Towards Requirements Aware Systems: Run-time Resolution of Design-time Assumptions

Kristopher Welsh, Pete Sawyer
School of Computing & Communications
InfoLab21, Lancaster University
Lancaster, UK
k.welsh@lancaster.ac.uk, sawyer@comp.lancs.ac.uk

Nelly Bencomo
INRIA Paris - Rocquencourt
Le Chesnay, Franceaine de Voluceau, B.P.105
78153 Le Chesnay, France
nelly@acm.org

***Abstract*—In earlier work we proposed the idea of requirements-aware systems that could introspect about the extent to which their goals were being satisfied at runtime. When combined with requirements monitoring and self adaptive capabilities, requirements awareness should help optimize goal satisfaction even in the presence of changing run-time context. In this paper we describe initial progress towards the realization of requirements-aware systems with REAssuRE. REAssuRE focuses on explicit representation of assumptions made at design time. When such assumptions are shown not to hold, REAssuRE can trigger system adaptations to alternative goal realization strategies.**

***Keywords*-requirements awareness; self adaptive systems; goals; claims;**

I. Introduction

In [1] previous work, we have proposed the LoREM goal-driven method for deriving requirements for self-adaptive systems [2]. In this paper we extend this work to derive the REAssuRE method (REcording of Assumtions in RE) to support reasoning over uncertainty. REAssuRE extends the i* [3] Strategic Rationale (SR) goal models used by LoREM to include *claims*. In REAssuRE, claims record the rationale for selection between alternative goal realization strategies.

Self-adaptation is often used to mitigate an inability to accurately predict the range of environmental contexts that a system will encounter at run-time. The specification and design of self adaptive systems is thus often subject to uncertainty, forcing the developer to make assumptions in order to identify and define the means to achieve the system's goals. Using claims, an assumption made in selecting a goal realization strategy can be made explicit. In REAssuRE, if data collected by monitoring provides evidence that such an assumption is false, the effects can be propagated to the goal models used to specify the goal realization strategy, which can then be automatically re-evaluated. This may trigger the system to adapt by binding to an alternative means of goal realization.

The primary contribution of this paper is to demonstrate the feasibility of maintaining goal models at runtime and their utility for guiding principled adaptations to contexts encountered at runtime that were imperfectly understood at design time. In succeeding sections we will describe REAssuRE in more detail and describe its use in an adaptive flood warning system, before briefly reviewing related work and drawing final conclusions.

II. REAssuRE

Goal modeling enables the intention (goals) of both the computational and the non-computational agents in a domain to be modeled. The partitioning of goal models by agent in i* [3] is a good match for the approach recommended by [4], and followed by LoREM [1], in which a volatile environmental context can be conceptualized as a finite set of discrete environmental contexts.

A further advantage of goal-oriented RE, is the ability to model non-functional requirements (NFRs) directly as *softgoals* and to represent the impact of different solution strategies on softgoal *satisficement*. This feature is exploited by LoREM, where for all contexts, the system is considered to have a common set of goals and softgoals. However, goal realization is specified on a per-context basis, and tailored to what is the optimal trade-off among the softgoals in each context. For example, in a sensor network whose goal was to gather data about a volcano, conserving battery power might be prioritized during quiescent periods, but frequent data collection might be the priority when an eruption appeared imminent.

In i*, reasoning about the best goal satisfaction strategy is informed by the values that the analyst gives to the *contribution links* that record the expected effect of each realization strategy on the softgoals. Realization strategies are modeled as *tasks*. In its simplest form, a task can make, help, hurt, break or have a neutral effect on satisficement of a softgoal. However, where uncertainty exists, perhaps because of imperfect knowledge about the environment, the effect of different goal realization strategies on the softgoals may be uncertain, forcing the analyst to make assumptions.

In REAssuRE, we extend the i* strategic rationale (SR) models used in LoREM with claims, a concept borrowed from the NFR framework [5]. Claims are attached to softgoal contribution links and are used to record the rationale for a choice of goal realization strategy when there is uncertainty about the optimum choice. Claims serve as markers for uncertainty, helping the analyst evaluate the consequences of assumptions proving false.

978-1-4577-1639-3/11/$26.00 © ASE 2011, Lawrence, KS, USA

Claims thus serve a static purpose of making the assumptions that may underly key design decisions implicit and traceable [6]. However, claims also have *dynamic* utility. By maintaining run-time representations of the goal models and monitoring claims via context variables, the effects of a claim proving false can be propagated to the goal model and the merits of the competing goal realization strategies can be dynamically reevaluated. Thus, a system may mitigate the corresponding mistaken assumption by self-adapting to a configuration that represents a better solution strategy.

The semantics of claim propagation are simple. The impact of the truth of claims on the contribution links to which they attach is boolean; they either make or break them resulting in make, neutral or break contributions:

$$claim = break \Rightarrow c' = neutral$$

$$claim = make \wedge (c = hurt \vee c = break) \Rightarrow c' = break$$

$$claim = make \wedge (c = help \vee c = make) \Rightarrow c' = make$$

Where c is the value that annotates the contribution link to which the claims is attached and *c'* is the value of the contribution link when combined with the effect of the claim. Thus, revising the truth value of a claim can affect the extent to which a task satisfices a softgoal, and this in turn can (e.g.) change the task from representing the optimal goal realization strategy to one that is no longer the best of those available.

The above makes a number of simplifying assumptions about the complexity of the i* models used. In particular, we assume that each alternative solution strategy is represented by a claim which is in turn modelled as a leaf of the goal tree. Moreover, for each such claim, its impact on satisficement of the relevant softgoals is represented by a contribution link. It is to these contribution links to which claims may be attached.

Thus, to exploit REAssuRE for run-time reasoning, the expressiveness of i* goal models is restricted to a constrained subset. The payoff of the these constraints is that an automatic reasoning system can evaluate softgoal satisficement at runtime which, when coupled with a self-adaptive capability in the system under design, should help guarantee adaptation decisions that are principled and taken explicitly to further the system's goals.

III. Example REAssuRE implementation

We have applied REAssuRE to GridStix [7, 8], an experimental flood warning system. GridStix was a sensor network of smart nodes capable of sensing the state of the river, processing the data and communicating it across the network. Flow rate and river depth data was used by a stochastic model which predicted the likelihood of the river flooding. GridStix acted as a lightweight Grid, capable of distributing tasks such as the processing of the digital camera images used for estimating surface river flow.

GridStix has been decomissioned but a simulator has been constructed that embodies the experience gained and which allows us to run scenarios and simulate events. System behaviour is defined by the configuration of components managed by an adaptive middleware system. Adaptation is achieved by swapping components in and out, binding them dynamically.

Run-time reasoning in REAssuRE focuses on identifying goals that act as *variation points* (e.g. Transmit Data in Fig 1), each of which may be satisfied by two or more i* tasks (e.g. Use Bluetooth and Use WiFi in Fig 1) representing alternative goal realization strategies. Each such task is mapped onto a particular component configuration.

Figure 1 shows one of the i* goal models developed for GridStix. It represents the agent S3, which is responsible for satisfying the goals of GridStix when the stochastic model indicates that the river is about to flood. The figure shows that S3's primary goal is to Predict Flooding, while it has three softgoals Fault Tolerance, Energy Efficiency and Prediction Accuracy. All the softgoals are to some extent in competition. For example, the choice of wireless communication technology entails a trade-off between Fault Tolerance and Energy Efficiency. Bluetooth consumes significantly less energy than Wi-Fi but has a much shorter range. Thus, if Bluetooth is chosen to satisfice the Energy Efficiency goal, Fault Tolerance may suffer because, depending on the physical topology of the network, the failure of a node is more likely to leave its neighbours isolated and unable to communicate than would be the case if S3 had been configured to use Wi-Fi.

S3 represents a situation in which there is a high risk of nodes failing by submersion or water-borne debris so Fault Tolerance is prioritized over Energy Efficiency. Similarly, frequent and accurate data about river flow is required so Prediction Accuracy is also prioritized over Energy Efficiency. Thus Figure 1 shows that the tasks use Wi-Fi, Use FH Topology and Multi Node Image Processing are selected as the goal realization strategies for the variation point subgoals Transmit Data, Organize Network and Calculate Flow Rate, respectively.

Each variation point has a claim attached to one of the tasks' contribution links providing the rationale for the choice of goal realization strategy. Thus, for example, SP too risky for S3 represents an assumption that node failure is likely to lead to fragmentation of the network if a shortest-path (SP) spanning tree is selected, rather than the normally more resilient fewest-hop (FH) spanning tree. The effect of the claim is to Make the Hurt contribution link between the Use Bluetooth task and the Fault Tolerance. As explained above, this in turn has the effect of Break-ing the link, so making the balance of contributions favour the Use WiFi goal realization strategy. Crucially, however, SP topologies aren't always prone to network fragmentation, particularly if the individual nodes fail only infrequently. This uncertainty over whether SP will really lead to poor resilience is what that motivates our making it explicit in the form of a claim.

Claims can derive other claims in a *Claim Refinement Model* (Fig 2) in which claims are organized in a hierarchy. Claims can be AND-ed or OR-ed, allowing the effect of refuted claims to be propagated down the tree to the bottom-level claims; the ones used to annotate softgoal contribution links. Claim

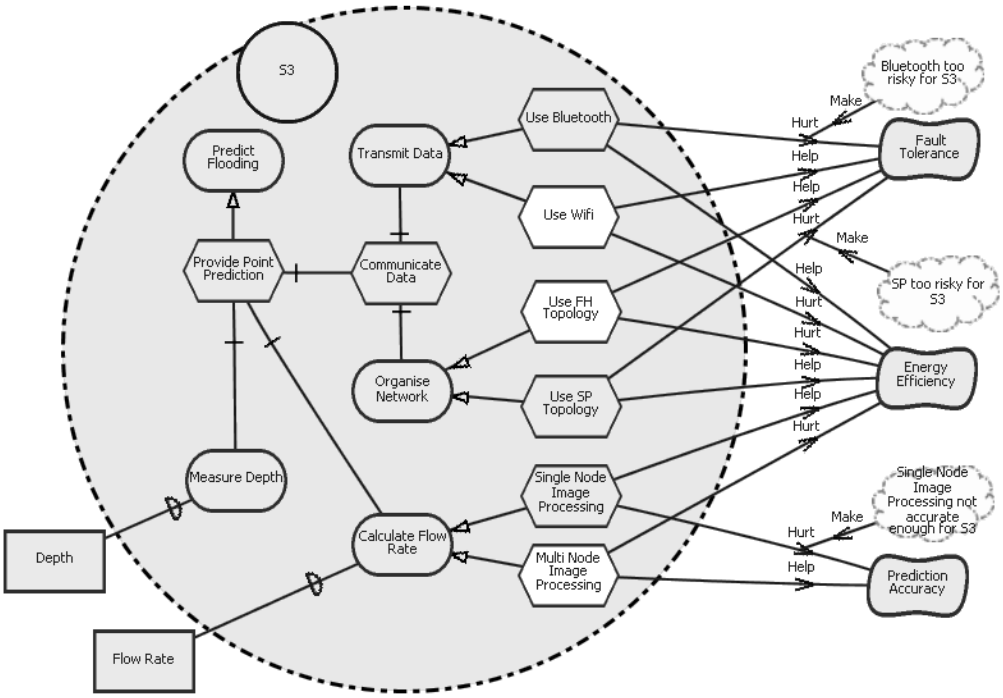

Fig. 1. GrdiStix SR Model

refinement allows the derivation of claims to be made explicit. It also allows high-level claims to be decomposed to primitive claims that can in turn be mapped onto context variables that can be readily monitored.

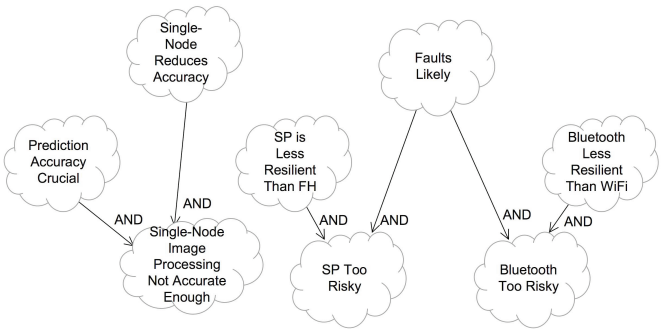

Fig. 2. S3 Claim Refinement Model

To date, we have evaluated the performance of REAssuRE by measuring the network's life; the time taken for fragmentation of the network to reach a point where no result was returned in response to the prevailing river conditions. The results show a consistent improvement in longevity when Gridstix uses claims than when the assumptions made at design time are fixed. The improvement is small, however (typically of the order of a 5% improvement), so further evaluation will be necessary to determine whether the added complexity of the run-time model is justified by the improvement in longevity.

IV. Related Work

DeLoach and Miller [9] explore how to maintain a run-time representation of goals. However, they do not deal with the run-time representation of softgoals or goal realization strategies. The main utility of their work has been for understanding what the systems is doing in terms of goals. No reasoning about partial satisfaction is done. This contrasts with [10], which formalizes a means for representing partial goal satisfaction based on KAOS [11]. A contrasting approach to partial goal satisfaction is taken by RELAX [12]. Although RELAX is not goal-based per-se, [13] illustrates the use of RELAX, with KAOS goal models, using obstacle analysis to identify when to RELAX a goal. [14] propose *adaptive goals* that are aware of their own degree of satisfaction during runtime and a means to trigger adaptation.

All of the above work represents in some way a blurring of the boundary between design and runtime [15, 16] and this is a trend that inspires our own work.

V. Conclusion

REAssuRE is a technique for making explicit where uncertainty underpins design decisions in goal models for self-adaptive systems. Our aim with REAssuRE was to investigate the feasibility of run-time goal models as a means to ensure that adaptations are principled and sensitive to changing context. Its application to GridStix has demonstrated that REAssuRE is able to reason about how design-time assumptions affect goal realization strategies, as evidence for or against design-time assumptions is gathered by claim monitoring. This in turn is used to drive run-time adaptation between alternative

goal realizations as determined by the balance of softgoal trade-offs.

Our intention with REAssuRE was to test whether maintaining and reasoning over goal models at runtime was feasible. The early results are promising. However, there remain many unanswered research questions about the achievement of requirements-aware systems. We need to analyze in detail the results of our initial evaluation and enrich our set of experimental results. We then need to incrementally eliminate the i* modeling restrictions that currently apply to REAssuRE. Ultimately, we will want to propagate the effects of claim refutation up the goal tree, which will involve reasoning over goal satisfaction, with an implication that mitigation of an unsatisfiable goal may require the goal model to be modified.

At the current time, however, REAssuRE represents a first step towards maintaining run-time requirements models on which the system may act. True requirements awareness is still some way off, but our confidence has increased in it being achievable.

ACKNOWLEDGMENT

This research is partially supported by EU FP7 CONNECT project and the Marie Curie Fellowship Requirements@run.time.

REFERENCES

[1] H. J. Goldsby, P. Sawyer, N. Bencomo, D. Hughes, and B. H. Cheng, "Goal-based modeling of dynamically adaptive system requirements," in *15th Annual IEEE International Conference on the Engineering of Computer Based Systems (ECBS)*, 2008.

[2] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein, "Requirements-aware systems: A research agenda for re for self-adaptive systems," *Requirements Engineering, IEEE International Conference on*, vol. 0, pp. 95–103, 2010.

[3] E. S. K. Yu, "Towards modeling and reasoning support for early-phase requirements engineering," in *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE97)*, USA, 1997.

[4] D. Berry, B. 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'05)*, Porto, Portugal, 2005.

[5] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Springer, 1999, vol. 5.

[6] K. Welsh and P. Sawyer, "Requirements tracing to support change in dynamically adaptive systems," in *REFSQ*, 2009.

[7] D. Hughes, P. Greenwood, G. Coulson, G. Blair, F. Pappenberger, P. Smith, and K. Beven, "Gridstix:: Supporting flood prediction using embedded hardware and next generation grid middleware," in *4th International Workshop on Mobile Distributed Computing (MDC'06)*, Niagara Falls, USA, 2006.

[8] N. Bencomo, P. Grace, C. Flores, D. Hughes, and G. Blair, "Genie: Supporting the model driven development of reflective, component-based adaptive systems," in *ICSE 2008 - Formal Research Demonstrations Track*, 2008.

[9] S. A. DeLoach and M. Miller, "A goal model for adaptive complex systems," *International Journal of Computational Intelligence: Theory and Practice.*, vol. 5, no. 2, 2010.

[10] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *Proc. of 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2004, pp. 53–62.

[11] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons, 2009.

[12] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel, "Relax: a language to address uncertainty in self-adaptive systems requirement," *Requirements Engineering*, vol. 15, no. 2, pp. 177–196, 2010.

[13] B. H. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, "Goal-based modeling approach to develop requirements for adaptive systems with environmental uncertainty," in *ACM/IEEE 12th International Conference On Model Driven Engineering Languages And Systems, MODELS'09*, 2009.

[14] L. Baresi and L. Pasquale, "Fuzzy goals for requirements-driven adaptatio," in *18th International IEEE Requirements Engineering Conference, RE'10*, 2010.

[15] L. Baresi and C. Ghezzi, "The disappearing boundary between development-time and run-time," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010.

[16] G. Blair, N. Bencomo, and R. B. France, "Models@ run.time: Guest editors," *Computer*, vol. 42, no. 10, pp. 22–27, 2009.