Whole-system programming of adaptive ambient intelligence

Simon Dobson and Paddy Nixon

Systems Research Group, School of Computer Science and Informatics
UCD Dublin IE
{simon.dobson, paddy.nixon}@ucd.ie

Abstract. Ambient intelligence involves synthesising data from a range of sources in order to exhibit meaningful adaptive behaviour without explicit user direction, driven by inputs from largely independent devices and data sources. This immediately raises questions of how such behaviours are to be specified and programmed, in the face of uncertainty both in the data being sensed and the tasks being supported. We explore the issues that impact the stability and flexibility of systems, and use these issues to derive constraints and targets for the next generation of programming frameworks.

1 Introduction

Ambient intelligence faces an uncomfortable duality. On the one hand, systems must be constructed from networks of independently-constructed and -programmed devices, often having very restricted local functionality; on the other, these systems must produce user-centric behaviour which reacts to complex scenarios in a way that retains a clear focus on the users' activities on an ongoing basis. While it is attractive to think that user-centric behaviour will simply "emerge" from the interactions of individual devices, there are strong reasons to suspect that a more directed approach will be needed in most cases. This suggests that ambient intelligence requires a "script" of some kind for each service or application being hosted, provided at the level of the network rather than the level of the individual devices.

While numerous pervasive *applications* have been demonstrated, large-scale *systems* remain elusive. It is hard to recognise what is happening in the real world; hard to translate this into meaningful adaptive behaviour; hard to ensure that the adaptive behaviours remain within an acceptable envelope; hard to ensure that individual services compose without interference; and hard to deal with the inevitable mistakes that occur when performing complex actions without explicit user instruction.

We have previously conjectured [1] that whole-system descriptions of adaptive systems provide both a more principled approach to design and a more verifiable platform upon which to perform development. Experience has borne-out this conjecture, but has also raised the significance that the uncertainty of

sensor data and task inference have for ambient intelligence applications. Our goal in this paper is to explore two questions which we believe are of critical importance for any platform for adaptive systems:

1. What impact do the low quality, diffuse and semantically heterogeneous nature of sensor data and task inference have on programming? and
2. What do these imply for the creation of middleware and other programming support frameworks for ambient intelligence?

We conclude that ambient intelligence is *not* primarily about sensing, but rather relies on leveraging information from a diverse range of sources with rich interconnections. This in turn changes the fundamental problem from being one of sensing and reasoning to one of information management and truth maintenance: a holistic view of the system's responses to its changing environment.

Section 2 explores the nature of sensing and inference in ambient systems, from which section 3 derives some criteria that must be met by any suitable programming platform. Section 4 concludes with some possible directions for future research.

2 Uncertainty in sensing and inference

The literature on pervasive computing and ambient intelligence contains a large number of example applications. Typically these follow a similar storyline (figure 1): a collection of sensors is fitted to an environment and used to collect data about the activities of users within that space, who may be using various electronic and networked devices. Events from these devices are used to select behaviour for the system to exhibit, which in turn is visible to and affects the environment. This closing of the feedback loop is characteristic of adaptive and autonomic systems [2].

Unfortunately this abstract description elides all the issues which make ambient intelligence problematic – and also those which make it most useful.

2.1 The uncertainty of sensors

Traditional applications provide a bounded context within which interaction may occur. For most systems there is very little uncertainty in these interactions: a user presses a key or selects a menu item, or doesn't. In some systems there may be errors due to lost, garbled or repeated messages which are typically handled using middleware; in almost all systems there will be operator errors which must be correctable in some way, typically with minimal impact[1]. The point is that interaction, while it may be *wrong*, is at least *definite*.

[1] Although not always: accountancy software, for example, will typically not allow an erroneous transaction to be deleted, forcing the user to explicitly record a cancelling transaction. This of course if the "correct" behaviour for this particular application domain

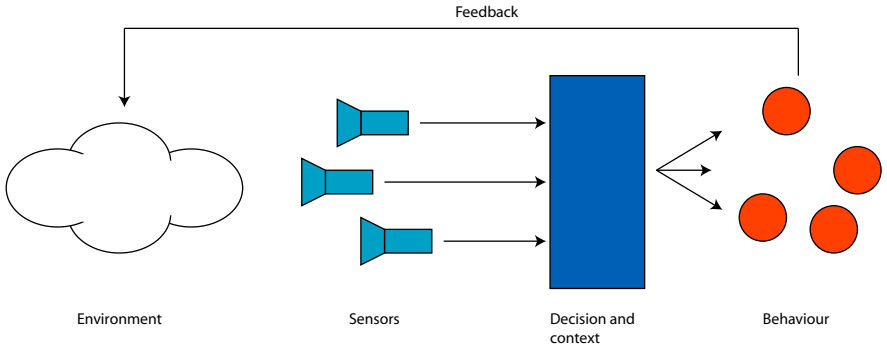

Fig. 1. Archetypal ambient intelligence applications match sensor data to a task description

Ambient interaction exhibits completely different characteristics. Interaction occurs (at least in part) by way of sensors observing "everyday" actions. Even within a single category such as location, each family of sensor has often radically different characteristics (figure 2).

System	Range	Precision
GPS	Global	1m
GSM triangulation	Global	50m
RFID	Building	room/door
Wifi triangulation	Local-area	20-50m
Ultra-wideband	Room	10-30cm
Cameras	Room	10cm

Fig. 2. The precision of location sensing can vary dramatically

First we must recognise the distinction between *precision* and *accuracy* – two terms that are sometimes used almost synonymously. A location sensor's precision describes the smallest detail it can resolve – or, to put it another way, the minimum distance that an object must move before the sensor registers a change. A sensor's accuracy describes the extent to which the location it reports for an object corresponds to that object's "real" position. For example, a Wifi location system such as PlaceLab might generally report a device's position to within 10m (precision), but might also (because of reflections or beacon misplacement) exhibit sporadic or systematic errors [3] which place a device in a totally different position unrelated to the way it is actually sensing at the physical level (accuracy). The terms apply analogously to most sensors.

This raises an interesting mis-match. The design of an application in the style of figure 1 will typically be phrased in terms such as "when A and B are

in room *X*, perform action *1*". No single sensor is likely to provide information of sufficient precision to make this decision with any confidence.

If these issues sound exotic, they can appear in some very simple scenarios. Suppose we build a system that detects meetings being held in an office, using an ultra-wideband location system such as Ubisense[2] (figure 3). When all the participants enter the office (area A), the meeting behaviour – whatever it is – is executed. If however one of the participants stands near the door for whatever reason (for example to draw on a badly-placed whiteboard near area C), the location system may report the person's location as being outside the office: the precision of the location report straddles two spaces. The system may then erroneously end the meeting. If the next location reported is inside the room again, the system may start another meeting (because all the participants are together again), or may do nothing (since the meeting is ended) – or do something else entirely.

There are a number of issues to be teased out here. Firstly, there is an interesting interaction between the sensors' imprecisions, the layout of the real world, and the behaviour a user will observe. For the system to potentially exhibit undesirable behaviour, a user must stand in an area in which, at the location sensor's resolution, straddles two spaces, *and* those spaces must exhibit different behaviours, *and* the user must stand in the area without meaning proceed into the other space. If any of these conditions does not pertain, the system will be more likely to behave correctly: if, for example the whiteboard is moved away from the door to the other side of area A, a user standing in the zone of uncertainty is less likely to be an unwanted transient.

The underlying problem is that *we cannot assign meaning to a single action*, only to a *constellation of actions* taken in context. Some actions "just happen", and so we need to be careful that we do not designate innocuous actions as triggers for behavioural change.

2.2 From sensing to task

Event-driven pervasive systems address this issue by providing a model of the tasks which the application is supporting. The events received are used to "move the task along" in the manner of a workflow system. Unfortunately processes in pervasive systems tend not to be well-specified – and if they are, users will anyway tend not to follow them with the precision required of workflow. In scenarios in which the process *can* be followed precisely (such as for the computer theatre of Pinhanez and Bobick [4]) then temporally-enriched forms of workflow such as [5] can usefully be used: in less structured cases, however, experience has shown that such systems do not provide significant benefits and have problems with their expressive power.

Things become even more complicated when we consider ambient *systems* containing several applications or services. A given action may then be interpreted by several applications simultaneously, each of which may then exhibit

[2] http://www.ubisense.net

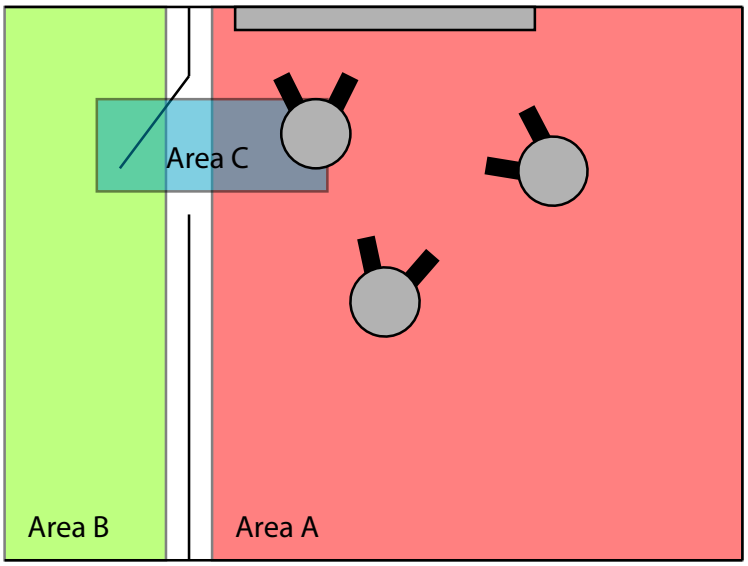

Fig. 3. There are some subtle interactions at play between sensor imprecision, behaviour and task

a behaviour. These behaviours are "independent", in the sense of not affecting one another; however, they may also be "dependent" in that one application's behaviour may detrimentally affect another's. The classic example is having two applications controlling the lighting level in a room: if they disagree on the "right" level, which is selected?

3 Designing and programming with uncertainty

To summarise, any data used to drive a pervasive computing system is inherently uncertain in ways that improved sensor technology will not completely address. The relationship of this data to the scenarios being supported by the pervasive systems is tentative, since an action often cannot be unambiguously assigned a particular meaning in a process that is subject to the uncertainties of users' everyday actions. This implies that *any decision taken within a task may be wrong*, in that it may be driven by data that has been mis-interpreted.

What does this imply for systems? One might take the pessimistic view that ambient intelligence is a flawed dream which cannot be achieved because of the limitations of sensing the real world. A more balanced view is that *any decision may be taken incorrectly*. This has a significant impact on programming. We can split the problem into two parts: how can we determine that a decision has been taken in error?, and what can be done if and when the mistake is detected?

Although sensors are individually imprecise, a sufficiently sensor-rich environment may be able to compensate for these individual deficiencies and develop a more accurate consensus estimate of the values being sensed. This raises a question about when an environment is sufficiently sensor-rich – a question to which we have no principled answer.

We may observe, however, that an ambient system can be made significantly richer without heavy investment in sensors. Using location as an example, a system may have many potential sources of information as to an individual's location: sensors form one part, but there are also diary entries, actions taken that correlate to locations, default behaviours that may be assumed, and so forth [6]. Such alternative sources are of low precision – people's diaries do not generally reflect their movements precisely – but this is simply a feature they share with any other sensor. We may therefore extract low-precision location information from a diary (for example) and combine it with the sensor readings we are obtaining from the environment. It is simply another aspect of context that can be "mined" for location information.

The important point is that we decouple behaviour from observations taken in isolation, and instead use a holistic model of the environment which is driven by *all* aspects of the environment that we can sense. This leads is to a somewhat different abstract model than we used in figure 1: use sensed data to maintain and refine a model of the world that is the "best fit" to the observed data. Applications and services, in this model, test scenario hypotheses against the world and activate those that are supported by the current consensus (figure 4). Scenarios are deactivated when they cease to be supported.

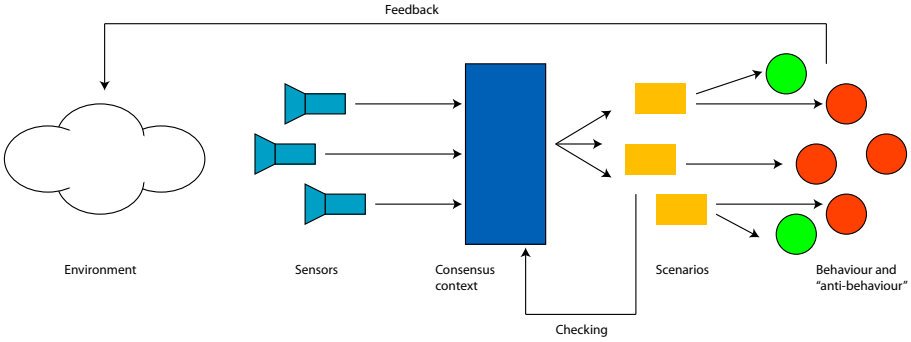

Fig. 4. We need to constantly check the validity of hypotheses, including past ones, against the evolving context

This leads to a number of general statements about programming platforms for this class of systems:

Events are too noisy to serve directly as a basis for programming. An event derived from a sensor, such as an observation of an individual in a

particular space, may be issued in error. If the application reacts to this event it will react to noise, so applications must attempt to identify noise locally before performing actions. In many cases the signal-to-noise ratio will be low, and applications can become cluttered with noise-handling code at the expense of application logic. This is hard to do purely at the event level, without reference to a wider context, and suggests that "composite" event systems such as those of Hayton *et alia* [7] may not be suitable for pervasive applications.

It is important to stress that this is not a criticism of events as a *communications* construct, where they can provide a very flexible and scalable communications framework. Rather, it is a criticism of using events directly as a *programming* construct, since they seem to devolve too much of the system's complexity onto applications programmers under an abstraction model that makes it difficult to deal with locally.

Don't take anyone's word for anything. One solution to noise is to construct systems which "damp" erroneous observations by fusing observations across a range of sources. By maintaining a context model, distributed across the pervasive system, which maintains a running "consensus estimate" of values of interest, a single observation has less scope for skewing the consensus.

A corollary of this approach is that it provides a way of dealing with semantically heterogeneous data sources. Each sensor can deliver data in its lower ontology, which can then be mapped to an upper used by applications [8]. The advantage is that applications are insulated from the details of the individual sensors available, reducing the impact of changes in the sensor population. The upper ontology can present the on-going consensus estimate of values such as location, allowing context fusion to occur within the platform and away from individual applications. This is the approach we are taking with the Construct platform [9, 10].

Interconnection is more important than data. Although it is a commonplace to think of ambient intelligence as pertaining to sensors, it might be more useful to think of it in terms of leveraging information from a richly-interconnected model. It is possible to build ambient intelligence with little or no traditional sensor data for example: a location-aware system could be driven by asking the user to spot landmarks, or by using default assumptions about behaviour.

By contrast, we cannot deem a system "ambiently intelligent" if it is simply driven by sensor events, taking no account of other factors. The model, its relationships and the ability to extract information through fusion is far more characteristic of the domain than the availability of sensors.

Any decision needs a mitigation strategy. No matter how successfully we perform consensus, however, it is inevitable that errors will remain, and these errors can cause major problems for applications. At a suitably abstract level, the issue is simple: a decision is made at time t, and at some subsequent time $t + \delta$ information becomes available to show that the decision was made incorrectly. The problem is then to incorporate this new information into the on-going computation. This may happen as a result of sensor error, of mis-interpreting an

action intended for another service, or simply because new information arrives over time.

The classical approach to such problems is rollback as found in databases. This will no work for pervasive computing: one might literally be shutting the stable door after the horse has bolted! We must therefore investigate non-classical solutions such as, for every action a resulting from a decision, providing an action a^{-1} that can be performed if action a is determined to have been in error. Actually the situation is a bit more subtle, since the appropriate action a^{-1} is actually a function of δ, and will vary depending on when the error is recognised.

The general point, however, is that applications must be written with failure in mind. In programming language terms, mistakes are often dealt with using exceptions; in ambient intelligence, *failure is not exceptional* and so requires a different programming idiom. It is by no means clear how to provide such an idiom without cluttering a system with mitigating code.

Everything interesting comes from composition. The final criterion involves the nature of systems. While it is possible to address many of the issues we have highlighted within a single application, such *ad hoc* approaches will inevitably break down when confronted with more complex systems, and especially with systems composed of applications which are not known *a priori*. It is therefore essential that ambient intelligence is constructed on a well-founded basis which is compositional in nature.

At the present state of the art, placing two ambient services in the same space is a somewhat fraught activity. We have no way of deciding whether two systems will interact positively or interfere. Similarly, independent behaviours may be suitable to run together or may not, or one may take precedence over another. The details of a particular system are not important for the current argument: what *is* important is the idea that composing applications and services has a system-wide impact, and the results of composition should be well-definable ahead of time. This is the only way in which we can reliably deploy large-scale ambient intelligence with the degree of confidence needed by a large organisation.

4 Conclusion

Ambient intelligence involves synthesising data from a range of sources in order to exhibit meaningful adaptive behaviour without explicit user direction. The imprecisions, inaccuracies, incompleteness and contradictions implicit in sensor- and inference-driven decision-making place strong requirements on the stability and flexibility of systems, which in turn impact on the programming solutions deployed.

In this paper we have attempted to highlight some of these issues in a way that lends itself to the development of programming platforms suitable for ambient intelligence. The challenges centre around dealing with through-going uncertainty in a principled fashion, whilst providing an environment in which services may be safely composed with known impact. This involves developing tools and techniques – at design, programming and analytic levels – which can improve

the confidence with which we make decisions based on often flimsy data. Exactly what sort of programming environments will emerge to address these challenges is an open – and exciting – area of current and future research across the community.

In some ways the current activity mirrors that of parallel and distributed programming. Initial approaches based on semaphores, suitable for simple applications but with rapidly intractable complexity as systems grow, were replaced by more structured approaches such as transactions and skeletons together with middleware abstractions reflecting the realities of the underlying domain. We hope that ambient intelligence will evolve similar mechanisms for manging the complexity of systems.

Acknowledgements

This work is partially supported by Science Foundation Ireland under grant numbers 05/RFP/CMS0062 ("Towards a Semantics of Pervasive Computing"), 04/RPI/1544 ("Secure and Predictable Pervasive Computing"), and 03/CE2/I303-1 ("LERO: the Irish Software Engineering Research Centre").

References

1. Dobson, S., Nixon, P.: More principled design of pervasive computing systems. In Bastide, R., Roth, J., eds.: Human computer interaction and interactive systems. Volume 3425 of LNCS., Springer Verlag (2004)
2. Kephart, J., Chess, D.: The vision of autonomic computing. IEEE Computer **36** (2003) 41–52
3. Hightower, J., LaMarca, A., Smith, I.: Practical lessons from PlaceLab. IEEE Pervasive Computing **5** (2006) 12–19
4. Pinhanez, C., Bobick, A.: Human action detection using PNF propagation of temporal constraints. In: Proceedings of CVPR'98. (1998) 898–904
5. Allen, J., Ferguson, G.: Actions and events in interval temporal logic. Journal of Logic and Computation **4** (1994) 531–579
6. Dobson, S.: Leveraging the subtleties of location. In Bailly, G., Crowley, J., Privat, G., eds.: Proceedings of Smart Objects and Ambient Intelligence. (2005) 175–179
7. Hayton, R., Bacon, J., Bates, J., Moody, K.: Using events to build large-scale distributed applications. In: Proceedings of the 7th ACM SIGOPS European workshop on systems support for worldwide applications, ACM Press (1996) 9–16
8. Clear, A.K., Knox, S., Ye, J., Coyle, L., Dobson, S., Nixon, P.: Integrating multiple contexts and ontologies in a pervasive computing framework. In: Contexts and ontologies: theory, practice and applications. (2006)
9. Coyle, L., Neely, S., Rey, G., Stevenson, G., Sullivan, M., Dobson, S., Nixon, P.: Sensor fusion-based middleware for assisted living. In Nugent, C., Augusto, J.C., eds.: Smart homes and beyond, IOS Press (2006) 281–288
10. Dobson, S., Coyle, L., Nixon, P.: Hybridising events and knowledge as a basis for building autonomic systems. Journal of Autonomic and Trusted Computing (2007) To appear.