Hybridising events and knowledge in an infrastructure for context-adaptive systems

Simon Dobson
Systems Research Group
School of Computer Science and Informatics
University College, Dublin IE

simon.dobson@ucd.ie

Abstract

Event-based systems are a popular substrate for distributing information derived from sensors to be used in driving adaptive behaviour. We argue that event systems only provide a poor model of context, and that a hybrid approach that uses events to populate and maintain a knowledge base provides a more stable solution. The inherent uncertainties imply that traditional knowledge-based system techniques are extended to deal with more uncertain reasoning. We discuss our plans for additional work in analysing and programming autonomic behaviours with this architecture.

1. Introduction

Autonomic systems are intended to adapt to their environment in a way that optimises performance, robustness and other features without requiring extensive human intervention. The challenges arise from the need to deal with complex and uncertain information about the environment, and to match this to appropriate changes in system behaviour.

In this position paper we describe the motivations for our current work within SRG on infrastructures and languages for adaptive systems. We argue for hybrid approaches, using an event-based infrastructure to drive and maintain a knowledge base. The resulting system may be programmed in both event-based and knowledge-based terms, allowing a range of approaches to adaptation to be explored.

Section two describes current approaches to building adaptive systems, highlights some deficiencies and argues for a hybrid model that combines event- and knowledge-based approaches. Section three briefly discusses some issues in programming such hybrids, while section four concludes with some directions for future work.

2. Context, events and knowledge

Designs for autonomic systems draw their inspiration from a number of sources. Prominent among these are biologically-inspired systems built around stygmergy or swarm intelligence, where simple individual responses to stimuli are aggregated to produces a global result[Bonabeau99]. At the other extreme are attempts to model adaptive systems in a closed-form way that allows more precise characterization of their behavioural envelopes[Dobson04]. The former relies on ideas from control theory, while the latter draws more on pervasive computing, continuous mathematics and AI.

The context of a system captures the environment in which it operates, including all "additional" or "non-functional" aspects that, while not being "core" to the system's behaviour, nevertheless affect the way in which that behaviour should be optimised. Pervasive computing systems are good representatives of the class of adaptive systems whose adaptations are constrained by their surrounding environment.

The Context Toolkit[Salber99] is the canonical example of programming pervasive applications based on events. Such systems consist of a number of adapters or *contextors*[Coutaz02], each capturing some

aspect of the environment such as the temperature, or the reading from a location sensor. The advantage of such systems is that it is straightforward to construct both the infrastructure and the adapters; the disadvantage is that they place a large load on the developer to build a sufficiently flexible decision-support system to drive adaptative behaviour.

Why events and adaptation don't match

To understand the problem of using events directly, consider the following scenario. Suppose we have two people, `A` and `B`, together with a room `R`. Two events are defined, `enter(a,b)` and `leave(a,b)`, indicating that entity `a` has entered (or left, respectively) place `b`. These events are to be used to drive a system that will adapt its behaviour when `A` and `B` and in `R`. We use angular brackets to denote event traces: given events `e1`, `e2`, and `e3` we use `<e1,e2,e3>` to denote the sequence of events occurring in the order given and `<e1,…,e2>` to denote `e2` occurring after `e1` with zero or more events in between.

In the simplest model there are two possible event traces that can bring the desired situation about: `<enter(A,R),…,enter(B,R)>` or `<enter(B,R),…,enter(A,R)>`. On observing either of these event traces the system may adapt.

The problem, however, is that this approach is only stable given three key assumptions. The first is that events cannot be "counteracted" by other events. Suppose we observe the event trace `<enter(A,R),…,enter(A,S),…,enter(B,R)>`. Does `A` entering `S` mean that `A` is no longer in `R`? – in other words, are `R` and `S` disjoint spaces? This cannot be definitively answered without an understanding of the spatial relationships involved.

Furthermore in a open system we might introduce new events which interact with existing events in unforeseen ways. Introducing an event `leave(a,b)` (with the obvious intention) means that an event trace such as `<enter(A,R),…,leave(A,R),…,enter(B,R)>` is also not a valid trigger for adaptation.

The second problem concerns triggers that rely on a correspondence between events. Suppose we see the event trace `<enter(A,R),…,enter(B,R),…,enter(B,R)>` – what do we conclude? Should the second `enter(B,R)` event be considered a duplicate, an error, or the start of another trigger for which we should wait for a corresponding `enter(A,R)` event?

This leads directly to the third problem. Event systems were developed from process algebra which in turn describes processes that might be termed *exact*: the events that occur are assumed *actually to have occurred*. The problem with many pervasive (and other) systems that have a close connection to the real world, for example by way of sensors, is that the processes they are engaged in are *inexact*: the events may be noise.

It seems intuitively likely, absent any intervening `enter()` or `leave()` events, that the second `enter(B,R)` event is a duplicate. However, knowing this implies an enormous amount of knowledge about the structure of the real world and the external semantics of events. Moreover, *encoding* this knowledge in a way that will be suitable for triggering an adaptation seems likely to be inordinately complicated for any realistic case.

Although simple, these cases would defeat most event-algebra systems (for example the one described in [Hayton96]). We hypothesise – without any formal justification – that the twin problems of openness and noise render such algebraic systems intractable.

The conclusion we may draw is that, while event systems may be scalable from a systems perspective, they are decidedly *not* scalable from a programmer's perspective.

The problem is that events are being used to two disjoint purposes. On the one hand, events are used to indicate that "something happened" (albeit with some uncertainty); on the other hand, event traces are being used as the system's model of the outside world. The former is a system-level issue that is handled well by events; the latter is a semantic-level issue that is not. If we decouple the two, we may develop a hybrid system having the disadvantages of neither.

A more knowledge-driven approach

We may observe that many adaptive systems decisions are phrased in logical terms: "when A and B are in the room then…". We might therefore import techniques from knowledge-based systems to drive adaptations when particular conditions are true.

This approach has given rise to other contextual systems, for example [Wang04] using RDF to represent knowledge. Several programming techniques are then possible, including the use of truth-maintenance techniques to execute adaptation code when a predicate changes truth-state.

Such techniques face twin problems of uncertainty and noise. Most information derived from sensors is inherently error-prone, and sensors give rise to incorrect observations. To take one example, several authors have used RFID sensors to observe tags attached to people or artifacts. However, RFID sensors will fail to spot some tags, perhaps because it is moving too slowly to activate. They will also sometimes mis-identify tags because of interference. This means that a sensor-derived event may be incorrect or may be missed. It is easy to see why event traces are such an inadequate source of modeling.

However, it is possible to use a knowledge base as a stabiliser on the context model. Events must be treated as evidence for a fact, rather than as true Boolean values. We may then use techniques such as Bayesian probability, fuzzy logic or Dempster-Schaffer evidence theory to combine individual pieces of evidence into a more confidently-held view of the environment, which can then in turn be used to drive adaptation. This helps combat the danger of a system changing state dramatically as a result of a single, erroneous event, since other already-accepted evidence can act as a counterweight. Adding more knowledge of about the system (such as the behaviour of people in space) further increases this stabilisation effect.

3. Programming hybrids

This leads to a hybrid model in which an event infrastructure is used to collect and distribute evidence for the state of the system's environment, with the evidence being used to populate a knowledge base that maintains levels of confidence (or uncertainty) about that environment.

What sort of applications can be built on such a system? This is one topic of our current research. However, the nature of the available information provides some constraints.

The first observation is that all decisions are necessarily tentative. Uncertain reasoning approaches may allow a system to maintain an on-going level of confidence about its environment. Having a confidence interval makes such systems sensitive to small changes: a small change in evidence may cause the decision-making process to "tip". It remains the case, however, that many adaptation decisions are "crisp", so that the uncertain reasoning collapses to Boolean logic when the decision is made. This uncertainty means that we need to maintain one or more recovery strategies for any adaptation or decision the system makes, since each may need to be undone for at least two reasons: because circumstances change to cause a new adaptation, or because the additional evidence shows the initial adaptation to have been mistaken.

A second observation is that adaptations are not arbitrary: systems do not change from one behaviour to another, completely unrelated behaviour, but rather change within an envelope according to environmental changes. A core task for engineering autonomic systems is to ensure that all adaptations do indeed remain within the design envelope and do not take the system to unacceptable parts of the behavioural space.

Finally, while autonomic systems of this type can make use of significant bodies of existing AI research, the levels of noise and uncertainty, coupled with the degree of unsupervised operation required, do seem to pose genuinely novel challenges. We believe that there are several foundational innovations to be made in the logics and reasoning approaches used to describe autonomic adaptation, as well as in the way this reasoning is used to select adaptive behaviour. In particular, we are becoming convinced that approaches that account for the entire system behaviour at once may have advantages over those which try to coalesce a number of individual independent adaptations. In a sense this is the difference between set theory and topology: we believe that topological approaches may prove useful both the analyzing and programming adaptive systems.

4. Future work

We believe that a combination of event-handling and knowledge management – distributed systems combined with AI techniques – offers a useful hybrid approach to modeling the context of adaptive systems. The knowledge base provides an important gain in the expressive power of the system in the face of erroneous events. The partial and tentative nature of all such knowledge means that programming techniques must make extensive use of uncertain reasoning and

other AI-derived techniques. We further believe that it is important to move away from one-adaptation-at-a-time engineering to adopt a more holistic, closed-form approach to describing, analyzing and programming adaptive behaviours.

Our work in this area is following three complementary strands. From the systems perspective, we are developing a hybrid event and knowledge context system and evaluating different strategies for distributing and maintaining knowledge. From a programming perspective, we are exploring a range of programming models using combinations of events and knowledge. Another area of interest is whether we can use failure and noise constructively to drive computation[1]. Underpinning these activities is work on the semantics of adaptive systems: what exactly does it *mean* to be adaptive, and how can we capture the "shape" of that behaviour.

Acknowledgements

A number of members of SRG are actively and invaluably contributing to pursuing the ideas presented here, especially Graeme Stevenson, Sergey Tsvetkov, Lorcan Coyle, Steve Neely, Abdur Razzaque, Joe Kiniry and Paddy Nixon.

References

[Bonabeau99] Eric Bonabeau, Marco Dorigo and Guy Theraulaz. Swarm intelligence: from natural to artificial systems. Oxford University Press. 1999.

[Coutaz02] Joëlle Coutaz and Gaëtan Rey, Foundations for a theory of contextors. In Computer-aided design of user interfaces **3**, pp. 13–34. Christophe Kolski and Jean Vanderdonckt (eds). Kluwer. 2002.

[Dobson04] Simon Dobson and Paddy Nixon. More principled design of pervasive computing systems. Proceedings of Engineering for Human-Computer Interaction and Design, Specification and Verification of interactive Systems (EHCI-DSVIS'04). Springer-Verlag. 2004. To appear.

[Hayton96] Richard Hayton, Jean Bacon, John Bates and Ken Moody. Using events to build large-scale distributed applications. Proceedings of the 7th ACM SIGOPS European workshop: Systems support for worldwide applications, pp. 9–16. ACM Press. 1996.

[Salber99] Daniel Salber, Anind Dey and Gregory Abowd. The Context Toolkit: aiding the development of context-enabled applications. Proceedings of CHI'99, pp. 434–441. 1999.

[Wang04] Xiaohang Wang, Jin Song Dong, Chung Yau Chin, Sanka Ravipriya Hettiarachchi, Daqing Zhang. Semantic Space: an infrastructure for smart spaces. IEEE Pervasive Computing **3**(3), July–September 2004. pp. 32–39.

[1] This work is due to Jake Beal of MIT.