Third International Conference on Software Testing, Verification, and Validation Workshops

Managing Testing Complexity in Dynamically Adaptive Systems

A model-driven approach

Kristopher Welsh, Pete Sawyer
Computing Department
Lancaster University
Lancaster, UK
[welshk,sawyer]@comp.lancs.ac.uk

Abstract—Autonomous systems are increasingly conceived as a means to allow operation in changeable or poorly understood environments. However, granting a system autonomy over its operation removes the ability of the developer to be completely sure of the system's behaviour under all operating contexts. This combination of environmental and behavioural uncertainty makes the achievement of assurance through testing very problematic. This paper focuses on a class of system, called an m-DAS, that uses run-time models to drive run-time adaptations in changing environmental conditions. We propose a testing approach which is itself model-driven, using model analysis to significantly reduce the set of test cases needed to test for emergent behaviour. Limited testing resources may therefore be prioritised for the most likely scenarios in which emergent behaviour may be observed.

Keywords-component; model-driven; model-directed; testing; autonomous; dynamically adaptive systems

Introduction

Autonomous computing is envisaged by many as Computer Science's ultimate challenge. Systems that are self-managing, self-configuring, self-tuning, self-repairing and self-maintaining are a staple of science-fiction, but complete autonomy is still some distance away. We are, however, starting to create systems that posses a limited ability to perform self-configuration as a means to achieve a degree of autonomy [1] [2]. These so called *self-adaptive* [3], or *Dynamically Adaptive Systems* (DASs) [4] represent an important first step on the road to autonomic systems, and are the subject of much research effort. [5] [6]

McKinley et al. [7] identified two distinct types of self-adaptive behaviour: parametric and compositional. Parametric adaptation is achieved by coding several different system or component behaviours, switched via some internal parameter or flag, that is set according to some pre-defined criteria. This style of adaptation is suitable for small-scale systems that need only to be able to switch between a small number of behaviours, where both the behaviours and the circumstances in which each is to be employed are fully understood when specifying the system. Compositional adaptation is achieved by allowing structural elements of the system to be combined and recombined at run-time. Many current state of-the-art DASs [1] [2] use pre-selected configurations of components, with run-time component substitution taking place in response to changes in the system's operating environment.

Although several architectural approaches to compositional adaptation have been proposed (for example, [6], [8] & [9]), it is more common for DASs to utilise some form of adaptive middleware (for example, [10] & [11]), thus separating the tasks of environmental monitoring, component substitution and state management from the system's business logic. Such adaptive middleware matches component selections with environmental conditions in accordance with adaptation policies, which specify the conditions under which components are substituted.

Recently, researchers have investigated ways to understand and specify the requirements for a DAS, with a consensus emerging that goal-based methods [12] [13] offer a useful means to reason about how competing system goals are traded-off as the environment changes. There are several goal-based modelling approaches for DASs that utilise the popular KAOS [14] and i* [15] techniques ([12], [16] and [17] for example). However, despite the issue of DAS verification being highlighted as needing attention [3], there has been little work on the issue to date. The work that has been completed (e.g. [18]) has focussed on verifying business logic after adaptation, as opposed to our interest: verifying that the adaptation itself is desirable, or better still optimal. This is a key issue since the dynamic behaviour of DASs represents an additional dimension of behavioural complexity over conventional, static systems. This additional complexity impacts on verification of a DAS's behaviour since it is necessary not only to verify that the DAS operates as desired when the environment is in a given state, but that the DAS adapts its behaviour appropriately as the environment changes.

We argue that allowing a system any degree of autonomy over its operation removes a proportionate degree of certainty over how the system will behave. This certainty is key to delivering assured, dependable systems and can only truly be achieved with complete knowledge of what decisions a DAS will make in any given scenario, coupled with a full and accurate understanding of the scenarios that it will face. Unfortunately, both of these prerequisites are extremely difficult to achieve for a DAS. The very nature of the type of environment in which a DAS will prove most beneficial is one of complexity and volatility, and devising

978-0-7695-4050-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ICSTW.2010.57

testing scenarios to cover every permutation of system configurations and environmental properties is not only wickedly complex, but exponentially proportional to the degree of system autonomy. Essentially, a DAS is likely to exhibit emergent behaviour as a result of uncertainty about the environment in which the DAS must operate and uncertainty about how the DAS will respond. Verification of a DAS is therefore not only about assuring that the specified behaviour is delivered, but, critically, also about ensuring that any emergent behaviour is not damaging. It is this latter problem that we are concerned with here.

Of course, testing is not the only means of validating software, and in some cases it may be possible to offer an acceptable degree of assurance through monitoring, or allowing full or partial human control over adaptation. However, such approaches only offer limited assurance, and do so at the expense of autonomy. We argue that even partial testing can offer a greater degree of assurance while still allowing greater freedom in terms of autonomy,

The rest of this paper is organised as follows: Section 2; the next section, examines related work in the DAS research area, with a particular focus on the model-driven engineering of DASs, which this work builds upon. Section 3 proposes a new class of DAS which offers a greater degree of autonomy than the current state of the art. Section 4 proposes a model-driven testing technique that allows limited testing resources to be directed to best offer assurance for this new class of DAS. Section 5 presents a case study to illustrate both the class of system described in section 3 and the testing methodology described in section 4, whilst section 6 concludes the paper.

Background

Our existing LoREM process [12] offers a requirements modelling approach for DASs operating in environments that can be partitioned into distinct *domains*, with a steady-state system, termed a *target system* being devised for each, as conceptualised in [19].

LoREM is based on the i* [15] framework which supports reasoning about systems in terms of agents, dependencies, goals and *softgoals*. Softgoals are a particularly useful concept since they represent goals that can typically be only partially satisfied or *satisficed*. In practice, softgoals often take the form of high-level non-functional requirements (NFRs). Softgoals often represent the foci of decision points since different solution strategies will satisfice different softgoals to varying degrees. Hence, selection of the 'best' solution will typically entail devising several solutions, comparing their softgoal satisficement properties and selecting the one that offers the best trade-off. A strength of i* is that it allows these trade-offs to be reasoned about and recorded explicitly. This proves to be particularly useful for a DAS where the relative priorities assigned to different softgoals may vary depending on which domain the environment is in at any given time.

Accordingly, with LoREM, a separate target system is specified to operate within each domain of the environment, while adaptation from one target system to another as the environment transitions between domains is specified separately for each valid transition.

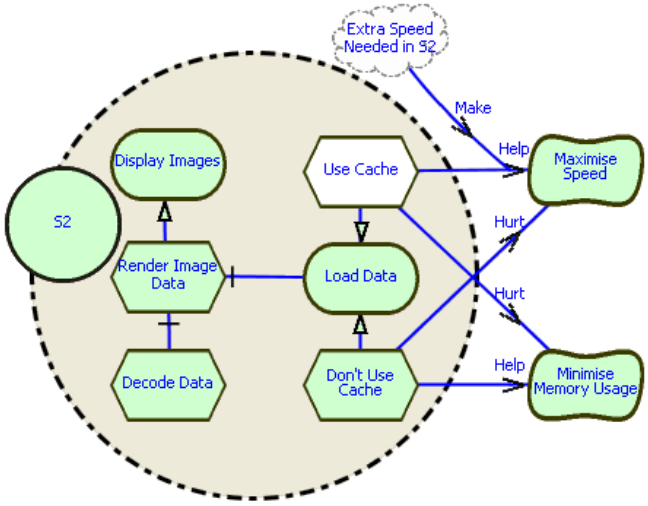

Figure 1. Level 1 Model of Image Viewer's S_2 Target System

In [20] we augmented the LoREM i* models with *claims*, a concept which we have borrowed from the NFR Framework's [21] tool-kit. Claims can be used to record assumptions about the domain or the behaviour of the DAS itself. This is significant because the potential for emergent behaviour is closely linked to the validity of these assumptions.

As a simple example, Figure 1 models one of two target systems for an adaptive image viewer, first described in [22]. The system's only adaptive capability is to introduce an image caching component as the latency encountered when loading files increases beyond a set threshold.

In Figure 1 the circle with the broken line represents the target system S_2 that operates in the domain D_2: High Latency (not shown). Ellipses represent *goals*. In this case, S_2 needs to satisfy the goal "Display Images". Hexagons are *tasks* that operationalise goals. Here, the task "Render Image Data" satisfies the "Display Images" goal, as depicted by the arrow-headed *means-end* relationship arc. "Render Image Data" itself is decomposed (as depicted by the arcs with crossbars) to the task "Decode Data" and the sub-goal "Load Data". "Load Data" is interesting because it can be satisfied by two alternate strategies represented by the tasks "Use Cache" and "Don't Use Cache". Both alternatives are shown to have different impacts on satisficement of the two softgoals "Maximise Speed" and "Minimise Memory Usage" that are depicted using the softgoal symbol: a lozenge. The arcs from the two alternative tasks to the softgoals are contribution links and show (e.g.) that if a cache is used, it *helps* satisficement of the "Maximise Speed" softgoal but *hurts* satisficement of the "Minimise Memory Usage" softgoal. Since both softgoals have one hurts and one helps contribution link for either solution strategy, the choice between them is not clear. However, the cloud attached to one of the contribution links is a claim and its effect is to assert that, in the context of domain D_2, the "Maximise Speed" softgoal should take priority, leading to selection of the "Use Cache" solution.

In the next section we describe how LoREM models can be consulted and modified at run-time. This helps make a DAS tolerant of unanticipated environmental conditions, but introduces the likelihood of emergent behaviour which represents a risk to the integrity of the system. Testing for such emergent behaviour is important but potentially very costly. Fortunately, by using claims to document assumptions about the environment, they can form the focus for testing. As we show in sections 4 and 5, claims can be reasoned about and used to prune the set of test cases that need to be developed.

MODEL-DRIVEN ADAPTIVE SYSTEMS

So far, we have described how we can specify dynamically adaptive behaviour by making assumptions about the environment in which the DAS will operate, and by specifying a target system implemented as a pre-defined configuration of features to operate in each identified environmental domain. Unfortunately, the nature of many environments for which DASs are conceived as the solution are characterised by volatile and poorly understood environments. Recently, however, researchers have started to investigate *models@run.time* [23] as a means to develop DASs. We call a DAS that uses the models@run.time paradigm a Model-Driven Adaptive System, or *m-DAS*. An m-DAS uses run-time abstractions to guide its adaptation. Conventionally, such abstractions occur off-line as models that inform analysis and design. An m-DAS derives its adaptive behaviour from run-time models, and monitors the assumptions upon which they were created. In the event that a monitored assumption no longer holds, it is removed from the model. Monitoring can, of course, continue and the claim may later be reinstated, but given that the system's adaptive behaviour is derived from the models, this model modification can potentially yield modified adaptive behaviour.

We have created an in-memory representation of LoREM models that can be modified and reasoned with by an m-DAS at runtime, and have developed a tool that allows LoREM models developed using the OME modelling tool [24] to be imported into an active system, converting them to this internal representation. We have also developed a tool that allows adaptation policies for the adaptive middleware component framework GridKit [10] to be generated from the in-memory models. The in-memory representation of the LoREM models, along with the tool that loads them, allows a system to analyse and reason about its own model. The policy generation tool allows the policies codifying a DAS' adaptive behaviour to be generated from the models after this analysis and reasoning. Together, the tools allow a GridKit-based m-DAS to load models developed during the system's design and specification phases, reason with them and potentially modify them, and for the m-DAS to derive its adaptive behaviour from the models independently.

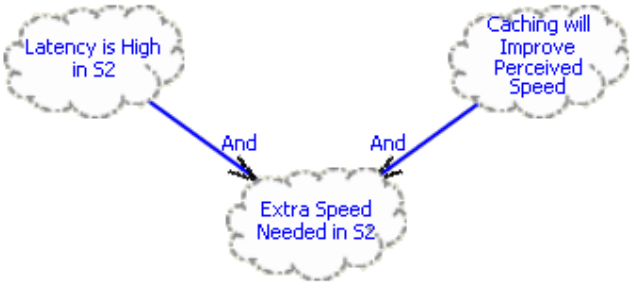

Figure 2. Claim Refinement Model for Image Viewer's S_2 Target System

Monitoring assumptions is analogous to requirements monitoring [25], where the levels of satisfaction for key requirements are monitored. A DAS offers the possibility of modifying the system in response to this data, which an m-DAS achieves via model analysis.

Returning to the adaptive image viewer, Figure 2 depicts a *claim refinement model*. A claim refinement model is an acyclic directed graph that serves as a record of the reasoning behind the claims used to make or break a softgoal contribution link. In short, this decomposition of assumptions allows broad, high-level assumptions to be broken down into smaller, more specific and crucially monitorable supporting assumptions. Figure 2 shows the claim refinement model for the S_2 (high latency) target system. The *bottom-level* claim on the model is "Extra Speed Needed in S_2". Such bottom-level claims are typically derived by combining assumptions and assertions on the environment and the system that, in a claim refinement model, are also modelled as claims. "Extra Speed Needed in S_2" was the claim used in Figure 1 to support the decision to "Use Cache" in S_2. The claim refinement model shows that "Extra Speed Needed in S_2" is derived from the conjunction of the claims "Latency is High in S_2" and "Caching will Improve Perceived Speed". Significantly, the latter claim is monitorable. Converting the adaptive image viewer to an m-DAS would allow the system to reconsider the decision to "Use Cache" in S_2 if monitoring data indicated that the cache was not in fact improving speed, which could occur if images were being accessed non-sequentially, for example.

As illustrated by the preceding example, this extra autonomy potentially allows an m-DAS to overcome deficiencies in the models upon which it is based, such as erroneous assumptions or minor errors made when partitioning the environment into domains. An m-DAS can then tailor its operations to unanticipated or misunderstood domains as it encounters them. However, this additional autonomy comes at a cost in terms of predictability and assurance: with a DAS, it remains possible to guarantee that the system will always take the form of one of its target systems, an m-DAS offers no such guarantee in that any of the target systems could have been modified.

MODEL-DIRECTED DAS TESTING

m-DAS testing can be separated into two categories of activity: testing of the system's business logic and of the system's adaptive behaviour. Testing of the system's business logic is akin to that carried out for traditional, static systems and its commission is assumed. Testing the system's adaptive behaviour seeks to verify that the configurations adopted in given scenarios are optimal, or at the very least not damaging.

To test the adaptive behaviour of conventional DASs, testing scenarios need to be designed for each of the target systems that can be selected, along with scenarios designed to test the system's ability to correctly identify which of the target systems should be used in the expected operating environment. This amounts to a considerable testing burden, but remains within the bounds of feasibility. However, m-DASs offer greater degrees of autonomy than conventional DASs, which allow systems to better tolerate unanticipated changes in their operating environment.

Unfortunately, in order to fully test an m-DAS' adaptive behaviour, it is necessary to identify and understand the system's scope for emergent behaviour completely. To offer this level of testing coverage, it would be necessary to devise scenarios for every possible combination of assumptions holding and being broken, with each combination of assumptions essentially representing a testing scenario at the modelling stage. This means that to find the number of additional testing scenarios for all potential modifications to an individual target system: $T=2^n-1$ Where n represents the number of bottom-level claims (i.e. claims that make or break softgoal contribution links) on the behaviour model for a particular target system of the m-DAS. We subtract 1 because the scenario in which all assumptions hold is the original model, in which the system will adopt the original target system design.

This explosion in testing burden may place such an approach beyond the bounds of feasibility for all but the simplest systems, and all but the biggest budgets.

To reduce the number of testing scenarios that need to be devised, we present an approach to direct limited testing resources to certain key scenarios. As with a conventional DAS, it is necessary to design testing scenarios for each of the m-DAS' target systems, along with scenarios to test the boundaries between target systems and the m-DAS' ability to switch between them. Our approach, however, focuses on reducing the number of *additional* scenarios required to test for emergent behaviour in the m-DAS in the event of it departing from the original target system designs.

We define the key additional testing scenarios as the most likely to occur of those that offer the potential to uncover emergent behaviour. We use claim refinement models to assess both the likelihood of a scenario being encountered during the system's operation along with the potential for emergent behaviour to be discovered using model analysis. We sort each claim in the claim refinement model into one of three categories. Sorting is informed by domain expertise possibly supported by the results of simulations or stochastic modelling. The three categories are:

Unbreakable claims. Here, the term "unbreakable" does not necessarily mean that the claim is axiomatic, but refers to the fact that the m-DAS has no means of monitoring, and thus invalidating the claim. These claims can essentially be removed from consideration, given that they will never cause models to be modified at run-time, and thus never trigger unexpected adaptations and associated emergent behaviour.

Qualified claims are monitored by the m-DAS, and could theoretically prove a source of emergent behaviour. However, the domain expert believes that the system will never encounter a situation where the claim is broken.

Uncertain claims, are monitored by the m-DAS, and although the domain expert believes that the such claims are true, they also consider it conceivable that the system may in extreme or unusual circumstances have to invalidate the claims.

Looking at the claim refinement model for the adaptive image viewer's S_2 target system, depicted in Figure 2, we can characterise "Latency is High in S_2" as a qualified claim, given that high latency is D_2's (the domain in which S_2 operates) defining characteristic. The "Caching will Improve Perceived Speed" claim, however, is more dubious. The possibility of monitoring the cache's usefulness by monitoring load-times with the cache and comparing this to the underlying file load and render speed, along with the possibility of the cache incorrectly predicting the next image for pre-loading, means that the claim should be considered uncertain.

Naturally, testing scenarios covering broken uncertain claims are the ones that are most likely to occur. However, by propagating the "unbreakable" label throughout the tree it becomes possible to identify some bottom-level claims that do not have the potential to cause the system to modify the run-time models, and thus offer no potential for emergent behaviour. Essentially, we are proposing that the testing priorities for each of the three groups of claim or underlying assumption are, first, Uncertain claims, then Qualified claims, with no need to consider the Unbreakable claims.

This model-driven procedure removes from consideration assumptions that have no potential to force model modification, which may cause emergent behaviour. By devising testing scenarios for every combination of the remaining uncertain and qualified bottom-level claims; developers test all the scenarios that offer the potential to uncover potentially undesirable emergent behaviour. If even this reduced number of scenarios proves too difficult or costly to construct and test against, qualified claims can also be removed from consideration. Considering only the uncertain claims, developers focus test coverage on the scenarios most likely to occur, but sacrifice a degree of assurance and introduce a danger that, should a claim classification prove incorrect and a qualified claim be broken, that the system may behave unexpectedly. Investigating qualified claims offers a greater degree of assurance, with no inherent danger stemming from claim classification inaccuracy, whilst investigating only uncertain claims reduces testing burden significantly.

Case Study

To illustrate our test case pruning method, we present a conceptually simple m-DAS and work through the model analysis for a single target system. We also present the results of the same analysis for the m-DAS's other target systems to show the kind of testing workload reduction the method can yield in even a modestly sized m-DAS.

GridStix [1] is a system deployed on the River Ribble in North West England that performs flood monitoring and

prediction. The system was originally developed as a conventional compositionally-adaptive DAS but we are in the process of producing an experimental m-DAS based on the original system design. It takes the form of an intelligent wireless sensor network with multiple nodes measuring river depth and flow rate using a variety of sensors, including the analysis of images taken with an on-board digital camera. The GridStix system uses the GridKit middleware [10], which provides the system's adaptive capabilities.

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 data available, the more accurate the prediction.

The environment in which GridStix operates is volatile, as the river is liable to flooding. When the river floods, the nodes are in danger of submersion, and of sustaining significant damage from water-borne debris. As such, the GridStix system needs to be able to maximize its ability to withstand node failure in order to maintain the connectivity of the surviving nodes.

The GridStix nodes have processing capability, which allows processing of the data and execution of flood prediction models on-site with the system acting as a lightweight grid. This enables pictures taken with the on-board digital cameras to be analysed yielding the river's flow rate, which removes the need to use expensive and difficult to maintain ultrasonic sensors. However, the nodes are resource-constrained and some tasks, particularly flow rate calculations, are best performed by distributing computation among the nodes.

Distributing computation has a cost in terms of the power consumed by inter-node communication, which is a serious issue since GridStix's location is remote, and power has to be provided by batteries and solar panels. The GridStix system can communicate using one of two spanning tree algorithms: Shortest Path (SP) or Fewest Hop (FH) to offer lower power consumption or a network more tolerant of node failure, respectively. As such, GridStix has three key conflicting softgoals: "Energy Efficiency", "Fault Tolerance" and "Prediction Accuracy".

When specifying the GridStix system, domain experts partitioned the operating environment into three distinct 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 suddenly flood. The D_3 (Flood) domain occurs when the depth increases and the river is about to flood, which means that the nodes are in imminent danger of failure.

For each domain, a target system was devised, with target system S_1 tailored for domain D_1, S_2 for D_2 and S_3 for D_3 respectively, as depicted in Figure 3.

The LoREM models for the system have previously been published in full [12]; and due to space constraints this section will focus only on those models key to illustrating our test scenario pruning technique: the Level 1 Behaviour

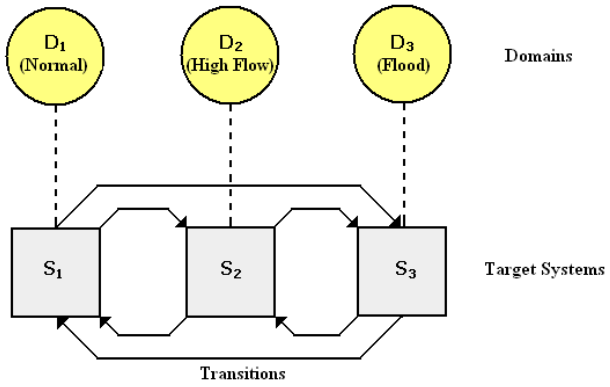

Figure 3. Mapping Between GridStix Domains and Target Systems

models. The Behaviour model for S_3 (Flood) is depicted in Figure 4.

The key goal in Figure 4 is for the target system (S_3) to "Predict Flooding". This goal is achieved by completing the task: "Provide Point Prediction" which can be further decomposed into the goals: "Measure Depth", "Calculate Flow Rate" and "Communicate Data". The "Communicate Data" goal depends on the "Transmit Data" goal being achieved for its own satisfaction, as represented by the "D" dependency link between the two. The "Measure Depth" and "Calculate Flow Rate" goals produce "Depth" and "Flow Rate" resources respectively, which are used in other LoREM models, which are beyond the scope of this case study.

The "Calculate Flow Rate", "Communicate Data" and "Transmit Data" goals all have several alternative methods of satisfaction, represented by tasks connected to the respective goals with means-end links.

The three key conflicting softgoals are represented by lozenges on the right of Figure 4, and the potential impact of selecting individual tasks to satisfy goals with several alternatives on each of the softgoals are represented by contribution links. In i* there are 7 types of contribution link indicative of impact magnitude. Figure 4, however uses only 2 of these contribution links: help and hurt to capture a moderate strength positive and negative impact, respectively. The tasks selected to satisfy goals in the S_3 target system are coloured white for clarity.

Attached to the contribution links in Figure 4 are three claims, represented by clouds on the diagram. The claims make or break the softgoal contribution links to which they are attached. *Makes* indicates that the contribution link was lent special credence when reaching the selection decision for this target system, whereas *breaks* indicates the contribution link was overlooked or dismissed. These claims are so-called bottom-level claims, whose basis can be examined in the associated claim refinement model. The claim refinement model for the S_3 target system is depicted in Figure 5.

The three bottom-level claims shown in Figure 4 appear at the bottom of Figure 5, connected to their supporting claims with contribution links. For example, it is possible to establish from Figure 4 that Wi-Fi was selected over

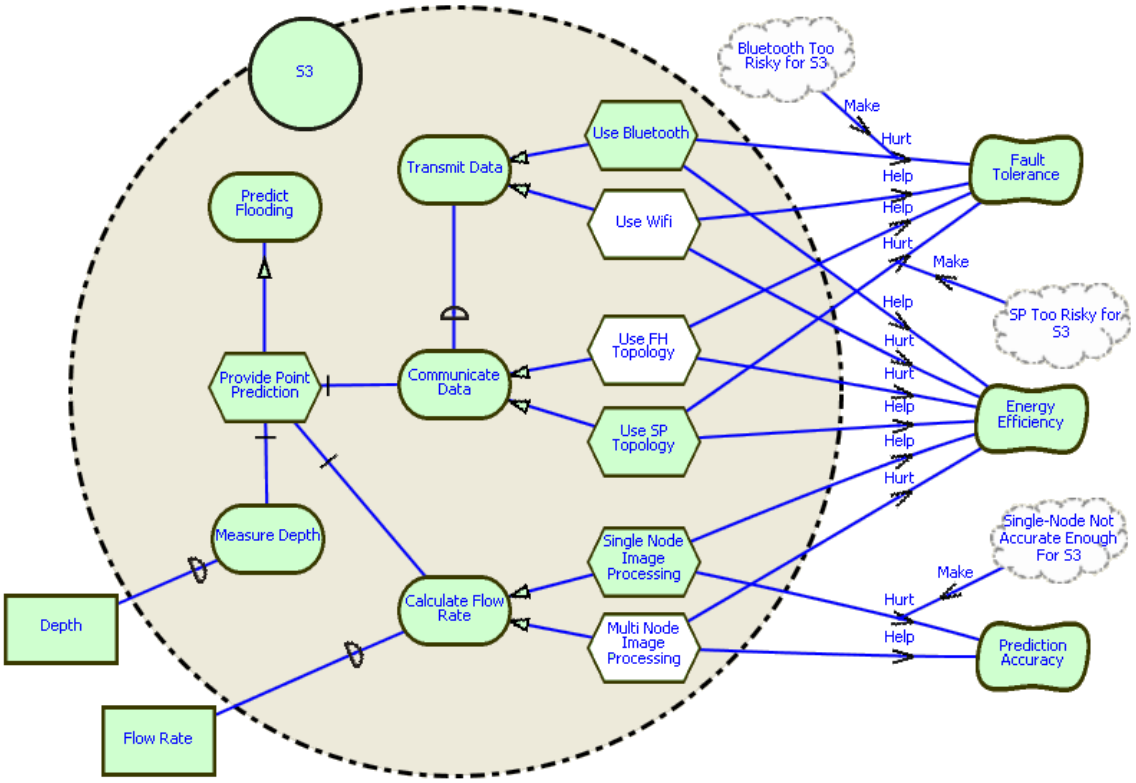

Figure 4. GridStix Behaviour Model for S_3 (Flood)

Bluetooth to satisfy the "Transmit Data" goal in S_3 because Bluetooth was considered too risky in terms of Fault Tolerance. Examining Figure 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.

Although the bottom-level claims in Figure 4 aren't directly monitorable by an m-DAS at run-time, some of their supporting claims are. Thus, if the m-DAS discovers at run-time that when nodes are submerged, Wi-Fi communication proves no more feasible than Bluetooth, the "Bluetooth too risky for S_3" claim can be invalidated, and the system can reconsider the decision to use Wi-Fi over Bluetooth in this target system. Using Bluetooth in this instance would allow the system to better satisfice its "Energy Efficiency" softgoal without further harming "Fault Tolerance", for which it has no means of satisfaction.

There are 3 bottom-level claims in Figures 4 & 5, and if we were to attempt to devise testing scenarios for every combination of claims we would be attempting to create 7 additional scenarios, not all of which would be feasible or even necessarily possible. The scenario in which all claims

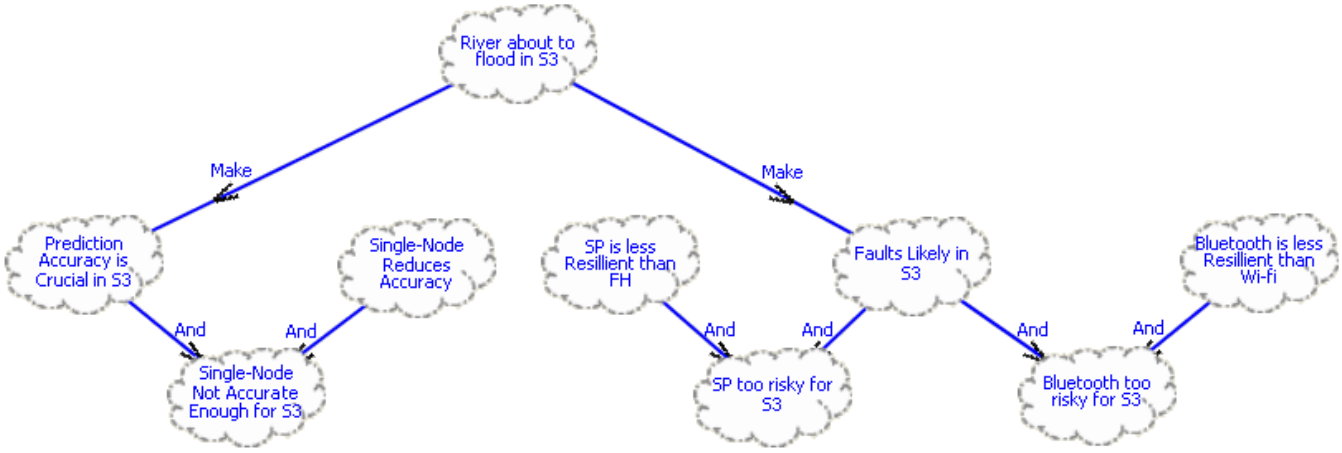

Figure 5. GridStix Claim Refinement Model for S_3 (Flood)

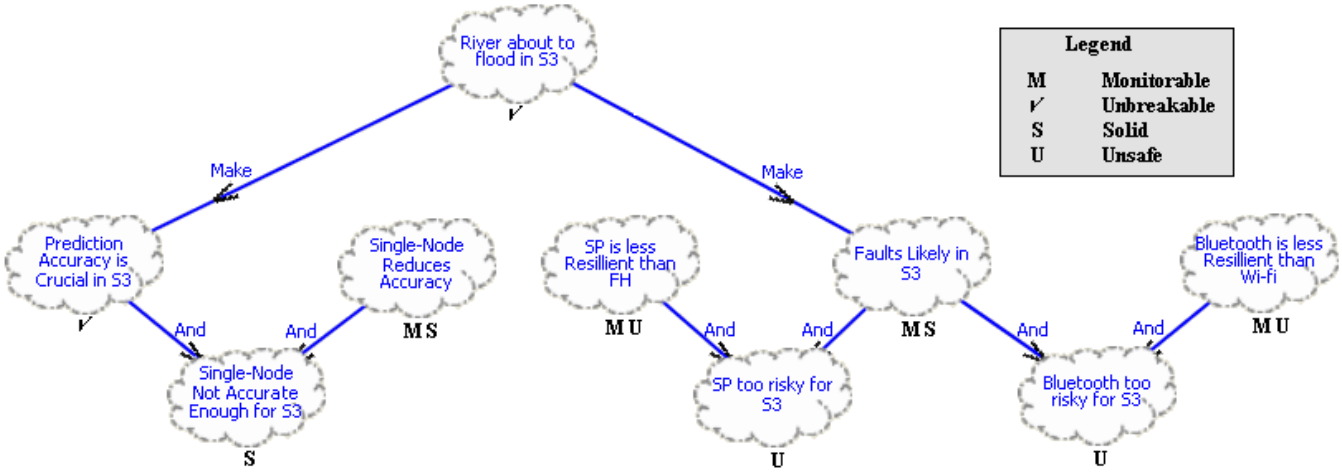

Figure 6. GridStix Claim Refinement Model for S_3 (Flood) Annotated with Claim Classifications

hold is the anticipated target system, for which appropriate testing scenarios would be created as a matter of course.

An analysis of all the claims by a domain expert yielded four monitorable claims, of which two were classed qualified, and two uncertain.

The analyst had qualified confidence in the "Single Node [Image Processing] reduces accuracy" claim because the time taken to perform image analysis on a single node is significant, and there is a possibility of the river's state having changed significantly in the meantime. Given the obvious possibility of nodes becoming submerged or damaged when the river floods, the "Faults Likely in S_3" claim was likewise classed as qualified.

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.

Propagating the "unbreakable", "qualified", and "uncertain" labels throughout the tree depicted in Figure 5, it becomes apparent that two of the three bottom-level claims could be impacted by an uncertain claim being proven false at runtime, thus offering the potential for emergent behaviour. The "Single Node Reduces Accuracy" claim, however, derives qualified confidence from its supporting claims.

In the entire GridStix system, only one of the bottom-level claims was assessed to be directly monitorable, which is understandable given their broad nature. However, a significant number of bottom-level claims could be monitored indirectly, via monitoring of their supporting claims and analysis of the claim refinement model. In Figure 6, the claim refinement model from Figure 5 is annotated to show the unbreakable, qualified and uncertain claims. The claims in Figure 6 that are labelled as either qualified or uncertain that are not also labelled monitorable are these indirectly monitorable claims.

As figure 6 shows, for the S_3 target system, there are no bottom-level claims classified as unbreakable, one considered qualified and two considered uncertain. Creating test scenarios for all combinations of uncertain claims, that is the scenarios with greatest potential to uncover emergent behaviour, would require only 2^2-1 (i.e. 3) scenarios, as opposed to the original 2^3-1 (i.e. 7) scenarios.

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 I.

TABLE I. 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 (5), which is representative of the complexity of the trade-offs in system design made for this domain. The S_1 domain prioritises "Energy Efficiency" particularly highly, with a delicate balance needing to be struck between it and the other, conflicting, softgoals. In [19] we suggested that the rationale recorded with claims should be as close as possible to that actually used, to maximise the technique's benefit in terms of traceability. From a monitoring and testing perspective, recording the same rationale, albeit imprecisely, using fewer claims would be advantageous. Hence, there is a tension between the needs of tracing and testing with the number of bottom-level claims used in S_1 reflecting a bias towards tracing. Although removing unbreakable claims from consideration in this domain seems particularly effective, perhaps due to this tracing bias, we would consider the second two domains more indicative of the scenario reduction method's performance in general.

Table II shows the numbers of additional testing scenarios needed to test for emergent behaviour in each (potentially modified) target system for all claims, qualified claims and uncertain claims respectively. This shows that using our strategy, the number of testing scenarios for S_1, even though it has 5 bottom-level claims, could be restricted to 3 (testing for uncertain claims only) or 7 (qualified and uncertain claims).

TABLE II. EMERGENT BEHAVIOUR TESTING SCENARIOS FOR GRIDSTIX SYSTEM

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 testing scenarios for all combinations of uncertain claims would require 9 scenarios, and for the qualified and the uncertain claims would require 17 scenarios. Devising scenarios for all combinations of claims in the GridStix system would require 45 additional scenarios. By devising testing scenarios for the 17 combinations of qualified and uncertain claims, all the claim-monitoring related potential sources of emergent behaviour are covered. This is of course preferable, but if infeasible, creating the 9 scenarios derived from uncertain claims will still offer some degree of assurance, although limited by the accuracy of the claim classification.

The GridStix system is only modestly complex from a modelling perspective, and the environment in which it operates although challenging for the system can be partitioned into a small number of domains. As such, the numbers of testing scenarios remains fairly low. However, the case study has shown that our model analysis can offer a real reduction in the number of test scenarios for an m-DAS' potentially modified adaptive behaviour without sacrificing assurance, or alternatively can trade a degree of assurance for a dramatic reduction in testing workload.

Conclusions

Designing any degree of autonomy into a system will have a negative impact on the degree of assurance it is possible to afford it, and a corresponding increase in the cost of testing the system to achieve this assurance. The more autonomy a system has, the more pronounced this assurance deficiency and testing challenge will become. This key trade-off is often overlooked by researchers looking at ways to specify, design and build autonomous systems.

Dynamically Adaptive Systems are a notable class of system, representing a first step on the road to fully autonomous systems. Researchers have been making progress in developing methods to specify, design and build DASs but the thorny issues of testing and delivering assurance have been relatively neglected to date. Testing workload in DASs multiplies with the number of target systems and potential transitions between them, and this complexity explosion will only become more pronounced as systems with greater autonomy are conceived.

The class of Dynamically Adaptive System which we call an m-DAS load representations of their design-time models into memory at run-time. Such systems use these run-time models to not only guide adaptation decisions and to derive adaptation logic, but also to monitor the assumptions on which the models were constructed. By re-evaluating the models in the event of an assumption being proven false, it becomes possible for an m-DAS to adopt a configuration that was unforeseen at design time but which best satisfies the system's requirements in circumstances that were similarly unforeseen. This sub-class of DAS has a greater degree of autonomy than previous DASs, and amplify the difficulty in testing over and above that seen in traditional DASs.

We have proposed a method of model-directed testing that addresses the problem of scalability in testing the adaptive behaviour of DASs, and m-DASs in particular, by identifying the scenarios most likely to uncover emergent behaviour. The method allows developers to achieve assurance when testing all scenarios is simply infeasible.

Even using this method, the testing workload for DAS, or m-DAS in particular is still considerable, and far greater than that for a traditional, static system. This additional complexity is essentially another facet of the inherent complexity in a DAS, which acts to impede the economic selection of a DAS or m-DASs solution in situations where a traditional, static system is feasible even if not perfectly suited.

Our testing reduction strategy is, of course, dependent on the quality of the assessment made by domain experts or other means of analysis of which assumptions are risky. A monitored assumption that is proven false at run-time, but classed as qualified when designing testing scenarios still has the potential to introduce unexpected emergent behaviour if only the uncertain scenarios were devised and tested against. Hence, there is a distinct preference for analysing both the qualified and the uncertain claims. For an m-DAS, it would be possible to prevent this emergent behaviour at the expense of the system's autonomy and ability to adapt to these circumstances even without creating the additional scenarios, by removing the monitor for assumptions considered safe. This, of course, is a refinement of the original autonomy vs. assurance trade-off, and is a decision best left to individual system designers.

References

[1] 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.

[2] A. Cerpa and D. Estrin "ASCENT: Adaptive self-configuring sensor networks topologies," IEEE Trans. Mobile Comput., vol. 3, page. 272, Jul./Aug. 2004.

[3] B.H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G.M. Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H.M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H.A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," Software Engineering for Self-Adaptive Systems, Springer-Verlag, 2009, pp. 1-26.

[4] H. Goldsby, D. Knoester, and B. H. C. Cheng, Digitally evolving models for Dynamically Adaptive systems. In SEAMS '07: Proceedings of the ICSE Workshop on Softwate Engineering for Adaptive and Self-Managing Systems, Minnesota, USA, 2007.

[5] P.-C. David and T. Ledoux. Towards a framework for self-adaptive component-based applications. In DAIS '03 Proceedings of the International Conference on Distributed Applications and Interoperable Systems, Paris, France, 2003.

[6] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl and P. Steenkiste. Rainbow: architecture-based self-adaptation with reusable infrastructure. Computer, vo.l 37 no. 10, pages 46-54, Oct 2004.

[7] P.K. McKinley, S. M. Sadjadi, E. P. Kasten and B. H. C. Cheng. A taxonomy of compositional adaptation. Technical Report MSU-CSE-04-17 Department of Computer Science and Engineering, Michigan State University, 2004.

[8] J. Kramer J. and J. Magee Self-managed systems: an architectural Challenge. In Future of Software Engineering. International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 259-268, 2007.

[9] S. White, J. Hanson, I. Whalley, D. M. Chess, J. Kephart, An architectural approach to autonomic computing, IN ICAC'04: Procedings of the 1st International Conference on Autonomic Computing, 2004.

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

[11] M. Roman, F. Kon, and R. H. Campbell. Reflective middleware: From the desk to your hand. IEEE DS Online, Special Issue on Reflective Middleware, 2(2), 2001.

[12] H. Goldsby, P. Sawyer, N. Bencomo, B. H. C. Cheng, and D. Hughes. Goal-Based modelling of Dynamically Adaptive System requirements. In ECBS '08: Proceedings of the 15th IEEE International Conference on Engineering of Computer-Based Systems, Ireland, 2008.

[13] J. Zhang and B. H. C. Cheng. Model-based development of dynamically adaptive software. In ICSE '06: Proceedings of the 28th international conference on Software engineering, pages 371–380, New York, NY, USA, 2006. ACM Press.

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

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

[16] B. H. C. Cheng, Pete Sawyer, Nelly Bencomo and Jon Whittle. A goal-based modelling approach to develop requirements of an adaptive system with environmental uncertainty. In MODELS 09: Procedings of IEEE 12th International Conference on Model Driven Engineering Languages and Systems. Colorado, USA, 2009.

[17] G. Brown, B. H. C. Cheng, H. Goldsby, J. Zhang. Goal-oriented specification of adaptation requirements engineering in adaptive systems. In SEAMS '06: Proceedings of the International Workshop on self-adaptation and self-managing systems. Shanghai, China, 2006.

[18] J. Hielscher, R. Kazhamiakin, A. Metzger, and M. Pistore, "A Framework for Proactive Self-adaptation of Service-Based Applications Based on Online Testing," Proceedings of the 1st European Conference on Towards a Service-Based Internet, Madrid, Spain: Springer-Verlag, 2008, pp. 122-133.

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

[20] K. Welsh and P. Sawyer. Requirements tracing to support change in Dynamically Adaptive Systems. In REFSQ '09: Proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Qulity, Netherlands, 2009.

[21] L. Chung, B. Nixon, E. Yu and J. Mylopolous. Non Functional Requirements in Software Engineering. Kluwer Academic Publishers. 2000.

[22] A. Lapouchnian, S. Liaskos, J. Mylopoulos, and Y. Yu. Towards requirements-driven autonomic systems design. In DEAS '05: Proceedings of the 2005 Workshop on Design and Evolution of Autonomic Application Software, pages 1-7, St. Louis, MO, USA, 2005.

[23] G. Blair, N. Bencomo, and R. France. Models@runt.time, IEEE Computer, October 2009.

[24] University of Toronto. Organizational Modelling Environment. Available at: http://www.cs.toronto.edu/km/ome/

[25] D. Cohen, M. S. Feather, K. Narayanaswamy, and S. S. Fickas. Automatic monitoring of software requirements. In ICSE '97: Proceedings of the 19th International Conerence on Software Engineering, pages 602–603, Boston, Massachusetts, United States, 1997.