Understanding the Scope of Uncertainty in Dynamically Adaptive Systems

Kristopher Welsh and Pete Sawyer

Computing Department, Lancaster University Lancaster, UK
{welshk,sawyer}@comp.lancs.ac.uk

Abstract. [**Context and motivation**] Dynamically adaptive systems are increasingly conceived as a means to allow operation in changeable or poorly understood environments. [**Question/problem**] This can result in the selection of solution strategies based on assumptions that may not be well founded. [**Principle ideas/results**] This paper proposes the use of claims in goal models as a means to reason about likely sources of uncertainty in dynamically adaptive systems. Accepting that such claims can't be easily validated at design-time, we should instead evaluate how the system will behave if a claim is proven false by developing a validation scenario. [**Contribution**] Validation scenarios may be costly to evaluate so the approach we advocate is designed to carefully select only those claims that are less certain, or whose falsification would have serious consequences.

Keywords: self adaptation, dynamically adaptive system, goal models, uncertainty, claims.

1 Introduction

Self-adaptation is emerging as a design strategy to mitigate maintenance costs in systems where factors such as complexity, mission-criticality or remoteness make off-line adaptation impractical. Self-adaptation offers a means to respond to changes in a system's environment by sensing contextual or environmental change at run-time and adapting the behaviour of the system accordingly. In this paper we refer to self-adaptive systems as dynamically adaptive systems (DASs) to reflect their ability to adapt autonomously to changing context at run-time.

Dynamically adaptive systems have now been deployed in a number of problem domains [1] yet remain challenging to develop because there is typically a significant degree of uncertainty about the environments in which they operate. Indeed, this uncertainty is the primary reason why a DAS must be able to self-adapt; to continue to operate in a range of contexts with different requirements or requirements trade-offs. DASs remain challenging to develop, despite advances made at different levels in the software abstraction hierarchy and by communities as diverse as AI and networking. At the architectural level [2], for example, compositional adaptation [3] promotes the re-use of DAS components. Compositional adaptation is one such approach in which structural elements of the system can be combined and recombined at run-time using adaptive middleware (e.g. [4]).

R. Wieringa and A. Persson (Eds.): REFSQ 2010, LNCS 6182, pp. 2–16, 2010.

Despite such advances, and the seminal work of Fickas and Feather [5] in requirements monitoring, the RE community has only recently started to address the challenges of dynamic adaptation. Our own recent contribution [6, 7, 8] has been to investigate the use of goal-based techniques for reasoning about the requirements for DASs. In [7], we advocated the use of *claims* from the NFR framework [9] in i* [10] strategic rationale models to enhance the traceabilty of DAS requirements.

In this paper, we go a step further and argue for the utility of claims in DAS goal models as markers for uncertainty. This research result has emerged as an unexpected side-effect of our work on claims for tracing. Design rationale in a DAS is not always founded on good evidence, but sometimes on supposition about how the system will behave in different, hard-to predict contexts. Thus claims can serve not only as design rationale but also as proxies for analysts' understanding. It is crucial that the consequences of decisions based on assumptions that may subsequently prove to be false are understood, even if there is insufficient data to validate the assumptions themselves. We propose that a *validation scenario* should be defined to evaluate the effect of a claim turning out to be false.

The primary contribution of the paper is a simple means for reasoning about hierarchies of claims to understand how uncertainty propagates throughout these hierarchies. We show how this knowledge can be leveraged to improve the robustness of a DAS' specification using validation scenarios while minimizing the number of validation scenarios that need to be defined and evaluated. We demonstrate our approach using a case study drawn from a sensor grid that was deployed on the River Ribble in the Northwest of England. This sensor grid has acted as a preliminary evaluation for our use of claim reasoning in DASs.

The rest of the paper is structured as follows. In the next section, section 2, we introduce what we mean by claim reasoning and in section 3 we explain how an existing DAS modeling process may be adapted to include claims. We then use a case study to illustrate claim reasoning about a DAS in section 4, and conclude with a brief survey of related work (section 5) and final conclusions (section 6).

2 Claim Reasoning

In [7] we augmented i* models used to model DASs with claims, a concept borrowed from the NFR toolkit [9]. As is now well-known within RE, i* supports reasoning about systems in terms of agents, dependencies, goals and softgoals. We showed how claims can be used to record requirements traceability information by explicitly recording the rationale behind decisions, in cases where the contribution links assigned to softgoals for different solution alternatives in i* strategic rationale (SR) models don't reveal an obvious choice. We argued that this enhances the tracing information in a way that is particularly important for a system that can adapt at run-time to changing context.

As described above, claims capture the rationale for selecting one alternative design over another. As an example and before considering the effect of claims with respect to self-adaptive behaviour, consider the fragment of a simple SR model of a robot vacuum cleaner for domestic apartments depicted in Fig 1. The vacuum cleaner has a goal to clean the apartment (*clean apartment*) and two softgoals; to avoid

causing a danger to people within the house (*avoid tripping hazard*) and to be economical to run (*minimize energy costs*). The vacuum cleaner can satisfy the clean apartment goal by two different strategies that have been identified. It can clean at night or when the apartment is empty. These two strategies are represented by two alternative tasks related to the goal using means-end relationships. The choice of best strategy is unclear because at this early stage of the analysis, it is hard to discriminate between the extent to which each solution strategy satisfices the softgoals. The balance of –ve and +ve effects on satisficement of the softgoals appears to be approximately the same for both, but to different softgoals. This is depicted by the contribution links labeled *help* and *hurt*. However, the choice is resolved using a claim, which has the effect of asserting that there is *no tripping hazard.* The claim thus *breaks* the *hurt* contribution link between the task *clean at night* and the softgoal *avoid tripping hazard*. The *break*-ing claim nullifies the contribution link to which it is attached. In this case it nullifies the negative impact that night cleaning was presumed to have on tripping hazard avoidance. In turn, this has the effect of promoting the night cleaning strategy over the empty apartment cleaning strategy since it now appears to better satisfice the two softgoals. The inverse of a *break*-ing claim is a *make*-ing claim, which lends additional credence to a contribution link, implying the importance of the satisfaction of a softgoal with a *helps* link or the unacceptability of failing to satisfy a softgoal with a *hurts* link[1]. Note that claims speak of the *importance* of the effects captured by the contribution link, not the *magnitude* of the effect, which can be captured using fine-grained contribution links.

The *no tripping hazard* claim is sufficient to select the night cleaning strategy, but only if there is sufficient confidence in the claim that no hazard is offered. However, the analyst may have greater or lesser confidence in a claim, so claim confidence spans a range from axiomatic claims in which full confidence is held, to claims that are mere assumptions. At the assumption end of the claim confidence range, a claim is essentially a conjecture about a Rumsfeldian "known unknown" [11] and thus serves as a marker of something about which uncertainty exists.

If a claim is wrong, the performance of the system may be unsatisfactory, or the system may exhibit harmful emergent behaviour, or even fail completely. Ideally, the claims should be validated before the system is deployed. In a DAS, this may be very hard to do, however. Since the world in which a DAS operates is imperfectly understood, at least some conjectural claims are likely to be impossible to validate at design-time with complete assurance.

Given this fundamental limitation on claim validation, the behaviour of a system should be evaluated in cases where claims turn out to be false. To do this, a validation scenario should be designed for each claim to help establish the effects of claim falsification, such as whether it causes undesirable emergent behaviour. We do not define the form of a validation scenario; it may be a test case or some form of static reasoning. However, developing and executing validation scenarios for each claim can be expensive. A validation scenario should be developed for every possible combination of broken and unbroken claims. Hence, the number of validation scenarios (T) is $T=2^n-1$ where n represents the number of claims that make or break a softgoal contribution link. One of the three *target systems* (explained below) of the *GridStix* system

[1] Note that there are several other types of contribution and claim link to those presented here.

we describe later, would need 31 validation scenarios. Fortunately, it is possible to reduce this number by focusing attention only on claims towards the assumption end of the claim confidence range and by using *claim refinement models.*

While claims provide the rationale for selection of one solution strategy over another, the derivation of a claim may be obscure. A claim derivation is obscure if the logic of the claim is not obvious from its name. In this case, the rationale for the claim can be represented explicitly in a hierarchical claim refinement model, with the facts and assumptions from which the claim is derived also represented as claims. The claims form nodes in claim refinement model the branches of which can be AND-ed or OR-ed.

In Fig 1, the *no tripping hazard* claim is obscure because it appears as an assertion with no supporting evidence. Fig 2 is a claim refinement model that shows that the *no tripping hazard* claim is derived from three other claims: *family sleeps at night*, *vacuum is easy to see* and *vacuum has warning light*, arranged as a hierarchy of claims in a claim refinement model. Note that the root of a hierarchy of claims is at the bottom. This *bottom-level claim* is what appears on the SR diagram. The claim *vacuum is easy*

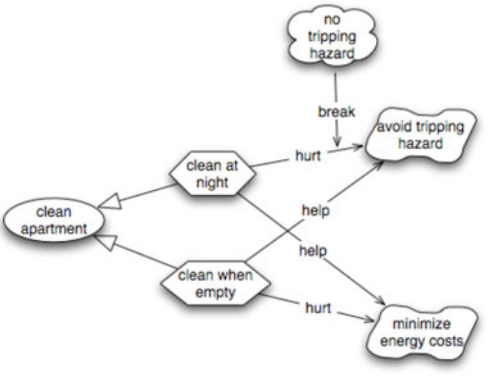

Fig. 1. A robot vacuum cleaner

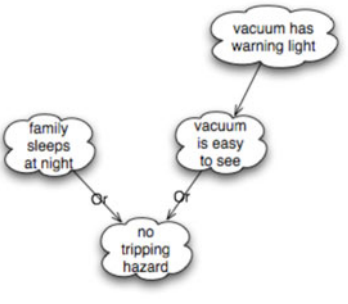

Fig. 2. Claim refinement model for *no tripping hazard* claim

to see is derived from the claim *vacuum has warning light*. The claims *family sleeps at night* and *vacuum is easy to see* together derive the bottom-level claim *no tripping hazard,* which is the claim underpinning the selection of the night cleaning strategy for the vacuum cleaner.

Falsity of any claim will propagate down the claim refinement model to the bottom-level claim. If the *family sleeps at night* claim is shown to be untrue or only partially true, the claim that there is *no tripping hazard* is upheld by the Or-ed claim *vacuum is easy to see*, provided the latter claim is sound. If confidence in all the claims in a claim refinement model was low, a validation scenario would have to be produced for every combination of true and false claims. However, some of the claims in the model may be axiomatic and the logic of claim derivations may give confidence in the bottom-level claims even where they are in part derived from non-axiomatic claims. To exploit this, a binary classification of claims, as axiomatic or conjectural, might be used:

Unbreakable claims are axiomatic and can be removed from consideration. *Vacuum has warning light* is such a claim.

Uncertain claims are conjectural in that it is considered possible that the claim could prove false at run-time. The claim that the *family sleeps at night*, and by implication would not encounter the vacuum cleaner, is conjectural because it cannot be assumed to be generally true. In contrast to the *vacuum is easy to see* claim, there is a significant possibility of this claim being proven false, since even a normally sound sleeper might awake and visit the bathroom while the vacuum cleaner is operating.

Using this classification a validation scenario should be developed for every bottom-level claim that resolves to *uncertain*, but need not be developed for one that resolves to *unbreakable*. Thus, for the robot vacuum cleaner, a validation scenario is needed to examine the consequences of the vacuum cleaner operating at night if the family was not asleep.

Unbreakable and *uncertain* represent the two extremes of the claim confidence spectrum. There may be claims that could be falsified but for which it is believed to be improbable that the system will encounter a situation where the claims are broken. We propose that such claims be classified as **Qualified.** However, a qualified claim should be re-classified *uncertain* if, despite the low probability of them being falsified, the consequences of them being so might be serious. Note that a claim with a high probability of falsification would already be classified *uncertain*, even if the consequences were considered minor.

The claim *vacuum is easy to see* might be classified as a qualified claim. Even a vacuum cleaner whose visibility is enhanced by a warning light may not be easily noticeable to people with visual impairment or people who are sleep-walking. However, sleep-walking is unusual and it is unlikely that robot vacuum cleaners would be recommended for people for whom they were obviously a hazard.

To propagate claim values (unbreakable, qualified, uncertain), the following rules apply:

- Where a claim is derived from a single claim (e.g. *vacuum is easy to see* is derived directly from *vacuum has warning light*), the derived claim inherits the value of the upper-level claim; the upper-level claim *make*s or *break*s the derived claim. Hence, *vacuum is easy to see* assumes the value unbreakable directly from *vacuum has warning light.*

- Where a claim is derived from two or more AND-ed claims, it inherits the value of the weakest of the claims – i.e. the claim least certain to be true. Hence if an unbreakable and an uncertain claim are AND-ed to derive a bottom-level claim, the bottom-level claim will be uncertain.
- Where a claim is derived form two or more OR-ed claims, it inherits the value of the strongest of the claims. Hence *no tripping hazard* assumes the value unbreakable from *vacuum is easy to see*, despite *family sleeps at night being* uncertain. In the example, this has the effect of validating selection of the *night-time cleaning* strategy in Fig 1.

The classification of claims cannot be easily formalized or automated. It requires human judgment by the analyst and stakeholders. However, if claims are classified thoughtfully and propagated through the claim refinement model, the result can be considered to be a form of threat modeling [12]. The exercise generates validation scenarios for the goal models but prunes back the set of all possible validation scenarios to include only claims that are judged plausible or high risk. In the next sections, we describe how claims can be used to extend LoREM, a previously-reported approach for modeling DAS requirements [6].

3 Modeling DASs with LoREM

LoREM offers a requirements modeling approach for DASs, following the principles set out in [13]. Here, to use Michael Jackson's terminology [14], the DAS represents the machine, while the environment or context in which it operates represents the world. The world is considered to comprise a set of discrete environmental states called *domains*. The machine is conceptualized as a set of distinct programs called *target systems*, each of which is designed to operate in a particular domain. In LoREM, each target system is explicitly specified, as are the domains and the conditions that trigger the DAS to adapt from one target system to another.

In LoREM, the set of domains is identified as part of the strategic dependency (SD) model. A different strategic rational (SR) model is then developed for each target system according to the principle of one target system for each domain, as demonstrated in the case study in the next section. Underpinning LoREM is an assumption that a DAS must satisfice a set of NFRs, but that the balance of satisficement and the consequent trade-offs differs according to domain. Hence, in the GridStix river flood warning system described below, *energy efficiency* and *fault tolerance* are in tension. Which is prioritized over the others depends on whether the river is (e.g.) quiescent or in flood. Using LoREM, these key NFRs are identified in the SD model as softgoals and each target system's SR model selects a solution strategy that optimizes the trade-offs in softgoal satisficements.

In [7] we augmented LoREM with claims. Claims are used in the target system SR models to annotate the contribution links between the tasks that represent the alternative solutions strategies and the softgoals, in exactly the same way as they were used for the vacuum cleaner model in Fig 1. The motivation for this was to make the requirements rationale (in terms of alternative selection) explicit and so aid tracing. Underpinning this rationale was the likelihood that, even with their self-adaptive capability, to have a usefully long life, a DAS would still be subject to off-line adaptive

and corrective maintenance over time. This would be necessary as understanding about its environment improved, as ways to improve its performance were identified and as its stakeholder's goals evolved.

As argued in the previous section, claims can also serve as markers for uncertainty, with conjectural or *uncertain* claims forming the focus of validation scenarios. A validation scenario captures a situation where combinations of uncertain claims are taken to be false. A claim refinement model is used to propagate the effects of the negated claims to other, derived claims. The validation scenario thus helps understand the impact on the validity of the solution strategies that those claims help select, either by static analysis or by testing.

At design time, the validation scenarios help answer what-if questions about the impact of the uncertainty that conjectural claims represent. Once the DAS is deployed, it may be possible to resolve many of the underlying uncertainties. If the validation scenarios have been applied correctly and complete system failure has not resulted from a broken claim, the resolved uncertainties may be put to good use.

A DAS monitors its environment and uses the monitored data to make adaptation decisions. In most DASs currently deployed, including those which LoREM was designed to support, a DAS can adapt in pre-determined ways to changes in its environment that were envisioned at design-time, albeit with elements of uncertainty. Where those envisioned environmental changes and the DAS's corresponding adaptations are based upon claims, the DAS may accumulate data that can be used to validate the claims. This is a form of requirements monitoring [5], the data from which may subsequently be used to inform off-line adaptive or corrective maintenance. Moreover, as we discuss in the conclusions section, it may even be possible to dynamically adopt an alternative solution strategy at run-time. At present, however, this is future work.

The next section presents a case study to illustrate the utility of claim reasoning in DAS goal models.

4 Case Study

To illustrate the use of claims for validation scenario identification in LoREM, we present a conceptually simple but real DAS and work through the model analysis for a single target system.

GridStix [15] is a system deployed on the River Ribble in North West England that performs flood monitoring and prediction. It is a sensor network with smart nodes capable of sensing the state of the river, processing the data and communicating it across the network. The hardware available includes sensors that can measure depth and flow rate, and wireless communication modules for the Wi-Fi and Bluetooth standards, all supplied with power by batteries and solar panels. The system software uses the GridKit middleware system [4], which provides the GridKit's self adaptive capabilities using component substitution.

The flow rate and river depth data is used by a *point prediction model* which predicts the likelihood of the river flooding using data from the local node and data cascaded from nodes further upstream. The more upstream nodes from which data is available, the more accurate is the prediction. GridStix acts as a lightweight Grid,

capable of distributing tasks. This is important because some tasks, such as execution of the point prediction models, can be parallelized and are best distributed among the resource-constrained nodes. However, distributing the processing comes at the cost of increased energy consumption and this is one of the factors that affect satisficement of the *energy efficiency* softgoal mentioned previously. Distribution may be effected by communicating between nodes using either the IEEE 802.11b (refered to as Wi-Fi in the rest of this paper) or Bluetooth communication standards. Bluetooth has lower power consumption, but shorter range. Bluetooth-based communication is hence thought to be less resilient to node failure than Wi-Fi-based communication. The choice of spanning tree algorithm can also affect resilience. A fewest-hop (FH) algorithm is considered better able to tolerate node failure than the shortest-path (SP) algorithm. However, data transmission via SP typically requires less power due to the smaller overall distance that the data must be transmitted.

The GridKit middleware can configure itself dynamically to support all the variations implied above; local node or distributed processing, Wi-Fi or Bluetooth communications and different network topologies, as well as others.

The environment in which GridStix operates is volatile, as the river Ribble drains a large upland area that is subject to high rainfall. The river is therefore liable to flooding with consequent risk to property and livestock among the communities sited on the flood plain. Stochastic models provide only an imperfect understanding of the river's behaviour. Moreover, events upstream, such as construction sites or vegetation changes, may alter its behaviour over time. There is therefore significant uncertainty associated with developing an autonomous sensor network for deployment on the river.

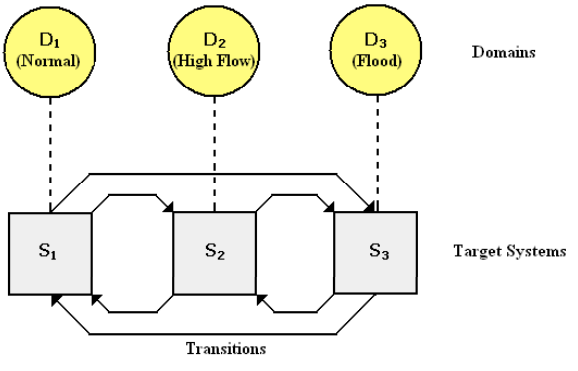

Fig. 3. Mapping Between GridStix Domains and Target Systems

The River Ribble, then is GridStix's environment. Hydrologists acting as domain experts partitioned the environment into three discrete domains: D_1 (Normal), D_2 (High Flow) and D_3 (Flood). The D_1 (Normal) domain is characterised by a quiescent river, with little imminent risk of flood or danger to local residents or the nodes themselves. The D_2 (High Flow) domain features a fast-flowing river that may presage an imminent flood event. The D_3 (Flood) domain occurs when the depth increases and

the river is about to flood. When this happens, GridStix itself is at risk of damage from node submersion and from water-borne debris.

GridStix is conceptualised as comprising three target systems, S_1, S_2 and S_3 tailored to domains D_1, D_2 and D_3 respectively, as shown in Fig 3.

The LoREM models for the system have previously been published in [6]. Here, we will focus on only one of the target systems to illustrate the use of claim reasoning. The SD model identified a single overall goal *Predict flooding*, and three softgoals; *Fault tolerance*, *Energy efficiency* and *Prediction accuracy*. The SR model for S_3 (Flood) is depicted in Fig 4.

Flood prediction in S_3 is operationalised by the task *Provide Point Prediction* which can be further decomposed into the (sub)goals: *Measure Depth*, *Calculate Flow Rate* and the task *Communicate Data*. The *Communicate Data* task depends on the *Transmit Data* and *Organize Network* subgoals being achieved. Satisfaction of the *Measure Depth* and *Calculate Flow Rate* goals produces *Depth* and *Flow Rate* resources respectively. These are used elsewhere in LoREM models which we do not consider here.

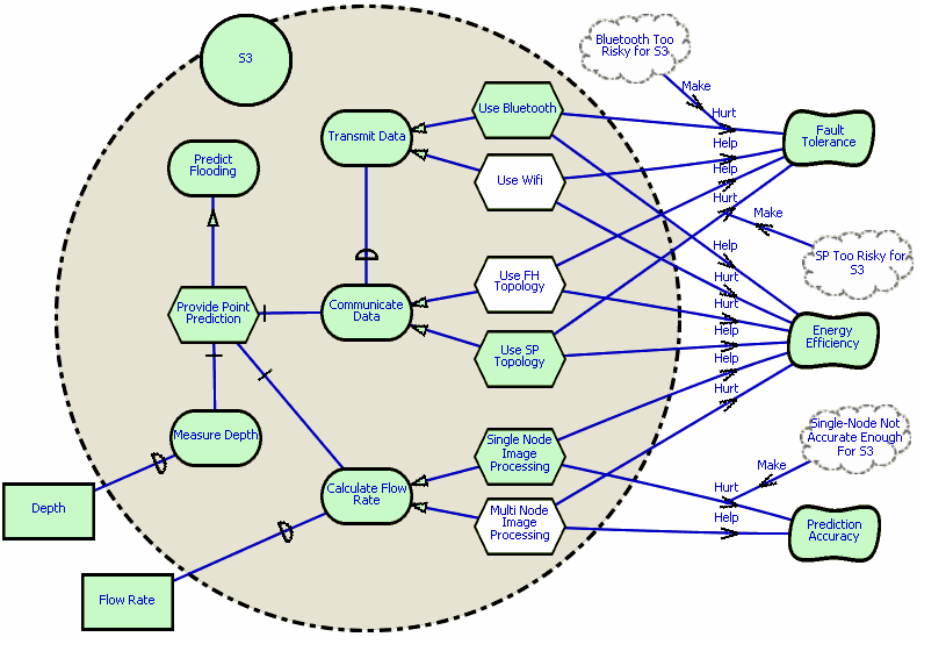

Fig. 4. GridStix Behaviour Model for S_3 (Flood)

The *Calculate Flow Rate*, *Organize Network* and *Transmit Data* goals all have several alternative satisfaction strategies, represented by tasks connected to the respective goals with means-end links. The three softgoals are depicted on the right of Fig 4, and the impact of selecting individual tasks on the satisficement of each of the softgoals is represented by contribution the links with values *help* or *hurt*.

The tasks represent alternative solution strategies and those selected to satisfy goals in S_3 are coloured white in Fig 4. Attached to some of the contribution links in Fig 4 are three claims that *make* or *break* the softgoal contributions. The claim refinement model for S_3 is depicted in Fig 5.

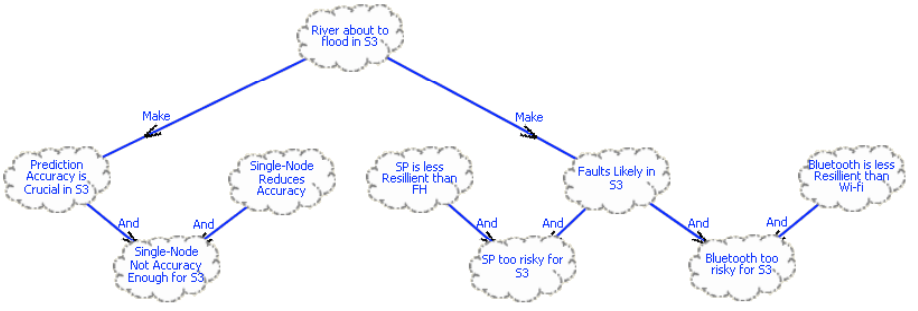

Fig. 5. GridStix Claim Refinement Model for S_3 (Flood)

The three claims shown in Fig 4 appear at the bottom of Fig 5, connected to the claims from which they are derived by contribution links. They show, for example, that Wi-Fi was selected over Bluetooth to satisfy the *Transmit Data* goal because Bluetooth was considered too risky in terms of Fault Tolerance. Examining Fig 5 allows the basis for this claim to be established: that Bluetooth is less resilient than Wi-Fi and that, given the river is about to flood in S_3, there is a significant risk of node failure. Bluetooth is considered less resilient than Wi-Fi because of its poorer range, which reduces the number of nodes an individual node may communicate with, so increasing the likelihood of a single node failure hampering communication throughout the network.

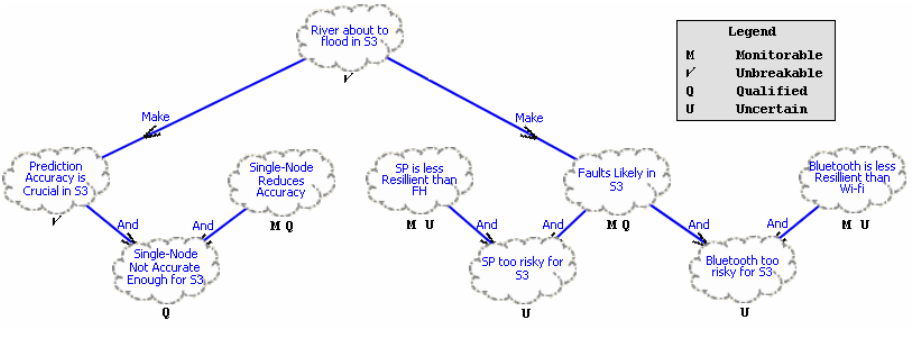

Fig. 6. GridStix Claim Refinement Model for S_3 (Flood) Annotated with Claim Classifications

There are three bottom-level claims in Figs 4 & 5, and we would thus need seven validation scenarios for every combination of claims. An analysis of the claims classified each according to the scheme presented in section 2. The results are shown in Fig 6.

Two claims were considered uncertain. The claims *SP is less resilient than FH* and *Bluetooth is less resilient than Wi-Fi* were both considered uncertain because the analyst was not certain whether the theoretically more resilient option in each case would actually prove demonstrably more resilient in the field. Note that the *Monitorable* tag (M) denotes a claim that could be directly monitored by the DAS at run-time. We return to claim monitoring in the conclusions.

Propagating the unbreakable, qualified, and uncertain values through the tree in Fig 6, shows that two of the three bottom-level claims, *SP too risky for S3* and *Bluetooth too risky for S3*, could be impacted by an uncertain claim and thus be uncertain themselves. The consequence of this analysis is that three validation scenarios are necessary. Recall that the purpose of the validation scenarios is to investigate the consequences of false claims, not to validate the claims themselves. The three identified validation scenarios contrast with the seven that would be necessary if all the claims were assumed to be falsifiable. The three scenarios are summarized in table 1.

Table 1. Validation scenarios for S3 uncertain claims

Scenario	*Single-Node[…] not accurate enough for S3*	*SP too risky for S3*	*Bluetooth too risky for S3*
1	True	True	False
2	True	False	True
3	True	False	False

Validation scenario 1, for example, would be designed to discover how GridStix behaved if the Bluetooth turned out to be no less fault-tolerant than Wi-Fi. Reasoning about the scenario concludes that GridStix's adaptations from Bluetooth to Wi-Fi would, at worst, simply waste energy (since Wi-Fi consumes more power than Bluetooth) should the *Bluetooth too risky for S3* claim indeed turn out to be false in practice. This is a relatively benign consequence but given the energy constraints on GridStix, it could be worth devoting effort to investigating the claims' validity, perhaps by simulation, or by monitoring the deployed system. In general, if a validation scenario reveals undesirable effects of a failed claim, there are two courses of action. As with the *Bluetooth too risky for S3* claim, additional effort might be assigned to understanding the claim's soundness. Alternatively, or in addition, an alternative solution strategy that circumvented the problem underlying the failed claim might be sought, using a new claim to record the rationale for its selection.

The same analysis was performed on the claim refinement models for the other two GridStix target systems: S_1 and S_2. The results of the analysis for all three are depicted in Table 2.

Table 2. GridStix Bottom-Level Claim Distribution by Target System

Target System	Unbreakable Claims	Qualified Claims	Uncertain Claims
S_1	2	1	2
S_2	1	0	2
S_3	0	1	2

The S_1 target system stands out as having the largest number of bottom-level claims (five), which reflects the complexity of the trade-offs in available solution strategies for this domain. Table 3 shows the numbers of validation scenarios that would be needed in each target system for all claims, qualified and uncertain claims, and uncertain claims only, respectively. This shows that using our strategy, the number of validation scenarios for S_1, even though it has five bottom-level claims, could be restricted to three for uncertain claims or seven if further assurance was felt necessary by including qualified claims.

Table 3. Emergent Behaviour Testing Scenarios for GridStix

Target System	All Scenarios	Qualified & Uncertain	Uncertain Scenarios
S_1	31	7	3
S_2	7	3	3
S_3	7	7	3
Total	45	17	9

Thus, devising validation scenarios for all combinations of uncertain claims would require nine scenarios, and for the qualified and the uncertain claims would require seventeen scenarios. Devising scenarios for all combinations of claims in the GridStix system would require forty-five validation scenarios.

The GridStix system, as presented here, is only modestly complex from a modelling perspective, with only three softgoals and three target systems. Many DASs will be significantly more complex. The effort involved in evaluating the consequences of poorly-informed softgoal trade-offs in a DAS increases rapidly with the number of potentially-conflicting NFRs involved. The technique described here has the potential to focus and limit this effort by the use of claims to explicitly record the rationale for solution strategy selection and by explicitly considering the soundness of those claims. An undesirable side-effect of using claims is that they add to the problem of complexity management in i* diagrams [21]. This is partially mitigated by the fact that we have designed claims to work with LoREM. Using LoREM, an SR model for a DAS is partitioned, with a separate SR model for each target system.

5 Related Work

There is currently much interest in software engineering for DASs [16]. Much of this work has been in the design of software architectures that enable flexible adaptations [2]. Much research in RE for self-adaptation has focused on run-time monitoring of requirements conformance [5, 17], which is crucial if a DAS is to detect when and how to adapt at run-time. More recently, attention has turned to requirements modeling for DASs, and a number of authors (e.g. [8, 18]) report on the use of goals for modeling requirements for DASs. Goal models are well suited to exploring the alternative solution strategies that are possible when the environment changes. Here, adaptation is seen as the means to maintain goal satisficement, while goal modeling notations such as KAOS [19] and i* [10] support reasoning about goals and softgoals.

A key challenge posed by DASs for RE is uncertainty about their environments and a number of modeling approaches for handling uncertainty have been proposed.

Letier and van Lamsweerde propose a formal means to reason about partial goal satisfaction. Cheng et al. [8] use KAOS's obstacle analysis to reason about uncertainty, utilizing a small set of mitigation strategies that include directly framing the uncertainty using the RELAX requirements language [20]. In this paper, we propose augmenting the i* models used by the LoREM approach to DAS modeling [6] with the *claim* construct adapted from the NFR framework [9]. We argue that by using claims as the rationale for selecting between alternative solution strategies, they can also serve as explicit markers for uncertainty where rationale is induced from assumed properties of the environment or the DAS's own behaviour.

There are two important differences between claims and the *belief* construct that is built in to i*. The first is that an i* belief represents a condition that an *actor* holds to be true. In our use of claims, the claim may also represent a condition that the analyst holds to be true. The second difference is that a belief attaches to a softgoal while a claim attaches to a softgoal's contribution link. Hence, a claim is able to provide the explicit rationale for selecting a particular solution strategy.

6 Conclusions

Claims attached to i* softgoal contribution links can be used to provide the rationales for selecting from several alternative solution strategies. Used this way, claims can be useful for tracing in DAS goal models [7]. Moreover, as we argue in this paper, claims may also be used as markers of uncertainty. The utility of claims may extend beyond DASs, but we focus on DASs because there is often significant uncertainty about a DAS's environment. Uncertainty may even extend to the DAS's own, *emergent* behaviour, if (e.g.) adaptation results in unexpected configurations of run-time-substitutable components.

Not all claims represent uncertainty, however. The confidence level in a claim will generally fall somewhere on a spectrum from axiomatic to pure conjecture. Conjectural claims represent uncertainty; assumptions that cannot be validated at design-time. Conjectural claims may therefore be falsified at run-time, possibly leading to a variety of undesirable effects. Accepting that such claims can't be easily validated at design-time, we should instead evaluate how the system will behave if a claim proves to be false by developing a *validation scenario*. A validation scenario subsumes a test case that may be developed for some combination of false claims, but also allows for static evaluation if the claims are hard to simulate in a test harness.

Validation scenarios may be costly to evaluate so the approach we advocate is designed to carefully select only those claims that have a significant probability of being false and those with a low probability of being false but whose falsification would be serious. To do this we advocate classifying claims as *unbreakable*, *qualified* or *uncertain*, and then propagating claim values through a claim refinement model.

As future work, we are developing the means to monitor claims at run-time, using the techniques of requirements monitoring [5]. Data collected about claim soundness may be used for subsequent off-line corrective maintenance. However, if the goal models can be treated as run-time entities where they can be consulted by the running system, the DAS may adapt by dynamically selecting alternative solutions when a claim is falsified. Such a system introduces new adaptive capabilities but also further

risk of undesired emergent behaviour so the identification of validation scenarios is acutely necessary. This run-time claim validation accounts for why we have chosen the name *unbreakable* rather than (e.g.) *axiomatic* for those claims at the opposite end of the spectrum from conjectural claims. Where claims can be validated and acted upon at run-time, we need to be able to monitor them. Not all claims are monitorable and the term unbreakable simply reflects that, at run-time, an unmonitorable claim can never appear to break from the perspective of the claim monitoring logic. Of course, such an unmonitorable claim may nevertheless be false and this may be exhibited externally as a failure.

References

1. Cheng, B., de Lemos, R., Giese, H., Inverardi, P., Magee, J.: Software engineering for self adaptive systems. In: Dagstuhl Seminar Proceedings (2009)
2. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: FOSE 2007: 2007 Future of Software Engineering, pp. 259–268. IEEE Computer Society, Los Alamitos (2007)
3. McKinley, P., Sadjadi, S., Kasten, E., Cheng, B.: Composing adaptive software. Computer 37(7), 56–64 (2004)
4. Grace, P., Coulson, G., Blair, G., Mathy, L., Duce, D., Cooper, C., Yeung, W., Cai, W.: Gridkit: pluggable overlay networks for grid computing. In: Symposium on Distributed Objects and Applications (DOA), Cyprus (2004)
5. Fickas, S., Feather, M.: Requirements monitoring in dynamic environments. In: Second IEEE International Symposium on Requirements Engineering (RE 1995), York, UK (1995)
6. Goldsby, H., Sawyer, P., Bencomo, N., Cheng, B., Hughes, D.: Goal-Based modelling of Dynamically Adaptive System requirements. In: ECBS 2008: Proceedings of the 15th IEEE International Conference on Engineering of Computer-Based Systems, Belfast, UK (2008)
7. Welsh, K., Sawyer, P.: Requirements tracing to support change in Dynamically Adaptive Systems. In: Glinz, M., Heymans, P. (eds.) REFSQ 2009. LNCS, vol. 5512, pp. 59–73. Springer, Heidelberg (2009)
8. Cheng, H., Sawyer, P., Bencomo, N., Whittle, J.: A goal-based modelling approach to develop requirements of an adaptive system with environmental uncertainty. In: MODELS 2009: Procedings of IEEE 12th International Conference on Model Driven Engineering Languages and Systems, Colorado, USA (2009)
9. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Springer International Series in Software Engineering 5 (1999)
10. Yu, E.: Towards modeling and reasoning support for early-phase requirements engineering. In: RE 1997: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE 1997), Washington DC, USA (1997)
11. Department of Defence: DoD News Briefing - Secretary Rumsfeld and Gen. Myers, http://www.defense.gov/transcripts/transcript.aspx?transcriptid=2636
12. Schneier, B.: Attack Trees - Modeling security threats. Dr. Dobb's Journal (1999)
13. Berry, D., Cheng, B., Zhang, J.: 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)
14. Jackson, M.: Problem frames: analyzing and structuring software development problems. Addison-Wesley Longman, Amsterdam (2000)

15. Hughes, D., Greenwood, P., Coulson, G., Blair, G., Pappenberger, F., Smith, P., Beven, K.: Gridstix: Supporting Flood prediction using embedded hardware and next generation grid middleware. In: 4th International Workshop on Mobile Distributed Computing (MDC 2006), Niagara Falls, USA (2006)
16. Cheng, B., et al.: Software engineering for self-adaptive systems: A research road map. In: Cheng, B., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. Dagstuhl Seminar Proceedings, vol. 08031 (2008)
17. Robinson, W.: A requirements monitoring framework for enterprise systems. Requirements Engineering 11(1), 17–41 (2006)
18. Lapouchnian, A., Liaskos, S., Mylopoulos, J., Yu, Y.: Towards requirements-driven autonomic systems design. In: DEAS 2005: Proceedings of the 2005 Workshop on Design and Evolution of Autonomic Application Software (DEAS), St. Louis, MO, USA (2005)
19. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. John Wiley & Sons, Chichester (2009)
20. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B., Bruel, J.-M.: RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In: Proc. 17th IEEE International Conference on Requirements Engineering (RE 2009), Atlanta, Georgia (August 2009)
21. Moody, D., Heymans, P., Matulevicius, R.: Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of the i* Visual Notation. In: Proc. 17th IEEE International Conference on Requirements Engineering (RE 2009), Atlanta, Georgia (August 2009)