adaptation scenario developer can focus solely on achieving the appropriate functional (possibly adaptive) behavior, without being constrained by adaptation technology.

4.2. Technology-Driven RE Process

In the technology-driven process model, the RE activities drawn from the LoREM are ordered: Level 1, Level 3, and then Level 2. This process model represents the current approach of many DAS developers where system functionality is determined at Level 1, an adaptation infrastructure is selected at Level 3, and then the adaptation scenarios are designed at Level 2, where the adaptation infrastructure significantly influences the possible range of adaptation scenarios. One reason for this strategy is that DASs are a relatively new technology with limited adaptation infrastructure support. Thus, many developers use this process to avoid modeling an adaptation scenario that cannot be supported by existing adaptation technology. Another reason for using this approach is if a particular adaptation infrastructure is mandated by the customer. Furthermore, often adaptation mechanisms and infrastructure are domain specific.

4.3. Discussion

Using either process model, this general approach to modeling the requirements for a DAS offers three key benefits. First, it identifies the key developers and partitions each developer's concerns and requirements into a separate Level of RE. This separation assists the developers in gaining a better understanding of the role they play in the context of the development of a specific DAS and also the dependencies that exist among their tasks, requirements, and models. Second, this approach produces models that can be used to guide the design of a DAS that takes into consideration a more complete view of the requirements, including softgoals, adaptation infrastructure, and the three key concerns of all adaptive systems (monitoring, decision-making, and adaptation). Third, this approach provides more flexibility in defining process models for performing RE for a DAS. As with most process models used for software engineering, these process models offer an idealistic approach. In reality, it is more likely that hybrid versions of the process models are used. Specifically, we expect there to be iterations between Level 2 and Level 3 where adaptation scenarios are tightly tied to the adaptation infrastructure.

5. Case study: A flood warning system

Many existing flood warning systems use on-site wireless sensor networks (WSNs) to collect data. Typically, such sensor networks can only record and transmit sensor data. The data is transmitted off-site to be processed by computationally-intensive predictive models. Flood warning systems are often located in remote areas, where only low bandwidth cellular network technologies are available.

The emerging availability of powerful embedded hardware and heterogeneous wireless networking technologies is enabling a new generation of flood warning systems. Although constrained by size and power supply, modern sensor nodes have sufficient CPU power and memory to perform useful computations and can communicate using short-range but high-bandwidth wireless technologies such as IEEE 802.11b. These characteristics enable a sensor network to act as a lightweight grid enabling the on-site execution of *point prediction models*. A node performing point prediction requires sensor data from at least one upstream node, combined with local data. Thus, a node depends on upstream nodes while simultaneously sharing its data with down-stream nodes. Point prediction models are sufficiently lightweight to be executed on a single node.

Grid-based computing is also enabling the exploitation of off-the-shelf hardware. For example, inexpensive digital cameras may be used instead of expensive ultrasound sensors for measuring river flow. Ultrasound sensors generate small volumes of data that can be continuously transmitted off-site for processing. They have to be mounted under the water, making maintenance challenging. In contrast, high-resolution digicams can be mounted on bridge parapets and can be used to detect the movement of tracer particles in the water. Digicams generate large volumes of data that are expensive to transmit off-site and processing the data is computationally expensive for a resource-constrained node. Therefore, the rate at which data can be sampled and processed is limited. Fortunately, digicams' easily decomposable datasets lend themselves to distributed processing among the nodes of the sensor network.

A sensor network with local computational power may enable a flood warning system to adapt dynamically in response to changing environmental conditions, e.g., CPU clock speeds can be varied, network topologies can change, and nodes can adapt to switch between wireless communication technologies as the requirements for energy conservation, real-time computation, and fault-tolerance vary according to environmental conditions.

A prototype flood warning system called GridStix [12] has recently been designed and deployed on the flood plain of the River Ribble in Yorkshire, England. GridStix was used as a preliminary case study to test our hypothesis that the separation of concerns provided by the Levels of RE was useful for modeling the requirements for a DAS.

5.1. GridStix

Developing and deploying GridStix was a collaboration between computer and environmental scientists. The computer scientists had previously developed the GridKit middleware technology [10, 11]. Hence, GridStix was developed to use GridKit as its adaptation infrastructure. Thus, the RE performed for GridStix is an example of the technology-driven process.

The models constructed by performing the LoREM activities were specified with the *i** notation [21]. The *i** framework was selected because it describes the dependencies among *actors* to accomplish *goals*, accomplish *tasks*, and produce *resources*, and thus is well-suited for addressing the concerns of the four DAS types of developers. The RE performed for GridStix used two types of *i** models: *strategic dependency* models that describe dependencies among actors and *strategic rationale* models that describe the actors' internal rationale for how dependencies are met and why dependencies are created. In each, an *actor* is depicted as a circle, and its boundary is depicted as a dotted line. All elements within the circle are under the responsibility of the actor. An *agent* is an instance of an actor and is depicted as a circle with a horizontal bar across the top.

A *role* is a persona that an actor can adopt and is depicted as a circle with a curved bar across the bottom. A *goal* is an objective of an actor and is depicted as a roundtangle. A *softgoal*, depicted as a cloud, is a non-functional goal whose satisfaction cannot be fully evaluated. A *task*, depicted as a hexagon, is an activity performed by an actor. A *resource*, depicted as a rectangle, is a physical or informational entity.

We use six types of *i** relationships: First, a *dependency relationship* (depicted as an arc with the letter D pointing toward the dependee) indicates an element depends upon another element. Second, a *task decomposition relationship* (depicted as an arc that is crossed) indicates the task is decomposed into its constituent parts. Third, a *means-end relationship* (depicted as an arc with a solid arrowhead) indicates that a task satisfies a specific goal. Fourth, a *contribution relationship* (depicted as an arc with the word helps or hurts) connects an element to a softgoal and indicates if the element helps or hurts the realization of the softgoal. Fifth, an *actor-agent relationship* (depicted by arrow-headed arcs with the label ISA) indicates that an actor is an agent. Sixth, an *actor-role relationship* (depicted by arrow-headed arcs with the label PLAYS) indicates that an actor plays a role. A legend is included in Figure 2.

5.2. GridStix Level 1

In Level 1, the system developer performed RE to first identify and model the objectives of GridStix and then identify and model the possible steady-state systems. Figure 2 depicts the GridStix objectives model as an *i** strategic dependency model. For GridStix, the Environment Agency (the UK organization charged with managing the environment) depended on the Flood warning system (the business logic of GridStix) to satisfy the goal Predict flooding and three associated softgoals: (1) Prediction accuracy to avoid warning failures and false alarms; (2) Energy efficiency because the deployed sensors had to depend on limited power supplies, such as batteries or solar panels; and (3) Fault tolerance because the sensors' remote location made it difficult to gain access for maintenance. This inaccessibility was particularly applicable during flood events when sensors were at most risk, thus requiring the system to tolerate sensor failures.

The Flood warning system depended on the Adaptation infrastructure to provide Monitoring, Decision-making and Adaptation mechanisms. Each mechanism was modeled as a role responsible for satisfying an associated goal. For example, the Monitoring mechanism depended on the availability of Depth and Flow rate data (i.e., resources) to achieve its goal, which was to Monitor River. Additionally, the Flood warning system depended on the Adaptation infrastructure to have a Small footprint, since the sensors were resource-constrained, and to provide Evolvability, since the flood models for the river and the developers' understanding of the conditions under which GridStix needed to adapt were expected to improve over time.

Flood warning is a highly specialized and complex domain, thus the GridStix system developer worked with a hydrologist domain expert to identify discrete domains of the river. For the River Ribble three domains were identified: Normal, Flow increase, and Flood. A different steady-state system was associated with each domain: S1: Normal, S2: Flow increase, and S3: Flood. For S1, the river flow rate and water depth were within ranges that indicate that flooding was not imminent. Under these conditions, the system needed to consume minimal energy, consistent with the routine sampling of flow and depth data needed to monitor the state of the quiescent river. For S2, the river flow rate had increased beyond a threshold value, indicating that a significant and potentially damaging increase in water depth was about to occur. For this domain, frequent data sampling was needed so that further changes in river state could be quickly detected to enable precise behavior predictions. For S3, the water depth had risen above a threshold value in which sensor failure was probable. The system needed to operate in a way that was fault-tolerant and, as with S2, there was a need for frequent and precise data collection to accurately monitor the dangerous river state and predict how the river state would develop. These three steady-state systems were modeled in the objectives model as agents of the Flood warning system.

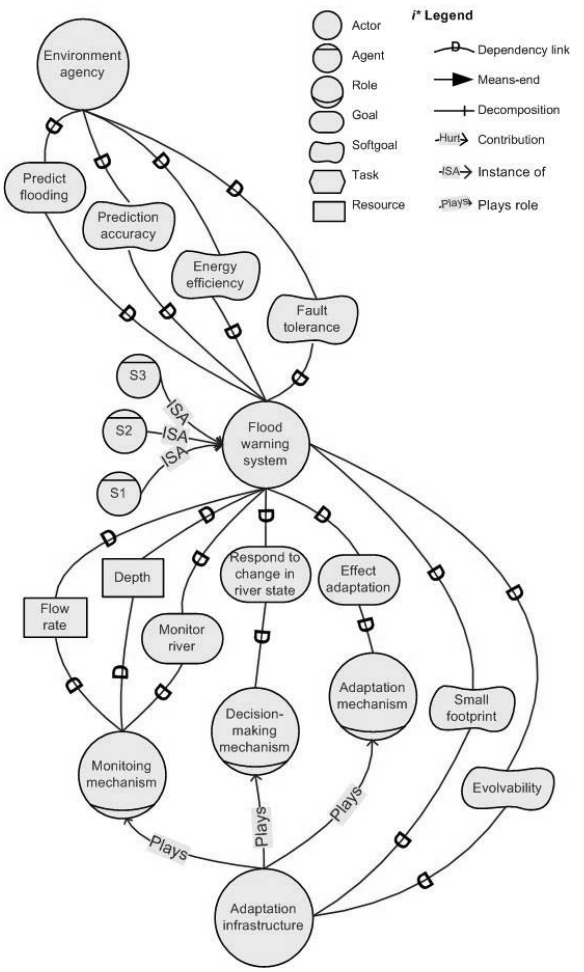

Figure 2. Level 1: GridStix objectives model

Next, the system developer constructed a behavior model

for each of the three steady-state systems. The models that were developed for S1, S2 and S3 are depicted in Figures 3, 4, and 5, respectively. In each, the steady-state system was depicted as an agent that addresses the goal and softgoals of the Flood warning system: Predict flooding, Fault-tolerance, Energy efficiency and Prediction accuracy.

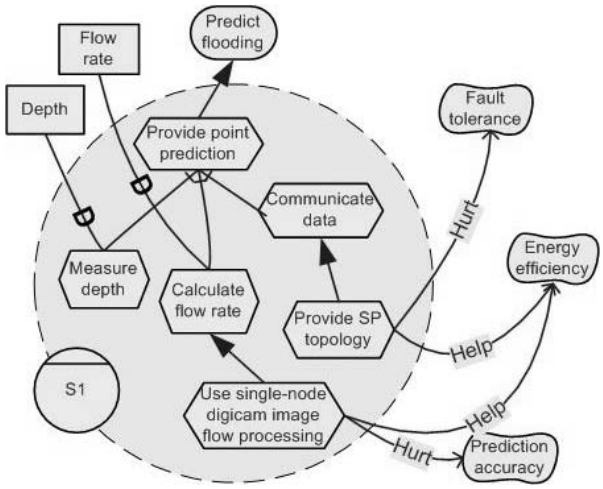

Figure 3. Level 1: Behavior Model S1: Normal

For example, for steady-state system S1 (Figure 3), the goal Predict flooding was achieved by the task Provide point prediction, which was decomposed into tasks Measure depth, Calculate flow rate, and Communicate data. Measure depth produced data modeled as the resource Depth, while Calculate flow rate produced Flow rate data. The Depth and Flow rate data represented GridStix' local properties and were referenced by the subsequently developed Level 2 models.

For S1, when the river was quiescent, Energy efficiency was judged to have a higher priority than Prediction accuracy and Fault tolerance. Thus, the Calculate flow rate was satisfied using digicam image processing to be performed by a single host node (Use Single-node digicam image flow calculation), which provided less accurate predictions than using multi-node image processing, but was more energy efficient. Similarly, because there was little risk of sensor failure, the relatively efficient, but less fault-tolerant Shortest Path network topology (Provide SP topology) was selected to Communicate Data.

Target systems S2 and S3 satisfied goal Predict flooding using tasks that differ in their contributions to the softgoals Fault-tolerance, Energy efficiency, and Prediction accuracy. In S2 (Figure 4), an increase in flow rate was taken to potentially presage an increase in depth, so Prediction accuracy was strengthened at the expense of Energy efficiency by applying Use Multi-node digicam image flow calculation. In S3 (Figure 5), the water depth had increased to the point where

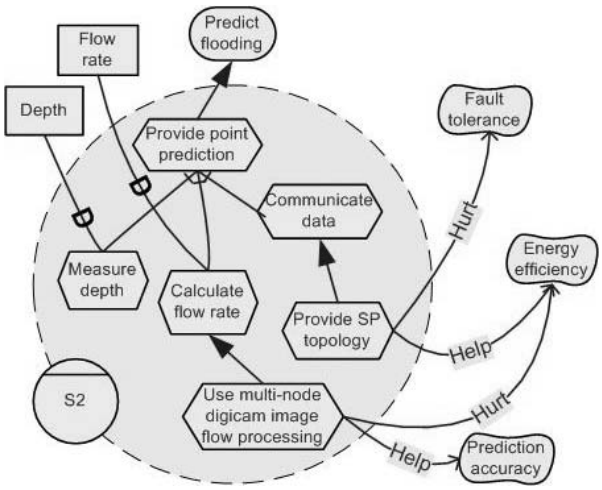

Figure 4. Level 1: Behavior Model: S2: Flow increase

sensors were threatened by submersion or debris, so Fault tolerance was helped to the detriment of Energy efficiency by using a Fewest-Hop spanning tree (Provide FH topology) for data communication.

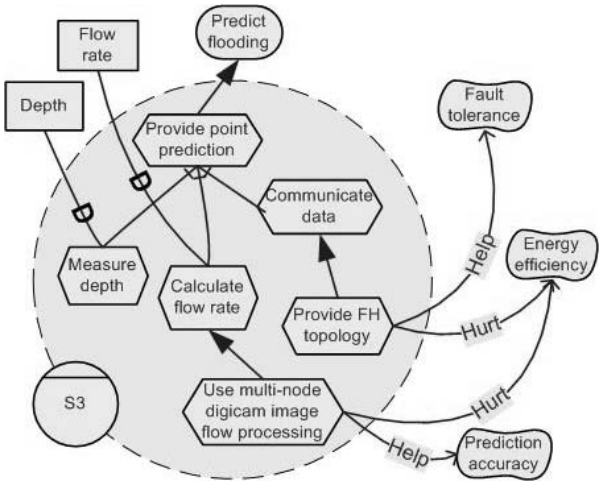

Figure 5. Level 1: Behavior Model: S3: Flood

5.3. GridStix Level 3

The next step in the technology-driven process was for the adaptation infrastructure developer to perform RE to identify adaptation infrastructure mechanisms that satisfy the goals specified by Level 1 and model them in an adaptation infrastructure model (Figure 6). For GridStix, GridKit was selected as the Adaptation infrastructure. GridKit, is based on the OpenCOM [3] component framework that uses a set of built-in reflective meta-models. GridKit is thus able to reason about its architecture and reconfigure itself

by component substitution at run time. Adaptive behavior in GridKit is policy-driven and defined by sets of rules. In concert with GridKit's context engine, the rules define how GridKit satisfies a requirement in a given environmental context, and also how GridKit adapts to changed environmental context (domains).

At Level 1, the system developer identified four goals for the adaptation infrastructure the objectives model (Figure 2): Enable adaptation and its subgoals, Monitor river, Respond to change in river state, and Effect adaptation. Specifically, these goals were specified as the responsibilities of the three roles (Monitoring mechanism, Decision-making mechanism, and Adaptation mechanism) played by the Adaptation infrastructure. These roles were effectively conflated into the single Adaptation infrastructure actor at Level 3 for convenience. In addition to satisfying the functional goals, the Adaptation infrastructure also had to contribute to softgoals Small footprint and Evolvability. GridKit satisfies the goals Monitor river, Respond to change in river state and Effect adaptation by tasks Use context engine, Use policy rules and Use component substitution, respectively. In addition, the softgoals Small footprint and Evolvability were helped by the use of component substitution and policy rules, respectively.

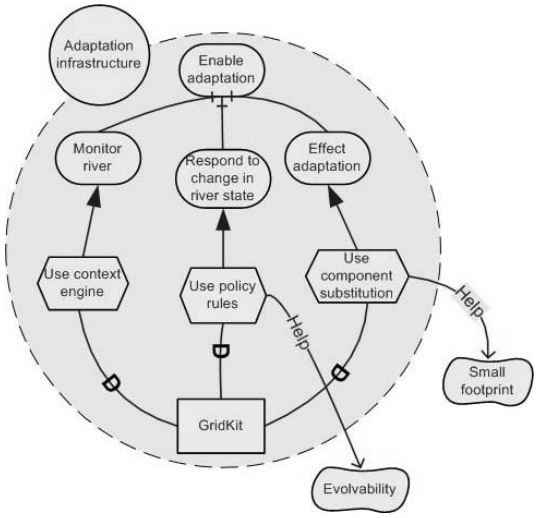

Figure 6. Level 3: Adaptation Infrastructure Model

5.4. GridStix Level 2

Lastly, the adaptation scenario developer identified and modeled adaptation scenarios that specify transitions between steady-state systems as the river state changes from one domain to another. Figure 7 depicts the adaptation scenario for adapting from S1 to S2 as the river state transitions between the domains Normal and High flow. In addition to specifying the source and target systems, each adaptation scenario must address three concerns that determine when and how to adapt. These are: what data to monitor; what changes in the monitored data trigger the adaption; and how the adaptation is effected. Each of these three concerns was conceptualized as the responsibility of a role of the adaptation infrastructure: Monitoring mechanism, Decision-making mechanism and Adaptation mechanism, respectively. As such, the goals of these roles are also modeled at Level 1 in the objectives model and at Level 3 in the adaptation infrastructure model (Figure 2 and Figure 6, respectively).

Briefly, we discuss the satisfaction of the goals by their respective roles. For GridStix, the goal of Monitoring mechanism was to Monitor river. This goal was decomposed into two tasks concerned with evaluating the Flow rate and Depth, which were modeled as local properties in the Level 1 behavior models. The results of these evaluations were used by the Decision-making mechanism to achieve its goal: Respond to changes in river state. Specifically, the Decision-making mechanism depended upon S1, the source system, and the evaluations to determine when GridStix needed to adapt from S1 to S2. This adaptation was specified to occur on a significant increase in river flow but before any significant depth increase. The Adaptation mechanism had to satisfy the goal Effect adaptation by performing the task Replace single-node digicam image processing with distributed digicam image processing, which defined, at a high-level, the difference between the S1 and S2 behavior models.

For GridStix, the specification of the behavior of the Monitoring mechanism and Decision-making mechanism were not dependent on knowledge of the underlying Adaptation infrastructure selected as part of Level 3. However, the specification of the behavior of the Adaptation mechanism was assisted by knowledge of the capabilities of the GridKit middleware, i.e., knowledge of the component substitution mechanism that enables switching between local and distributed processing.

5.5. Discussion

Due to space constraints, only a subset of the *i** models for GridStix have been depicted. For example, we have not depicted tactics to enable submerged sensors to continue transmitting data. However, we have described the principal goals, softgoals, and tasks that were used to drive the specification of GridStix and elaborated how they were used to derive lower-level requirements and selection of an adaptation infrastructure.

GridStix's developers needed to understand the characteristics of the problem domain in order to first specify the requirements and then derive a system that satisfied these requirements. The complexity of the problem and the developers' lack of familiarity with the problem domain necessitated the explicit separation of normal river conditions

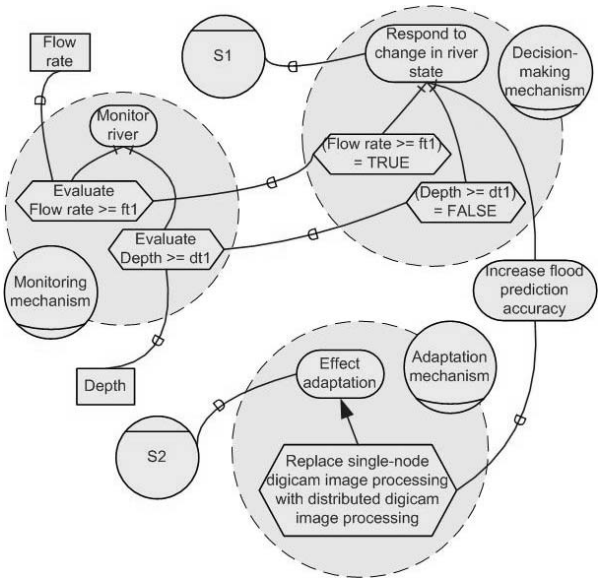

Figure 7. Level 2: S1 to S2 Adaptation Model

from adverse situations, such as flooding. The GridStix case study provided evidence that the separation of concerns imposed by [2] and the process model and goal-based modeling constructs of LoREM do provide a useful means to model the requirements of a DAS. The ability to reason about the non-functional goal trade-offs was particularly helpful to the GridStix developers because, in GridStix, these trade-offs defined the target systems and the adaptation scenarios. However, since we collaborated with the GridStix designers to apply LoREM we have not yet validated its unsupervised usability by other practitioners.

6. Related Work

While numerous techniques have emerged to support DAS development, we limit our discussion here to goal-oriented approaches to modeling DAS requirements. In general, these approaches [7, 6, 22, 14, 23] address Level 1 of the LoREM. Specifically, Fickas, Feather, *et al.* [7, 6, 4] model DAS steady-state system behaviors using the KAOS specification language [5] and leverage these models to generate run time monitoring mechanisms. Yu *et al.* [22, 14, 23] model DAS steady-state system behavior using a requirements goal model, a hybrid goal-oriented modeling technique using concepts and notations from KAOS and *i** [21]. Additionally, they provide a process for annotating these specifications in order to infer design information. All of these approaches differ from our approach in that they do not offer process models and focus only on specifying the steady-state systems at Level 1 RE, but do not specify the adaptation scenarios (at Level 2 RE) or the adaptation infrastructure (at Level 3 RE).

7. Conclusions

In this paper, we propose the LoREM, where each level describes the RE activities of a different DAS developer (i.e., the system developer, the adaptation scenario developer, the adaptation infrastructure developer, and the DAS research community) to construct *i** goal model(s) of their requirements. We leverage the LoREM to propose two processes for performing RE for a DAS, where each process step corresponds to the activities performed for one level. The order of the levels is dependent upon the maturity of the adaptation infrastructure. Thus, the application-driven process assumes a mature set of adaptation mechanisms; whereas, the technology-driven process assumes a less mature set of adaptation mechanisms, which constrains the range of possible adaptive behavior.

We illustrated the technology-driven process in the context of the GridStix case study, an adaptive flood warning system. There were three key benefits to this approach. First, the LoREM separated the concerns of the different types of DAS developers into different levels. This separation assisted the developers in more thoroughly understanding their role in the context of GridStix development and also the dependencies that exist among their tasks, requirements, and models. Second, the *i** goal models produced by performing LoREM activities could be integrated into the model-driven development of GridStix. Specifically, the behavior models and adaptation scenario models could be refined to construct design models. Third, specifying the requirements of the GridStix business logic at Levels 1 and 2 and the requirements of the GridStix adaptation infrastructure, GridKit, at Level 3 supported model reuse. Specifically, the GridStix developers could reuse the objectives, behavior, and adaptation scenario models in conjunction with a new adaptation infrastructure model that describes a different adaptation infrastructure selection, such as DynamicTAO [13], UIC [16], or RAPIDware [18].

There are several possible directions for future work. First, we are interested in addressing some current limitations of our approach. More research is needed to evaluate if DASs could perform RE at runtime, as proposed in [2], thus alleviating the need to construct adaptation scenario models. Currently, each target system and adaptation model must be constructed by hand. There is a need for new techniques to help system developers identify and generate models of candidate target systems that handle the known, potentially adverse conditions and are sufficiently resilient and robust to handle variations of the adverse conditions. Recent work at Michigan State University is exploring how biologically-inspired techniques can be used to generate models for target systems more resilient than human-generated behavioral models [9]. Additionally, we are interested in performing additional studies that compare our approach to other requirement specification techniques.

Second, we are exploring the connection between the goal models developed by applying one of the LoREM process models and the design models [19]. The approach offers the potential to achieve strong traceability through the different models, thus helping ensure that the resultant behavior is consistent with the requirements. Third, there is a nascent research community [1, 8] that is exploring the potential for models to drive automatic system reconfiguration at run time. Our work, and the opportunities it illustrates for mapping from system goals to run time adaptation mechanisms, serves as an early demonstration that run time model-driven engineering is feasible. Our vision is to make requirements the drivers of these run-time models.

Acknowledgements

The authors are grateful to Dan Berry for his insightful comments on an earlier draft of this work.

References

[1] N. Bencomo, G. Blair, and R. France. Summary of the workshop models@run.time at models 2006. In *Lecture Notes in Computer Science, Satellite Events at the MoDELS 2006 Conference: Springer-Verlag*, 2006.

[2] D. M. Berry, B. H. C. Cheng, and J. Zhang. The four levels of requirements engineering for and in dynamic adaptive systems. In *11th International Workshop on Requirements Engineering Foundation for Software Quality (REFSQ)*, 2005.

[3] G. Blair, G. Coulson, J. Ueyama, K. Lee, and A. Joolia. Opencom v2: A component model for building systems software. In *IASTED Software Engineering and Applications*, USA, 2004.

[4] D. Cohen, M. S. Feather, K. Narayanaswamy, and S. S. Fickas. Automatic monitoring of software requirements. In *ICSE '97: Proceedings of the 19th International Conference on Software Engineering*, 1997.

[5] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. In *IWSSD: Selected Papers of the Sixth International Workshop on Software Specification and Design*, pages 3–50, 1993.

[6] M. S. Feather, S. Fickas, A. V. Lamsweerde, and C. Ponsard. Reconciling system requirements and runtime behavior. In *IWSSD '98: Proceedings of the 9th International Workshop on Software Specification and Design*, 1998.

[7] S. Fickas and M. S. Feather. Requirements monitoring in dynamic environments. In *RE '95: Proceedings of the Second IEEE International Symposium on Requirements Engineering*, 1995.

[8] R. France and B. Rumpe. Model-driven development of complex software: A research roadmap. In L. Briand and E. A. Wolf, editors, *Future of Software Engineering*. IEEE-CS Press, 2007.

[9] H. J. Goldsby, D. B. Knoester, B. H. C. Cheng, P. K. McKinley, and C. A. Ofria. Digitally evolving models for dynamically adaptive systems. In *Proceedings of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, Minneapolis, MN, May 2007.

[10] P. Grace, G. Coulson, G. Blair, L. Mathy, D. Duce, C. Cooper, W. K. Yeung, and W. Cai. Gridkit: Pluggable overlay networks for grid computing. In *Proceedings of the Symposium on Distributed Objects and Applications (DOA)*, Cyprus, 2004.

[11] P. Grace, G. Coulson, G. Blair, and B. Porter. Deep middleware for the divergent grid. In *Proceedings of the 6th Annual IFIP/ACM/USENIX Middleware Conference*, Grenoble, France, 2005.

[12] D. Hughes, P. Greenwood, G. Coulson, G. Blair, F. Pappenberger, P. Smith, and K. Beven. An intelligent and adaptable flood monitoring and warning system. In *Proceedings of the 5th UK E-Science All Hands Meeting (AHM) (Best Paper Award)*, 2006.

[13] F. Kon, M. Roman, P. Liu, J. Mao, T. Yamane, L. Magalhaes, and R. Campbell. Monitoring, security, and dynamic configuration with the dynamictao reflective orb, 2000.

[14] A. Lapouchnian, S. Liaskos, J. Mylopoulos, and Y. Yu. Towards requirements-driven autonomic systems design. In *DEAS '05: Proceedings of the 2005 Workshop on Design and Evolution of Autonomic Application Software*, 2005.

[15] R. R. Lutz. Targeting safety-related errors during software requirements analysis. In *SIGSOFT '93: Proceedings of the 1st ACM SIGSOFT Symposium on Foundations of Software Engineering*, 1993.

[16] F. K. M. Roman and R. H. Campbell. Reflective middleware: From the desk to your hand. In *IEEE DS Online, Special Issue on Reflective Middleware, vol. 2*, 2001.

[17] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng. Composing adaptive software. *Computer*, 37(7), 2004.

[18] S. Sadjadi, P. McKinley, and E. Kasten. Architecture and operation of an adaptable communication substrate. In *Proceedings of the Ninth IEEE International Workshop on Future Trends in Distributed Computing*, San Juan, Puerto Rico, 2003.

[19] P. Sawyer, N. Bencomo, D. Hughes, P. Grace, H. J. Goldsby, and B. H. C. Cheng. Visualizing the analysis of dynamically adaptive systems using i* and dsls. In *Proceedings of the RE Workshop on Requirements Engineering Visualization (REV 2007)*, New Dehli, India, October 2007.

[20] V. M. Systems. Dynamic adaptive radiotherapy. http://www.varian.com/orad/drt000.html.

[21] E. S. K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *RE '97: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, Washington, DC, USA, 1997.

[22] Y. Yu, J. C. S. do Prado Leite, and J. Mylopoulos. From goals to aspects: Discovering aspects from requirements goal models. In *Proceedings of the 12th IEEE International Conference on Requirements Engineering (RE 2004)*, 2004.

[23] Y. Yu, J. Mylopoulos, A. Lapouchnian, S. Liaskos, and J. C. Leite. From stakeholder goals to high-variability software design. Technical report csrg-509, University of Toronto, 2005.