Relaxing Claims: Coping With Uncertainty While Evaluating Assumptions at Run Time

Andres J. Ramirez[1], Betty H.C. Cheng[1], Nelly Bencomo[2], and Pete Sawyer[2,3]

[1] Michigan State University, East Lansing, Michigan 48823
{ramir105, chengb}@cse.msu.edu
[2] INRIA Paris - Rocquencourt, 78153 Le Chesnay, France
nelly@acm.org
[3] Lancaster University, InfoLab 21, Lancaster, UK LA12WA
sawyer@comp.lancs.ac.uk

Abstract. Self-adaptation enables software systems to respond to changing environmental contexts that may not be fully understood at design time. Designing a dynamically adaptive system (DAS) to cope with this uncertainty is challenging, as it is impractical during requirements analysis and design time to anticipate every environmental condition that the DAS may encounter. Previously, the RELAX language was proposed to make requirements more tolerant to environmental uncertainty, and Claims were applied as markers of uncertainty that document how design assumptions affect goals. This paper integrates these two techniques in order to assess the validity of Claims at run time while tolerating minor and unanticipated environmental conditions that can trigger adaptations. We apply the proposed approach to the dynamic reconfiguration of a remote data mirroring network that must diffuse data while minimizing costs and exposure to data loss. Results show RELAXing Claims enables a DAS to reduce adaptation costs.

1 Introduction

Dynamically adaptive systems (DASs) are systems built to continuously monitor their environment and then adapt their behavior according to changing environmental conditions. [4]. The inherent uncertainty associated with the operational environment of a DAS is challenging as it is often impractical to anticipate every environmental condition that a DAS will encounter throughout its lifetime [4, 30]. Previously, the RELAX [30] requirements specification language was proposed to make requirements more tolerant to environmental uncertainty, and Claims [28, 29] were applied as markers of uncertainty to record the rationale for a decision made with incomplete information in a DAS. This paper integrates both techniques to assess the validity of Claims at run time while tolerating minor and unanticipated environmental conditions that can otherwise trigger adaptations.

Certain properties about the DAS or its execution environment might not be known until run time. This uncertainty forces developers to make assumptions about the design or configuration of the system. A Claim [28, 29] can be used at

design time to document and analyze assumptions about how a DAS achieves its goals in different operational contexts. For example, a Claim can be used to document that data should be encrypted since a network might not be secure. A Claim can also be monitored at run time to prove or disprove its validity [28], thereby triggering an adaptation to reach more desirable system configurations if necessary. Nevertheless, Claims are also subject to uncertainty, in the form of unanticipated environmental conditions and unreliable monitoring information, that can adversely affect the behavior of the DAS if it spuriously falsifies a Claim.

This paper proposes the RELAXation of a Claim in recognition that environmental uncertainty may prevent a DAS from categorically proving or disproving a Claim at run time. Specifically, our approach uses RELAX operators to introduce a fuzzy logic layer upon the evaluation criteria that establishes a Claim's validity. Ideally, RELAXing a Claim should enable a DAS to tolerate environmental uncertainty that may otherwise mistakenly disprove a Claim. In this manner, RELAXing a Claim may also reduce adaptation costs for a DAS by preventing frequent, and perhaps unnecessary, adaptations and reconfigurations of its goal realization strategies at run time.

In this paper we propose a stepwise process for RELAXing Claims. First, sources of uncertainty that can disprove the validity of a Claim must be identified. Next, a Claim *applicability metric* must be derived to compute the veracity of a Claim at run time. Both ordinal and temporal RELAX operators can be applied to relax the constraints that define this applicability metric. At run time, if the value produced by a RELAXed Claim applicability metric drops below a predetermined threshold, then the value of the corresponding contribution link must be updated and, if necessary, the system may have to reconfigure towards a different goal realization strategy depending on the current set of valid Claims.

We assess the effectiveness of RELAXing Claims by applying the proposed approach to an industry-provided remote data mirroring network [11, 12]. Remote mirroring is a technique that improves data protection and availability by replicating and distributing data to all nodes within a network, such as that used for cable television servers. We RELAX Claims that captures sources of uncertainty that can trigger network reconfigurations in response to adverse conditions, such as network link failures and dropped network messages. The remainder of this paper is organized as follows. Section 2 provides background information on remote data mirroring, goal-oriented requirements modeling, RELAX, and Claims. Section 3 presents the proposed approach for RELAXing Claims. Next, Section 4 presents experimental results. Section 5 describes related work. Lastly, Section 6 summarizes results and overviews future directions.

2 Background

This section provides background information on remote data mirroring, goal modeling, the RELAX requirements specification language, and Claims.

2.1 Remote Data Mirroring

Remote data mirroring (RDM) [11, 12] is a data protection technique that stores copies of data at physically isolated locations to protect data against loss, unavailability, or corruption. An RDM can be configured in terms of the network's topology, such as a minimum spanning tree, as well as how data is distributed among data servers. For instance, synchronous propagation distributes data whenever it is modified. In contrast, asynchronous propagation batches data modifications, ideally enabling multiple edits to the same data to be coalesced. Each configuration provides different levels of data protection, performance, and cost. That is, while synchronous propagation provides better data protection than asynchronous propagation, it incurs a network performance penalty as every change must be distributed across the network. Asynchronous propagation provides better network performance than synchronous propagation but it also provides a weaker form of data protection because batched data could be lost in the event of a site failure.

2.2 Goal-oriented requirements engineering and modeling

A goal declaratively specifies the objectives and constraints that a system-to-be and its execution environment must satisfy [14]. A key objective in goal-oriented requirements engineering (GORE) is to systematically elicit, analyze, and refine high-level goals into finer-grained goals. While goals that represent required functional properties can be evaluated in an absolute manner, a special category of goals called soft goals can only be *satisficed* [5] or satisfied to a certain degree. Soft goals typically represent non-functional properties (e.g., performance) that constrain how functionality should be delivered to stakeholders.

A goal-oriented requirements model provides a graphical framework for capturing relationships between goals. Formally, a goal-oriented requirements model is a directed acyclic graph where a node represents a goal and an edge represents a specific type of refinement. For instance, the i* framework [31] provides an agent-oriented approach for modeling strategies of multiple actors within social contexts. In i*, a Strategic Rational (SR) model captures how a system's configuration addresses the interests and concerns of an actor.

The i* SR goal model in Figure 1 captures the following soft goals for the RDM application: "Minimize Operational Expenses", "Maximize Data Reliability", and "Maximize Network Performance". In order to satisfice these soft goals, the RDM must achieve functional goals such as constructing a connected network and distributing data. These functional goals can be achieved through alternative goal realization strategies (modeled in i* as tasks) that include constructing different network topologies, such as a minimum spanning tree or a redundant topology, and changing propagation parameters.

2.3 RELAX specification language

RELAX is a specification language that explicitly addresses uncertainty in adaptive systems. In particular, RELAX was developed to identify and declaratively

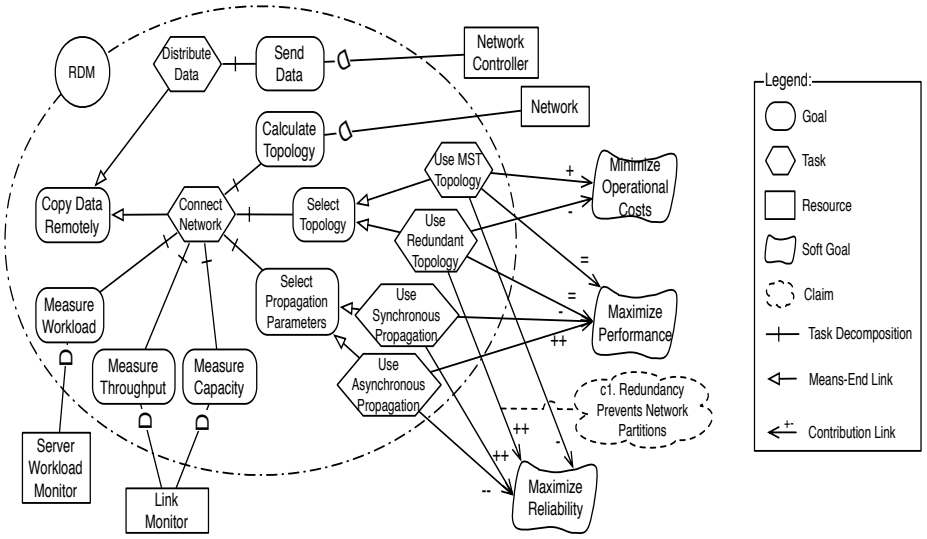

Fig. 1. i* SR goal model for the remote data mirroring application.

specify sources of uncertainty that occur at the shared boundary between the system-to-be and its environment [10] . Sources of uncertainty are specified using ENV, MON, and REL elements. ENV specifies properties about the operating context of the DAS, MON lists sensors in the monitoring infrastructure of the DAS that can directly observe and contribute information towards determining the values of ENV properties, and REL defines relationships for computing ENV properties from MON elements.

The RELAX language uses fuzzy logic to express uncertainty in requirements and, by definition, enable developers to systematically design systems that are more flexible and amenable to adaptation. To this end, RELAX provides both ordinal and temporal operators to add flexibility in how and when a functionality may be delivered, respectively. For example, a key requirement in the RDM application states that "The system *shall* distribute new data throughout the network". Environmental uncertainty, in the form of unpredictable link failures and dropped network messages may temporarily hinder the satisfaction of this requirement. RELAXing this requirement to specify that "The system *shall* distribute new data throughout the network AS EARLY AS POSSIBLE" provides temporal flexibility to account for unanticipated events while distributing data. Other RELAX operators include AS LATE AS POSSIBLE, AS CLOSE AS POSSIBLE TO, AS MANY AS POSSIBLE, and AS FEW AS POSSIBLE.

2.4 Claims

Claims were introduced by the NFR framework [5] where they were used to record decision rationale. This role was extended in REAssuRE (REcording of

Assumptions in Requirements Engineering) [29] where Claims are used as *markers of uncertainty* to record the rationale for a decision made with incomplete information in a DAS. REAssuRE showed that Claims are useful where dynamic adaptation is used to maximize the satisficement of a system's soft goals by dynamically selecting between alternative goal operationalizations. The extent to which alternative operationalizations satisfice the system's soft goals under every context cannot always be predicted by the requirements engineer, and Claims serve to record this uncertainty. At run time, Claims can be monitored to test their veracity. If a Claim holds, then the predicted impact of the operationalization on the soft goal is assumed to hold too. If the Claim is falsified by evidence collected by run-time monitoring then the predicted impact of the operationalization on the soft goal is considered unsound and the optimal configuration of operationalizations must be re-evaluated.

In i*, Claims are attached to contribution links, whose values represent the predicted degree of satisficement provided by a task at one end of the link and a soft goal on the other end. In Figure 1, Claim *c1* asserts that the operationalization strategy "Use Redundant Topology" has a strongly positive (++) effect on satisficement of the soft goal "Maximize Reliability". If a Claim proves to be false, such as if RDM nodes were vulnerable to common-cause failures, then the value of the contribution link to which it is attached should be revised to reflect the observed reality. When this happens, the goal model needs to be updated and dynamically re-evaluated. A reconfiguration is triggered if re-evaluation determines that an alternative goal operationalization exists that has a greater predicted net impact on the soft goals.

This automatic determination of the best operationalization is as follows:

Let the function *satisfices* represent the contribution value for a task, soft goal pair:

$$satisfices: \ T \times SG \rightarrow C$$

where T is the set of tasks, SG is the set of soft goals, and C is the set of possible contribution values {*- -, -, =, +, ++*}. These as are interpreted as corresponding to the range of integer values {*-2, -1, 0, 1, 2*}, respectively. Moreover, i is an index in the set of tasks that represent alternative operationalization of goal g, and t_{ig} is thus one of these tasks.

The task selected as the operationalization strategy for goal g is the one with the net greatest value of contribution link values for all of the soft goals it influences. This is given by the following weighted sum formula in which w represents the relative priority of each softgoal, given as a numeric weight. Note that for simplicity in the remote data monitoring example, we treat each softgoal as being of equal priority, so $w = 1$ in all cases.

$$max_i \sum_{sg \in SG} w_{sg} satisfices(t_{ig}, sg) \tag{1}$$

In the RDM example, if falsification of *c1* meant that the contribution link value linking "Use Redundant Topology" and "Maximize Reliability" reduced to neutral (=), then taking the aggregate of the contribution link values of "Use

Redundant Topology" and its alternative "Use MST Topology" over the three soft goals, the latter would emerge as the better solution because its set of contribution link values (+,=,-) has a higher net contribution value than that of "Use Redundant Topology" (-,=,=). This evaluation would trigger an adaptation to replace the network configuration that implements "Use Redundant Topology" with "Use MST Topology".

3 Approach for RELAXing Claims

Next we describe how to RELAX a Claim and illustrates with an example.

3.1 Motivation for RELAXing Claims

Figure 2 shows a Claim refinement model that captures assumptions for why a redundant network topology contributes positively to the "Maximize Reliability" soft goal in Figure 1. Although the underlying assumptions might seem reasonable, they are subject to system and environmental uncertainty. In particular, Claims *c4* and *c5* state that link faults do not partition the network nor do they occur coincidentally, respectively. Thus, if two or more network links fail simultaneously then top-level Claim *c1* becomes automatically falsified.

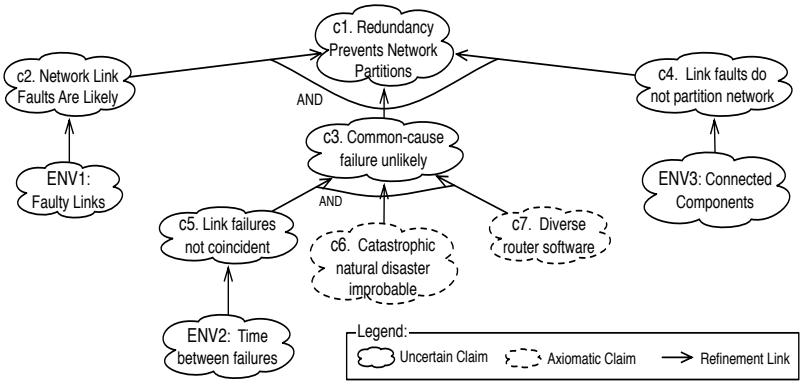

Fig. 2. A Claim refinement model describing why redundancy improves reliability in RDM application.

In this scenario, it is *possible* for multiple network link failures to occur simultaneously while the network remains connected. Although a redundant network topology prevents the network from becoming disconnected in this scenario, the top-level Claim *c1* would become disproven by the simultaneous failure of two or more network links. The objective of introducing RELAX operators into the specification of a Claim is to lessen the thresholds or bounds that define the veracity of the Claim itself. Ideally, RELAXing a Claim prevents transient and unanticipated environmental conditions from unnecessarily disproving the validity of a Claim at run time and thus triggering consequential adaptations.

3.2 Process for RELAXing Claims

Figure 3 presents a data flow diagram that overviews the Claim RELAXation process. We describe each step in the Claim RELAXation process in detail:

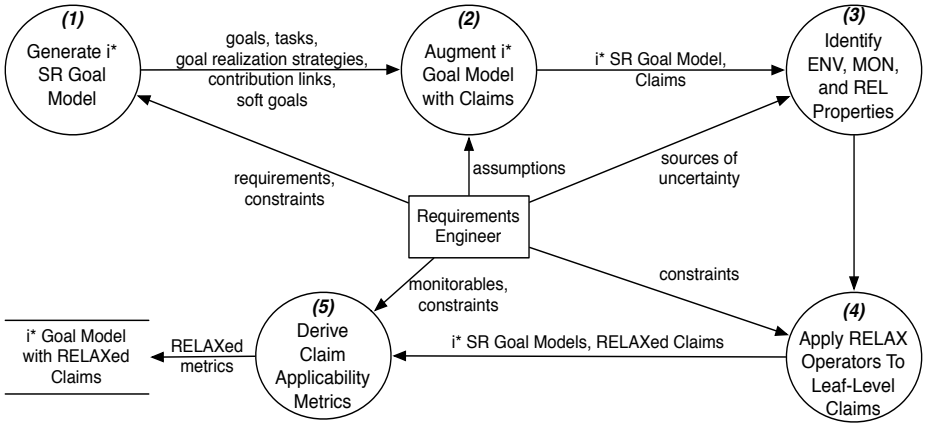

Fig. 3. Data flow diagram describing process for RELAXing Claims.

(1) Generate i* SR Goal Model. Given a set of requirements and constraints, a requirements engineer must generate an i* SR goal model to capture the goals, tasks, resources, and soft goals of the system-to-be. It must also specify how the alternative goal realization strategies affect the contribution links of soft goals. As an example, the i* SR goal model in Figure 1 captures the various goal realization strategies that the RDM application can use to replicate data across the network, as well as how these strategies affect each soft goal.

(2) Augment i* Goal Model with Claims. The i* SR goal model generated in (1) must be augmented with Claims [29] to document uncertain assumptions about how goal realization strategies affect the contribution links of soft goals. To this end, a Claim refinement model can be used to specify a set of assumptions that collectively support the veracity of a top-level Claim. In a Claim refinement model, certain low-level Claims can be considered axiomatic and need not be monitored at run time. In contrast, evidence for or against the assumption of an uncertain Claim has to be collected at run time [21].[1]

Figure 2 presents a Claim refinement model for the RDM application. The top level Claim in this model, *c1* states that redundancy prevents network partitions, and thus a redundant topology helps the "Maximize Reliability" soft goal. The validity of this top-level Claim is based on the validity of three sub Claims. Namely, redundancy prevents network partitions because, while network

[1] As an optional step, a requirements engineer can specify how the value of a contribution link should be updated if an attached Claim is falsified, otherwise the value will default to neutral ("=").

link faults are likely (*c2*), common cause failures are unlikely (*c3*), and network link faults do not partition the network (*c4*). Similarly, Claim *c3* is refined into additional assumptions that collectively state that common-cause link failures are unlikely because link failures tend to not coincide (*c5*), catastrophic natural disasters are improbable (*c6*), and routers use diverse software systems (*c7*). At run time, Claims *c2*, *c4*, and *c5* must be monitored to prove or disprove their validity. In contrast, Claims *c6* and *c7* are considered axiomatic and need not be monitored.

(3) Identify ENV, MON, and REL Properties. A requirements engineer must identify ENV, MON, and REL properties for each leaf-level non-axiomatic Claim in the Claim refinement model. Furthermore, each ENV property must be mapped to its corresponding MON elements. Specifically, an ENV property that a DAS can directly observe with its sensors can be expressed solely by MON elements. In contrast, an ENV property that a DAS cannot directly observe with its sensors must be indirectly inferred via an REL relationship that might comprise MON elements, as well as constraints and algorithms.

Figure 2 presents three ENV properties associated with leaf-level Claims in the Claim refinement model. Within these leaf-level Claims, *ENV1* refers to the number of faulty links in the RDM network, *ENV2* measures the time between any two link faults, and *ENV3* captures whether the network is connected or partitioned. Since *ENV1* is a property that can be directly observed by LinkMonitor sensors, it can be evaluated as follows: $ENV1 = \sum_{i=1}^{|\text{Links}|} \text{faulty(i)}$, where |Links| is the number of links in the RDM network, and faulty(i) returns 1 if the i^{th} link has failed and 0 otherwise. *ENV3*, on the other hand, cannot be directly observed by sensors and must instead be computed algorithmically by aggregating information from available MON elements. In particular, *ENV3* represents the number of partitions in the RDM network that must be computed with a reachability algorithm [25] that examines which RDM nodes can be reached by traversing active network links.

(4) Apply RELAX Operators to Leaf-Level Claims. A requirements engineer must identify sources of uncertainty that can affect ENV properties associated with each non-axiomatic leaf-level Claim in a Claim refinement model. In addition, a requirements engineer must also determine which of these ENV properties can be safely RELAXed without affecting the satisfaction of invariant goals and requirements. If an ENV property can temporarily deviate from its expected value, then a corresponding RELAX operator must be applied to specify how its value can vary due to environmental uncertainty. Once a RELAX operator is applied, bounds must be established to constrain the extent to which the ENV property can vary without unnecessarily disproving a Claim.

As an example, consider that while Claim *c2* states that network link failures are common, unreliable monitoring information that fails to detect link failures can disprove this Claim. Nevertheless, the validity of Claim *c1* should not be necessarily disproven because no link failures are observed. As such, Claim *c2* can be RELAXed to state that it SHALL hold when UNTIL n units of time have passed without link failures. Likewise, Claim *c4* can be RELAXed to state that

it SHALL hold when the number of connected components in the network is AS CLOSE AS POSSIBLE TO one when network link failures occur. Lastly, Claim *c5* can be RELAXed to state that it SHALL hold when network links fail at the same time AS CLOSE AS POSSIBLE TO never.

(5) Derive Claim Applicability Metrics. A requirements engineer must derive a *Claim applicability metric* for each non-axiomatic leaf-level Claim in the Claim refinement model. Specifically, a Claim applicability metric uses fuzzy logic functions (i.e., triangle, trapezoid) to evaluate monitoring information and assess the validity of a Claim at run time. The peak of a Claim applicability metric function represents the ideal case (i.e., when the assessment is solidly true), and the respective tails are still acceptable but not optimal. As input, each Claim applicability metric accepts monitoring information from sensors in MON. If necessary, MON values may need to be mapped to ENV properties through REL relationships. As output, a Claim applicability metric generates a numerical value, between zero and one, that is proportional to the Claim's validity.

Figure 4 shows three RELAXed Claim applicability metrics that can be used to evaluate the validity of Claims *c2*, *c4*, and *c5*. As this figure illustrates, the applicability metric for Claim *c2* returns values close to one as the number of faulty links increases, thereby validating the assumption that network link faults are likely to occur. Similarly, the applicability metric for Claim *c4* returns one when the number of connected components in the RDM network equals the desired number of connected components. Since a connected network has exactly one connected component, then this applicability metric serves to validate the assumption that link failures do not partition the network. Lastly, the Claim applicability metric for Claim *c5* returns values close to one as the time between link failures increases, thereby validating the assumption that network link faults are not coincidental.

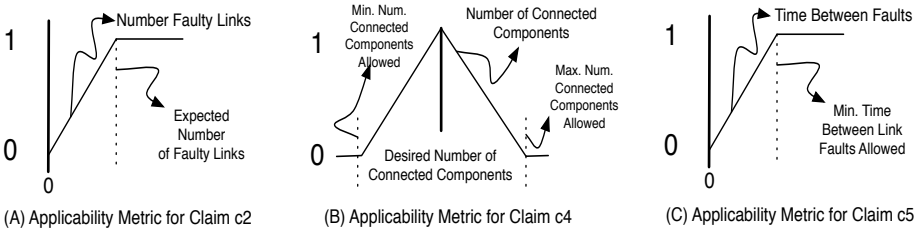

Fig. 4. RELAXed Claim applicability metrics for Claims c2, c4, and c5.

Since each Claim applicability metric is associated with a leaf-level Claim, its value must be propagated upwards through the Claim refinement model in order to compute the validity of the top-level Claim. Depending on the type of the Claim refinement link, two methods can be used to compute the applicability value of a parent Claim. While the applicability value of an AND-refined Claim is equal to the minimum value of each sub-Claim, the applicability value of an OR-

refined Claim is equal to the maximum value of each sub-Claim. As an example, assume that the applicability values of Claims *c2*, *c3*, and *c4* in Figure 2 are equal to 0.92, 0.89, and 0.98 respectively. In this scenario, the applicability value of top-level Claim *c1* equals 0.89, the minimum value of all AND-refined Claims.

3.3 Adapting in Response to Falsified Claims

A RELAXed Claim is falsified when its Claim applicability value drops below a predetermined threshold. When a Claim is falsified, the value of the contribution link to which it is attached must be updated to indicate that a given goal realization strategy does not necessarily address current environmental conditions. While the value of a contribution link could be inverted (e.g., converting a "+" to a "-"), this approach would correspond to an arbitrary transformation that might not hold either. In the absence of new evidence, a DAS cannot assume anything about the value of the contribution link except that the original value is probably wrong. Thus, the value of a contribution link of a falsified Claim becomes neutral and undefined, regardless of its original polarity or strength. Alternatively, a requirements engineer can override these default values on a per-Claim basis if desired (see footnote in Section 3.2, Step (2)).

Once the value of a contribution link is updated, the DAS must re-evaluate the entire set of goal realization strategies to determine whether it should self-reconfigure towards a more suitable goal realization strategy. In this manner, run-time updates to the goal model drives the self-reconfiguration.

4 Experimental Results

This section describes experimental results for demonstrating the utility of RELAXing Claims.

4.1 Experimental Setup

In the following experiments, each RDM must replicate and distribute new data to all other RDMs. We modeled and implemented an RDM network as a completely connected undirected graph where each node and edge represents an RDM or a network link, respectively. The network itself comprises 25 RDMs and 300 network links that can be activated to distribute data. The operational characteristics of each RDM node and network link, such as workload or capacity, were generated from a random uniform distribution based on RDM operational models previously presented by Keeton *et al.* [11, 12]. Throughout the simulation, approximately 25 new data items were randomly inserted at different RDMs from which they had to be efficiently distributed across the network.

The RDM network might self-adapt at run time in response to adverse environmental conditions, such as link failures, repeatedly dropped messages, or unreliable monitoring data. To this end, each RDM implements the dynamic change management (DCM) protocol introduced by Kramer and Magee [13] such

that it may reach passive and quiescent states in bounded time. An RDM in a passive state can accept data distributed by other RDMs but may not replicate nor distribute data itself. Similarly, a quiescent RDM can neither distribute nor accept data from other RDMs since both its neighbors and itself are in passive states.

In addition, we implemented a rule-based adaptation engine that leverages Claims constructs as conditionals for adaptation. Specifically, the applicability of each Claim is monitored and evaluated at run time. If a Claim becomes falsified, then the value of its contribution link is updated to reflect this new information. Next, the set of goal realization strategies are re-evaluated to select the one that best addresses soft goals given current system and environmental conditions (see Section 2.4). If an adaptation is required, then a target configuration is selected and the DCM protocol is executed to generate an adaptation path that safely transitions the executing RDM network from its source to its target configuration.

The first experiment explores how a DAS can leverage RELAXed Claims to trigger adaptations at run time, and the second experiment explores how RELAXing Claims can reduce adaptation costs in uncertain environments. For statistical purposes, we conducted 40 trials of each experiment. Where applicable, we plot the mean values with corresponding error bars.

4.2 No Environmental Uncertainty

This experiment evaluates how a falsified RELAXed Claim can trigger the runtime reconfiguration of a DAS. For this scenario, the values of the contribution links in Figure 1 suggest that the best goal realization strategy is to use a redundant network topology with synchronous propagation. As such, Figure 5(A) depicts the initial configuration comprising 25 RDMs using synchronous propagation and 32 active network links, 8 of them redundant. This configuration is based on the validity of Claim *c1* (see Figure 2) that states that a redundant network topology prevents network link failures from partitioning the network. Nevertheless, for this scenario we disabled adverse environmental conditions to purposefully cause the falsification of this Claim and thereby enable us to evaluate how the RDM network self-reconfigures in response.

Since network failures are not possible in this scenario, the applicability of Claim *c2*, which states that network link faults are likely, gradually decreases until it falsifies top-level Claim *c1*. The falsification of Claim *c1* states that, for the given set of system and environmental conditions, the "Use Redundant Topology" goal realization strategy does not necessarily prevent network partitions and therefore does not contribute as positively ("++") to the "Maximize Reliability" soft goal. As a result, the value of the corresponding contribution link in Figure 5(B) changes from "++" to "=" to reflect reduced confidence that a network topology contributes positively to the "Maximize Reliability" soft goal.

Once the value of the contribution link is updated, the DAS re-evaluates its goal operationalizations. As Figures 1 and 5(B) illustrate, once Claim *c1* is falsified, the best goal realization strategy becomes to use a minimum spanning

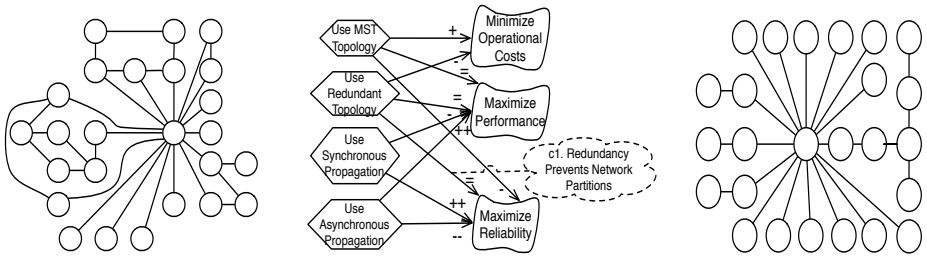

Fig. 5. Adaptation progression in response to falsified Claim c1.

tree (MST) topology with synchronous propagation. The reconfigured RDM network, shown in Figure 5(C) comprises 25 RDMs propagating data synchronously through 24 active network links. Combined, a MST minimizes operational costs while synchronous propagation maximizes network reliability. Note that while asynchronous propagation would also maximize performance, it is not the optimal solution for this scenario since its advantages are offset by contributing equally negatively against the "Maximize Reliability" soft goal.

4.3 Environmental Uncertainty

This experiment compares the benefits, in terms of adaptation costs, that RELAXed Claims can provide over traditional Claims in uncertain environments. For this work, we define adaptation costs based on the adaptation quality property of *settling time et al.* [26]. Specifically, we measure adaptation costs by counting the number of components placed in passive and quiescent states during adaptation. Thus, the null hypothesis, H_0, states that there is no difference in adaptation costs between RELAXed Claims and traditional Claims. Similarly, the alternative hypothesis, H_1, states that RELAXed Claims will incur lower adaptation costs than traditional Claims. The rationale for H_1 is that a RELAXed Claim is more tolerant to uncertain environmental conditions that might otherwise disprove a traditional Claim and thus trigger an unnecessary adaptation.

As in the previous experiment, the RDM is initially configured to use a redundant network topology and synchronous propagation. However, while the previous experiment did not introduce any forms of environmental uncertainty into the RDM, we now randomly kill network links, drop messages, and either delay or introduce noise into monitoring information. These types of environmental conditions are intended to affect the applicability of Claims *c3* and *c4*, as well as the applicability of top-level Claim *c1*.

Figure 6 presents a bar chart with the various adaptation cost metrics tracked throughout each experimental simulation, where error bars denote statistical significance where applicable. Specifically, the first two bars count the mean number of adaptations performed in each trial. The remaining bars count the mean number of components in passive and quiescent mode throughout these adaptations, respectively. As this plot illustrates, RELAXing Claims significantly reduces the

number of adaptations triggered by environmental uncertainty, and also significantly reduces the number of components placed in passive mode during adaptations ($p < 0.05$). Although statistically insignificant, this plot also illustrates a decreasing trend in the number of components that are placed in quiescent mode during adaptations, thus minimizing the impact on system functionality even in the face of environmental uncertainty.

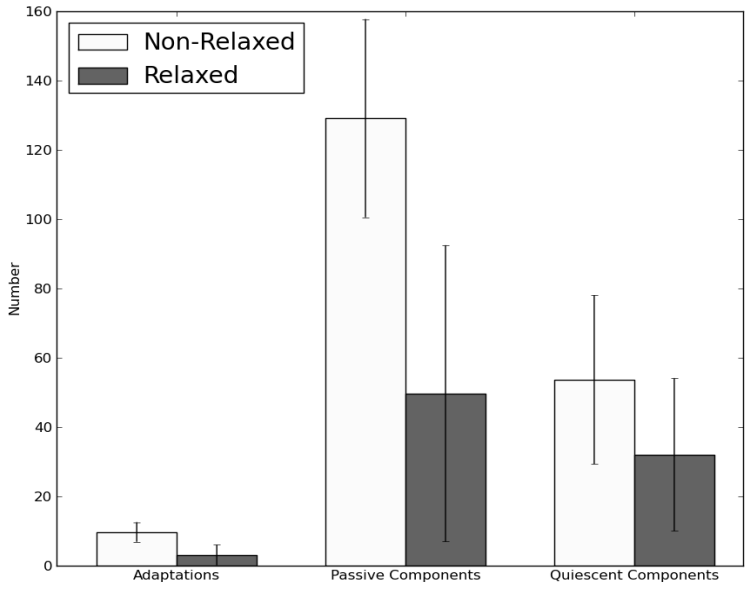

Fig. 6. Comparing adaptation costs between RELAXed Claims and traditional Claims.

Given the differences in adaptation costs between RELAXed Claims and traditional Claims we reject the null hypothesis H_0. Likewise, we accept the alternate hypothesis, H_1, as it concerns both the total number of adaptations performed and the number of components placed in passive mode during adaptations. These differences in adaptation costs are caused primarily by the falsification of Claim *c5* (see Figure 2). While coincident link failures automatically falsify non-RELAXed Claim *c5*, in most cases it did not disprove its RELAXed Claim counterpart as long as the network remained connected. As such, RELAXing Claims enable the RDM network to reduce the number of adaptations it incurred in response to adverse environmental conditions.

5 Related Work

This section presents related work on model-based approaches to the requirements engineering of a DAS. In particular, this section includes related work on

documenting and analyzing obstacles and sources of uncertainty, requirements monitoring, and requirements reflection.

Documenting and Analyzing Obstacles and Sources of Uncertainty. A number of requirements-level techniques have been developed to deal with changing environmental conditions faced by a DAS. For instance, Letier and van Lamsweerde [15] introduced a goal modeling approach for identifying and resolving obstacles that can prevent the satisfaction of a goal. Obstacles are identified by negating requirements and then elaborating preconditions for the obstacles to arise. Letier and van Lamsweerde [17] also introduced a probabilistic framework for specifying the partial satisfaction of goals. While these approaches are intended for functional goals, conceptually their techniques could be applied to identify and evaluate conditions that can falsify assumptions.

Fuzzy logic has been used to represent effects of uncertainty on the satisfaction of goals and requirements. RELAX [4, 30] and FLAGS [2] are requirements specification languages for making more flexible the satisfaction criteria of functional goals to cope with uncertainty. RELAX can also be used in goal-modeling for identifying and mitigating sources of uncertainty. Serrano *et al.* [23] introduced an approach for RELAXing the contribution links of soft goals, thereby enabling a DAS to evaluate the satisficement [5] of soft goals while coping with uncertainty. None of these approaches, however, focus on how uncertainty can affect the validity of assumptions at run time – the focus of this paper.

Requirements Monitoring. Traditionally, requirements monitoring frameworks [7, 8, 20] use monitoring information to trace through state-based models that specify allowed states of the system. This information enables a DAS to identify and address conditions that can prevent the satisfaction of requirements. Utility functions have also been applied to monitor requirements [9, 19, 27]. Our approach is similar to these approaches in that it is intended to detect and respond to run-time conditions that prevent a DAS from satisfying its objectives. Instead of directly monitoring requirements, however, our approach monitors Claims to detect falsified assumptions that can obstruct goals. Both types of techniques can and should be used in conjunction at run time.

Requirements Reflection. Sawyer *et al.* [22] suggested that requirements should be run-time entities about which can be reasoned to determine their level of satisfaction and used to support adaptation decisions. Our approach adopts this view in that it uses Claims to reason about requirements at run time. GMoDS [6] also maintains run-time representations of goals, where a goal can transition between achieved, failed, obviated, or removed states. In contrast to our approach, GMoDS does not perform reasoning about goal satisfaction.

Another approach by Chen *et al.* [3] maintains run-time goal models to reason about tradeoff decisions aimed at achieving *survivability assurance*. As with our approach, their live goal models postpone the necessary quality tradeoff decisions until run time. Unlike our approach, however, their approach deals with functional goals, selectively allowing them to become disconnected from the goal model to spare resources. Awareness Requirements (AwReqs) [16, 24] refer to the success or failure of other requirements, providing information on run-time

divergence from specified requirements as a feedback loop in an explicitly control-theoretic model of requirements-driven dynamic adaptation. Although AwReqs do not focus on detecting run-time divergence from assumptions, exploring the management of requirements and assumptions at run time would be interesting.

Ali *et al.* [1] present a goal-based framework for modeling and analyzing contextual requirements. Their approach shares similarities with our use of Claims, where context plays a similar role to the domain assumptions that are represented with Claims. While context analysis is similar to Claim refinement, any connection between context and monitoring is implicit.

6 Conclusions

Recent research in RE has applied dynamic adaptation to mitigate uncertainty about the environmental contexts that a system-to-be may encounter at run time [2–4, 18, 28–30]. This paper has presented an approach to account for system and environmental uncertainty by RELAXing Claims when there is uncertainty about the evidence for or against a Claim's truth. We argue that failing to recognize both the fallibility of monitoring information, as well as the transient effects of minor environmental disturbances, are likely to cause erratic behavior in a DAS, and may risk unnecessary, and perhaps costly, adaptations. We evaluated our Claim RELAXation technique by applying it to an industry-provided case study of RDM system. Experimental results performed on an RDM simulator show that RELAXing Claims enables a DAS to reduce the number of adaptations when compared to traditional non-RELAXed Claims.

Future work will explore several open issues raised by our investigations. First, we will continue to explore how RELAX can be used to tolerate uncertainty from the environment and the monitoring infrastructure, as RELAX is applied to Claims and functional goals. Next, we are interested in making the propagation of a failed Claim to the goal model more flexible. Currently, a failed Claim results in loss of trust in the predicted degree of soft goal satisficement. Finally, we will explore how the RELAXation process can be more automated.

7 Acknowledgements

This work has been supported in part by NSF grants CCF-0541131, IIP-0700329, CCF-0750787, CCF-0820220, DBI-0939454, CNS-0854931, Army Research Office grant W911NF-08-1-0495, Ford Motor Company. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, Army, Ford, or other research sponsors.

This research is also partially supported by the Marie Curie Fellowship Requirements@run.time and the EU CONNECT project.

References

1. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requir. Eng. 15, 439–458 (2010)
2. Baresi, L., Pasquale, L., Spoletini, P.: Fuzzy goals for requirements-driven adaptation. In: Proceedings of the 18th IEEE International Requirements Engineering Conference. pp. 125–134. IEEE, Sydney, Australia (October 2010)
3. Chen, B., Peng, X., Yu, Y., Zhao, W.: Are your sites down? requirements-driven self-tuning for the survivability of web systems. In: 19th International Conference on Requirements Engineering (2011)
4. Cheng, B.H., Sawyer, P., Bencomo, N., Whittle, J.: A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS'09). pp. 468–483. Lecture Notes in Computer Science, Springer-Verlag, Denver, Colorado, USA (October 2009)
5. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers (2000)
6. DeLoach, S.A., Miller, M.: A goal model for adaptive complex systems. International Journal of Computational Intelligence: Theory and Practice. 5(2) (2010)
7. Feather, M.S., Fickas, S., van Lamsweerde, A., Ponsard, C.: Reconciling system requirements and runtime behavior. In: Proc. of the 8th Int. Workshop on Software Specification and Design. pp. 50–59. Washington, DC, USA (1998)
8. Fickas, S., Feather, M.S.: Requirements monitoring in dynamic environments. In: Proc. of the Second IEEE Int. Symp. on Requirements Eng. pp. 140–147. Washington, DC, USA (1995)
9. de Grandis, P., Valetto, G.: Elicitation and utilization of application-level utility functions. In: In the Proceedings of the Sixth International Conference on Autonomic Computing (ICAC'09). pp. 107–116. ACM, Barcelona, Spain (June 2009)
10. Jackson, M., Zave, P.: Deriving specifications from requirements: an example. In: Proceedings of the 17th International Conference on Software Engineering. pp. 15–24. ACM, Seattle, Washington, USA (April 1995)
11. Ji, M., Veitch, A., Wilkes, J.: Seneca: Remote mirroring done write. In: USENIX 2003 Annual Technical Conference. pp. 253–268. USENIX Association, Berkeley, CA, USA (June 2003)
12. Keeton, K., Santos, C., Beyer, D., Chase, J., Wilkes, J.: Designing for disasters. In: Proceedings of the 3rd USENIX Conference on File and Storage Technologies. pp. 59–62. USENIX Association, Berkeley, CA, USA (2004)
13. Kramer, J., Magee, J.: The evolving philosophers problem: Dynamic change management. IEEE Trans. on Soft. Eng. 16(11), 1293–1306 (1990)
14. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley (March 2009)
15. van Lamsweerde, A., Letier, E.: Handling obstacles in goal-oriented requirements engineering. IEEE Tran. on Soft. Eng. 26(10), 978–1005 (2000)
16. Lapouchnian, A.: Exploiting Requirements Variability for Software Customization and Adaptation. Ph.D. thesis, University of Toronto (2011)
17. Letier, E., van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. In: Proc. of the 12th ACM SIGSOFT Int. Symp. on Foundations of Software Eng. pp. 53–62. ACM, Newport Beach, California (2004)

18. Ramirez, A.J., Betty H.C. Cheng: Adaptive monitoring of software requirements. In: Proceedings of the IEEE Workshop on Requirements at Run Time. pp. 41–50. RE@RunTime, Sydney, Australia (September 2010)
19. Ramirez, A.J., Cheng, B.H.: Automatically deriving utility functions for monitoring software requirements. In: Proceedings of the 2011 International Conference on Model Driven Engineering Languages and Systems Conference. pp. 501–516. Wellington, New Zealand (2011)
20. Robinson, W.N.: Monitoring software requirements using instrumented code. In: HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences. pp. 276–285. IEEE Computer Society, Hawaii, USA (2002)
21. Salifu, M., Yu, Y., Nuseibeh, B.: Specifying monitoring and switching problems in context. In: IEEE Int. Requirements Engineering Conference. pp. 211–220 (Oct 2007)
22. Sawyer, P., Bencomo, N., Letier, E., Finkelstein, A.: Requirements-aware systems: A research agenda for re self-adaptive systems. In: Proceedings of the 18th IEEE International Requirements Engineering Conference. pp. 95–103. Sydney, Australia (September 2010)
23. Serrano, M., Serrano, M., Sampaio, J.C., Leite, P.: Dealing with softgoals at runtime: A fuzzy logic approach. In: Proceedings of the 2nd International Workshop on Requirements at Run Time. pp. 23–31. Trento, Italy (August 2011)
24. Silva Souza, V.E., Lapouchnian, A., Robinson, W.N., Mylopoulos, J.: Awareness requirements for adaptive systems. Tech. rep., University of Trento (2010)
25. Tarjan, R.: Depth-first search and linear graph algorithms. In: Proceedings of the 12th Annual Symposium on Switching and Automata Theory. pp. 114–121 (October 1971)
26. Villegas, N.M., Müller, H.A., Tamura, G., Duchien, L., Casallas, R.: A framework for evaluating quality-driven self-adaptive software systems. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 80–89. Waikiki, Honolulu, HI, USA (May 2011)
27. Walsh, W.E., Tesauro, G., Kephart, J.O., Das, R.: Utility functions in autonomic systems. In: Proceedings of the First IEEE International Conference on Autonomic Computing. pp. 70–77. IEEE Computer Society, New York, NY, USA (2004)
28. Welsh, K., Sawyer, P., Bencomo, N.: Towards requirements aware systems: Runtime resolution of design-time assumptions. In: Proc. of the 26th IEEE/ACM Int. Conf. on Automated Software Engineering. pp. 560–563. Lawrence, Kansas, USA (November 2011)
29. Welsh, K., Sawyer, P.: Understanding the scope of uncertainty in dynamically adaptive systems. In: Proceedings of the Sixteenth International Working Conference on Requirements Engineering: Foundation for Software Quality. vol. 6182, pp. 2–16. Springer, Essen, Germany (June 2010)
30. Whittle, J., Sawyer, P., Bencomo, N., Betty H.C. Cheng, Bruel, J.M.: RELAX: Incorporating uncertainty into the specification of self-adaptive systems. In: Proc. of the 17th Int. Requirements Eng. Conf. (RE '09). pp. 79–88. Atlanta, Georgia, USA (September 2009)
31. Yu, E.S.: Towards modeling and reasoning support for early-phase requirements engineering. In: Proc. of the Third IEEE Int. Symposium on Requirements Eng. pp. 226–235. Annapolis, MD, USA (January 1997)