Deriving Adaptive Behaviour from *i** models

Kristopher Welsh, Pete Sawyer

Lancaster University, Computing Dept., Infolab21 LA1 4WA Lancaster, UK
{k.welsh, p.sawyer}@lancs.ac.uk

Abstract. Dynamically Adaptive Systems (DASs) adjust their behaviour at run-time to tolerate changes in context. With potentially incomplete or inaccurate knowledge of an operating environment, tailoring specific behavioural adjustments to individual contextual changes is a time-consuming and error-prone process. i* models of a DAS' behavioural adjustments can aid understanding, and can form part of the specification of the DAS' adaptive behaviour; speeding the specification process. This paper presents an approach by which a DAS' adaptive behaviour may be derived directly from a set of i* models, and a tool capable of performing the derivation automatically.

Keywords: Dynamically Adaptive Systems, Istar, Model-Derived

1 Introduction

Dynamically Adaptive Systems monitor changes in their operating environment, and re-configure to better suit prevailing conditions. We have previously [1] likened this to the balancing of conflicting softgoals as the environment dictates changes in what constitutes an acceptable balance between them. DASs commonly utilise some form adaptive middleware (for example [2]), which separates the concerns of environmental monitoring, adaptation planning and effecting component substitution from the DAS' business logic. Adaptive middleware codifies system configurations (as combinations of components) and the conditions under which they are adopted in adaptation *policies*. These can be thought of as re-statements of decisions made after analysis of the environment, and of the available system components during the Requirements Engineering process.

Our existing LoREM process [3] offers an i*-based modelling approach for DASs operating in environments that can be partitioned into distinct *domains*. An individual steady-state system, termed a *target system* is derived for each domain, as conceptualised in [4]. From a modelling perspective, each target system is as complex as a traditional, non-adaptive system developed for the domain, with the key artefact to emerge from the modelling process being a selection of components to be adopted in each domain. This paper explores a method by which these selections of components, (modelled in i* SR diagrams) combined with i* models of environmental partitions can be used to derive the policies used by the system's adaptive middleware to control the DAS' adaptive behaviour.

2 Objectives of the Research

Given our belief that adaptation policies re-state the results of component selection decisions taken during the RE process, and that those decisions are recorded in i* models in LoREM, the research has three aims: **1.**To establish a link between the i* based environmental modelling carried out as part of the LoREM process and DAS adaptation policies. **2.** To develop an approach by which the adaptation policies can be derived from the i* models and **3.** To automate the process.

Understanding the link between the LoREM i* models and a DAS' final adaptation policies is important in pinpointing modelling deficiencies and performing maintenance on the DAS. Given the complex nature of the environments for which a DAS will typically prove most useful, understanding may be incomplete or inaccurate. Being able to trace a sub-optimal or failing target system back to the environmental analysis is crucial to a time and cost efficient evolution process.

The ability to derive adaptation policies directly from LoREM models could reduce the time and effort spent in their development. DASs are hugely complex systems, and any support mitigating some of this complexity is beneficial.

Automating the derivation process not only saves time and helps to reduce the possibility of errors being made during policy derivation, but opens up an interesting possibility: the DAS may be able to perform the derivation itself. A combination of this ability, requirements monitoring and models@runtime [5] could allow a DAS to perform some limited self-maintenance, in the form of correcting modelling deficiencies in response to monitored data, and re-creating its adaptation policies based on the updated model. This possibility is discussed further in section 5.

3 Scientific Contributions

The LoREM process involves creating i* SR models of each target system, illustrating the degree to which it satisfices the DAS' softgoals in each domain. These models are known as level 1 models. Level 2 models, show how the DAS' adaptive infrastructure monitors the environment, plans and effects adaptation for each valid transition between target systems. Level 3 models aid in the selection of the adaptive infrastructure, but are beyond the scope of this paper.

In [6] we augmented the level 1 i* models with NFR framework [7] claims, which are used to record assumptions about the domain, or the behaviour of the DAS itself. Claims are attached to contribution links, either making or breaking the attached link. A made contribution link (of any i* type) is lent special credence in the decision-making process, whereas a broken link's contribution is disregarded. This differs from the use of fine-grained contribution links in that a claim speaks to the importance of a contribution, whereas fine-grained contribution links speak to its magnitude. Claims differ to i* beliefs too, in that a belief is held by an actor to be true, with no presumption of truth by the analyst. Claims allow us to strike different softgoal balances for different target systems, even if softgoal contributions are unchanged.

To allow us to demonstrate our policy derivation method, we present a conceptually simple DAS first presented in [8]. The adaptive image viewer was designed as a pedagogical example, and its sole adaptive capability is to introduce a caching component as the latency encountered in loading files increases beyond a set threshold. The system's operating environment can be easily divided into 2 domains: low (D_1) and high (D_2) latency. For each domain, a target system is modelled: S_1 and S_2 respectively. Figure 1 shows the level 1 i* models for each target system.

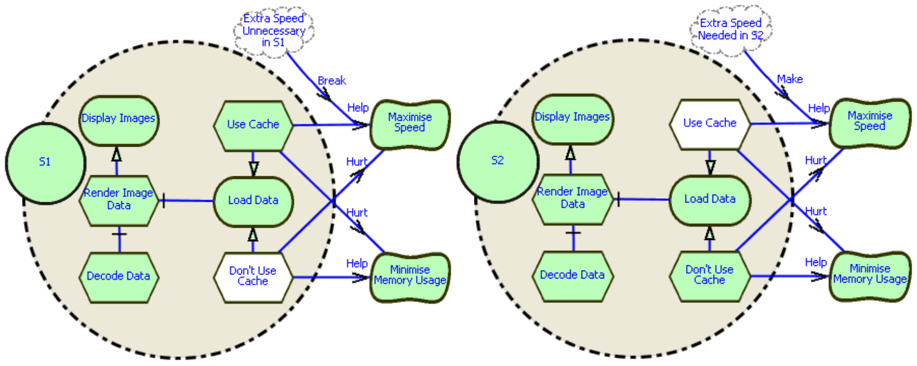

Figure 1. LoREM Level 1 models for Image Viewer S_1 and S_2 Target Systems

As Fig. 1 shows, the only difference between the two target systems is the means by which the "Load Data" goal is satisfied. With two conflicting softgoals: "Maximise Speed" and "Minimise Memory Usage", the system opts not to waste memory using a cache in the low latency domain, but tries to prioritise speed in the high latency domain. The claims on the models explain the rationale behind the selection decision, despite the contributions of the “[Don't] Use Cache” tasks remaining constant.

By comparing the two level one models in Fig. 1, it is possible to infer the component substitutions involved in transitioning from S_1 to S_2 and vice-versa. The precise component (or class) names associated with each selectable task can be placed in a lookup table when generating policies. Noting which component substitutions are necessary to transition between target systems is the first step in deriving the policies.

Figure 2 shows the level 2 i* model for the S_1-S_2 transition. A similar model is produced for each valid transition, showing the three roles of a DAS' adaptation infrastructure. The monitoring mechanism observes the environment, providing data to the decision-making mechanism, which identifies when the environment switches from one domain to another and triggers adaptation. The Adaptation mechanism performs the component substitutions needed to adopt the appropriate target system.

To derive adaptation policies, the two key elements of the level 2 model are the transition being triggered, and the trigger itself. The transition is represented by the “Adapt from S_1 to S_2” task on Fig. 2, and the trigger by the “Fire HIGH_LATENCY event” task. As with the level 1 models, a lookup table can be used to associate a specific class with the "Fire HIGH_LATENCY event" task. Identifying these two elements is the second step in deriving adaptation policies.

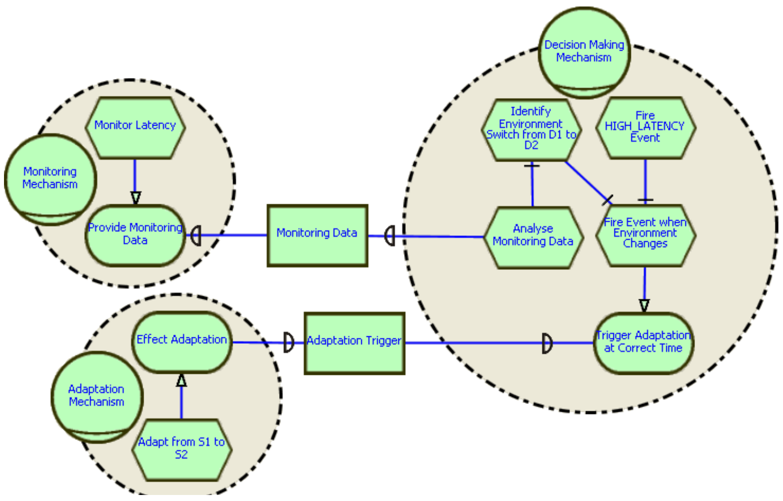

Figure 2. LoREM Level 2 model for Image Viewer S_1-S_2 Transition

From Fig. 2, we can see that the S_1-S_2 transition is triggered by the HIGH_LATENCY event, and by comparing the two level 1 models in Fig. 1, we can see that the only reconfiguration necessary is to swap the "Don't Use Cache" image loader with the "Use Cache" one. Hence, Fig. 3 shows the relevant portion of an adaptation policy compatible with the GridKit adaptive middleware [2].

```
<ReconfigurationRule>
   <FrameWork>Cache</FrameWork>
   <Events><Event><Type>HIGH_LATENCY</Type><Value/></Event></Events>
    <Reconfiguration>
        <FileType>Java</FileType><Name>Reconfigurations.Cache</Name>
    </Reconfiguration>
</ReconfigurationRule>
```

Figure 3. Snippet from Image Viewer Adaptaion Policy

The rule shown in Fig. 3 is a snippet taken from a full adaptation policy generated by the tool created to automate the process. The tool operates on LoREM level 1 and 2 models created using the Organisation Modelling Environment (OME) i* modelling tool . The tool can be adjusted to produce policies for other adaptive infrastructures, and operates on the OME tool's saved models.

4 Conclusions

DASs are a notable class of system, representing a first step on the road to fully autonomous systems. The complexity of DASs, and the environments for which they are conceived presents a problem of scale to the software engineering process. The i* based LoREM process offers a way to model and specify DASs where the environment can be partitioned. This work builds upon LoREM, speeding the development of adaptation policies, aiding DAS implementation and maintenance.

We have demonstrated a link between a DAS' level 1 and 2 i* models and the system's adaptation polices. It is possible to derive the policies automatically, directly from models created with the OME i* modelling tool. Confirming this link not only improves support for policy creation, but allows sub-optimal adaptive behaviour to be traced back to the environmental understanding that led to its specification.

5 Ongoing and Future Work

The LoREM process is applicable only to DASs in partitionable environments, which is not always the case. We are considering ways in which other environments may be better supported by similar processes. Within the LoREM process, we are examining ways in which i* models can be used to validate developed DAS behaviour.

Perhaps the most important possibility opened up by this work is (as mentioned in section 2) is that of a DAS deriving its own adaptation policies from models at runtime. For this to be useful, the models would also need to be modifiable. By monitoring system performance and the environment it may be possible to identify modelling deficiencies automatically. In response, the models could be modified by the DAS, which would then re-derive its adaptation policies, thus adjusting its behaviour to fit the new models. Systems performing this kind of adaptation could be described as self-maintaining or self-tuning, and are the subject of ongoing work.

References

1. Welsh, K., Sawyer, P.: When to Adapt? Identification of Problem Domains for Adaptive Systems. Requirements Engineering: Foundation for Software Quality. pp. 198-203 (2008).
2. Grace, P., Coulson, G., Blair, G., Mathy, L., Yeung, W.K., Cai, W., Duce, D., Cooper, C.: GRIDKIT: Pluggable Overlay Networks for Grid Computing. 1463--1481 (2004).
3. Goldsby, H.J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Hughes, D.: Goal-Based Modeling of Dynamically Adaptive System Requirements. Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems. pp. 36-45 IEEE Computer Society (2008).
4. Berry, D.M., Cheng, B.H.C., Zhang, J.: The four levels of requirements engineering for and in dynamic adaptive systems. 11th International Workshop On Requirements Engineering Foundation For Software Quality (REFSQ). (2005).
5. Blair, G., Bencomo, N., France, R.B.: Models@ run.time, (2009).
6. Welsh, K., Sawyer, P.: Requirements Tracing to Support Change in Dynamically Adaptive Systems. Proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 59-73 Springer-Verlag, Amsterdam, The Netherlands (2009).
7. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering. Springer (2000).
8. Lapouchnian, A., Liaskos, S., Mylopoulos, J., Yu, Y.: Towards requirements-driven autonomic systems design. SIGSOFT Softw. Eng. Notes. 30, 1-7 (2005).