Soft-interfaces for Ubicomp Appliances

ALLAN WONG, MEURIG BEYNON

Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK

Email: {allan, wmb}@dcs.warwick.ac.uk

Abstract: In appliance design, one of the most difficult issues is capturing the requirements of the intended users, especially as, in a ubiquitous computing (ubicomp) environment, uses are not always foreseeable. Traditional methods for information appliance design essentially involve circumscribing the roles of the users, often through the explicit consideration of use cases. This is a result of the desire to create fully-automated systems. It is also reflected in the interfaces provided by most appliances today, which are usually rigid and inflexible. This paper argues that full automation is not desirable in a ubicomp environment, and that we need more flexible interfaces that can be adapted through human interaction according to use situations as they arise dynamically. We propose a soft-interface approach for controlling ubicomp devices based on Empirical Modelling principles developed at Warwick University. A visual modelling tool has been developed to demonstrate the idea of controlling devices by constructing simple definition-based models.

Introduction

Ever since the 1960s, researchers have been finding ways of using computers to support everyday activities. One ambitious early venture was the Honeywell Kitchen Computer in 1969. An advertisement reproduced in [Spi00] shows a housewife standing beside a computer the size of a kitchen table with a display panel and teletype interface. The main virtue of the computer was its capacity for storing a huge number of recipes. The user was intended to input all available ingredients into the computer, which would then tell the user what could be made with them. The scale of

the user intervention required for this task was reflected by the fact that the purchase price included attendance at a two-week programming course. As mentioned in [Spi00], no Kitchen Computer was ever sold.

The failure of the Kitchen Computer is the kind of experience that has led designers to consider minimising human intervention. The common vision nowadays is of ubiquitous computing (ubicomp) systems that require virtually no human intervention. Negroponte advocates the use of "intelligent agents" as digital butlers that do all the work for you while you take it easy [Neg96]. Joseph describes this is as "the top of the IT agenda" [Jos02]. Tennenhouse, a vice president in the Intel Corp, advocates "getting the human out of the interactive loop" [Ten00]. This vision is arguably only an industrial hype. Full automation is not plausible for ubicomp. The main reason is that the ubicomp environment is the environment we live in – where activities are situated and exceptions are the norm. Since we cannot prescribe the ubicomp environment, human intelligence has to be involved in solving ubicomp problems. Even the authors who advocate full automation seem to recognise a role for some form of intelligence. Joseph [Jos02] envisages that "computers will be intelligent enough to manage, configure, tune, repair, and adjust themselves to varying circumstances to handle the workload exposed to them efficiently". Tennenhouse [Ten00] wants to automate the software creation process in ubicomp by generating software from specifications and constraints. Such visions presume that we can automate management and specification tasks that seem to require human intelligence.

Undeniably, many people dream of sitting back and relaxing and allowing machine servants to help them to do all their work. This dream has become one of the driving motivations of ubicomp development. However, we cannot desire automation blindly for every device and aspect of ubicomp. Automation introduces problems of

predictability and accountability. Edwards [Edw01] asks: "how will the occupant-users adapt to the idea that their home has suddenly reached a level of complexity at which it becomes unpredictable?" To paraphrase Langheinrich *et al.* [Lan02], "in order to lower the demands on human intervention in … a dynamic world … [we require] the concept of *delegation of control*, where we put automated processes in control of otherwise boring routines, yet provide *accountability mechanisms* that allow us to understand complicated control flows".

This paper explores principles for system implementation that provoke a reappraisal of the appropriate balance between automatic and manual aspects of a ubicomp system. We are led to propose that, in the ubicomp environment, creating and supervising the automatic processes should be part of the users' role. We also argue for systems that keep the human in the loop and aim to enhance and complement rather than displace human interaction.

Adaptation in ubicomp systems

In the ubicomp environment, requirements are unsettled. Users' needs change over time, so that a ubicomp system should facilitate dynamic adaptation to various situations. Research on ubicomp usually associates adaptation with context-awareness of applications (e.g. [Abo00, Dey01, Lae01]). In [Abo00], Abowd *et al.* point out that "ubicomp applications need to be context-aware, adapting their behaviour based on information sensed from the physical and computational environment". The definitions of the term 'context' given in the literature vary but a generic definition can be found in [Dey01]: "Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." Because of the aspiration to develop fully automated

systems discussed earlier, the typical research focus is on capturing context by using sensor technologies. Capturing context through sensors only works when we can characterise the set of possible contexts needed in advance. However, in a ubicomp environment what is relevant to the interaction between a user and an application cannot be fully specified, and usually relevance changes over time. Sometimes, what is relevant is very subjective to the user. Inferences made through the preset sensors may not be accurate. As Edwards *et al.* [Edw01] observe "... [simple sensing] may report that I am present in a room when, instead, I have simply left my active badge on the desk".

A complementary way of trying to fulfil the adaptation requirement of ubicomp is through 'user modelling'. Traditionally, user modelling is about constructing an explicit profile of properties and preferences of the user in the system. The profile is used as the basis for adaptation and personalisation of the system. There are two approaches to user modelling: *adaptive* and *adaptable* [Fis00, Kul00]. In the adaptive approach, the system dynamically adapts itself to the current task and the current user. In the adaptable approach, the system gives substantial support to allow the user to change the functionality of the system.

Systems like GUIDE (a tourist guide system [Che01]) and PDS (a personal daily system [Byu01]) use both context-aware and user modelling approaches. Combining user modelling with context-awareness in an application improves the system's adaptation capability to some extent. However, there is no way for a system to take full account of all the preferences of the user by just maintaining an explicit representation of the properties of the user. In fact, even the user might not know his or her preferences in respect of a system. Consider one of the scenarios for a PDS described in [Byu01]: "When a user passes by a theatre, the PDS can notify the user that the theatre is playing the user's favourite movie." In this case, the location

together with a film preference of the user triggers the notification. The question is: who is to specify this behaviour of the system? If it is to be the system, we have the issue of properly predicting the user's state of mind; if it is to be the user, we have the issue of adequately supporting the user's need to specify the behaviour. Whichever is the answer, however, it is clear that context-awareness and user model maintenance alone are not sufficient for system adaptation. Even in this simple situation, we need mechanisms to *customise* the system to suit individual needs. The system should have some means to represent the user's preference and have some reflective capability concerning the way in which these preferences are represented within itself. The user should likewise be able to understand his or her preferences and relevant functionality of the system. Evolvability of the system depends upon establishing a close link between the system and a user's understanding, which in turn comes from openness to interaction and customisation.

An Empirical Modelling conceptual framework

Empirical Modelling (EM) is an approach to computer-based modelling that has been under development at the University of Warwick for many years. This section sketches the context for its proposed application to ubicomp to be described below (for more background and details of EM, see [EMWeb] and the references cited later in the section). EM is based on two key premises:

1. that human understanding of the world is intimately linked to our perception of the possible *agents* that are at work, the nature of the *observables* that mediate their interaction, and the *dependencies* that link the atomic changes to these observables in such interaction.

2. that such understanding is primarily acquired and developed through the construction of interactive artefacts to embody perceived patterns of agency, observation and dependency.

Premise 1 identifies the basic concepts that underlie the EM agenda for domain understanding. There is a fundamental ontological distinction between agency, observation and dependency in EM and the abstract notions that these terms designate in other computing literature. In particular, EM does not merely promote a new form of abstract analysis, alternative to a standard object-oriented, functional or agent-oriented analysis. In accordance with premise 2, EM emphasises the essentially practical, experimental and constructive nature of the development of domain understanding. The exploration of understanding proceeds in conjunction with concrete model-building (typically computer-based), primarily with reference to experimental interactions in a specific situation, rather than to an abstract behaviour based on preconceived patterns of interaction. In this respect, EM models resemble the models that are created by an experimental scientist in the course of investigating a phenomenon. As explained by Gooding in his analysis of the role of experiment in the making of meaning [Goo90], such models serve as *construals*, and embody understanding of a phenomenon in a way that we regard as consonant with premises 1 and 2.

In the practical application of EM, the interactive artefact referred to in premise 2 typically takes the form of a computer-based model based on a *script of definitions* that captures acyclic dependencies between observables (cf. Figure 3 below). The values and dependencies in such scripts are conceptually open to redefinition by the modeller at any time, subject only to the constraint that the modeller can imagine an interpretation for the state change this provokes. Such interaction helps the modeller to identify the agents, observables and dependencies referred to in premise 1. Such

agents can be represented by groups of definitions and their possible actions and behaviours recorded by patterns of redefinition (see [EMWeb] for a variety of example models).

Much previous work has shown how EM models of this nature can capture commonsense understanding that has particular relevance for ubicomp applications. EM can be applied, for instance, in developing Interactive Device Models (IDMs) of the appliances that constitute a system (cf. [Bey01, Run02]), in developing Interactive Situation Models (ISMs) to represent the situations in which a system is to operate (cf. [Sun99, Bey99, Bey00, Bey01]), and in modelling the interactions between the designers and users that shape the development of the system itself (cf. [Adz94]).

By way of illustration, in developing a ubicomp central heating system: a solar panel appliance might be represented by an IDM in which the observables included power output, angle of panel orientation, and current time; agents would include the automated mechanism for orienting the solar panels and the human users who might exceptionally wish to set the orientation for security or maintenance purposes; and a dependency might relate the default orientation of each panel to the current time. An ISM that embraced this appliance in its operational context would include: observables such as the current abstract location and level of intensity of the sun; agents such as the sun, the wind and birds that might perch upon the appliance; and dependencies such as determine the current stresses on panels as determined by wind forces. Finally, in designing the appliance: the relevant agents would include the different categories of designer, whose interests might embrace both thermal and structural engineering aspects, aesthetic, material and cost concerns; the relevant observables would include all the observables associated with the current proposals from each such designer (as represented by their personal ISMs for the appliance as a design in progress) together with observables to represent the current status of the

concurrent design process (such as deadlines for completion, budget contraints, expected work patterns for cooperation between designers etc); dependencies might reflect the manner in which the interactions between designers were constrained at each stage in the design process (e.g. the total area of panels might depend upon the choice of material and the budget).

The EM conceptual framework for ubicomp (to be referred to as "soft interfaces for the control of devices (SICOD)"), can be interpreted as controlling devices through building special-purpose EM models called Interactive Control Models (ICMs). The EM conceptual framework aims to:

- make the infrastructure of ubicomp more *visible* to the users
- provide principles to help users to maintain the conceptual integrity of their views of ubicomp systems.

An ICM typically consists of a set of IDMs and an ISM. The left of Figure 1 shows an ICM with three IDMs (depicted by pentagons) linked by dependencies to an ISM (depicted by a circle). The whole ubicomp environment comprises a network of ICMs (see the right of Figure 1). Notice that an IDM can be shared by more than one ISM. This reflects the fact that many devices are shared resources in a ubicomp environment.

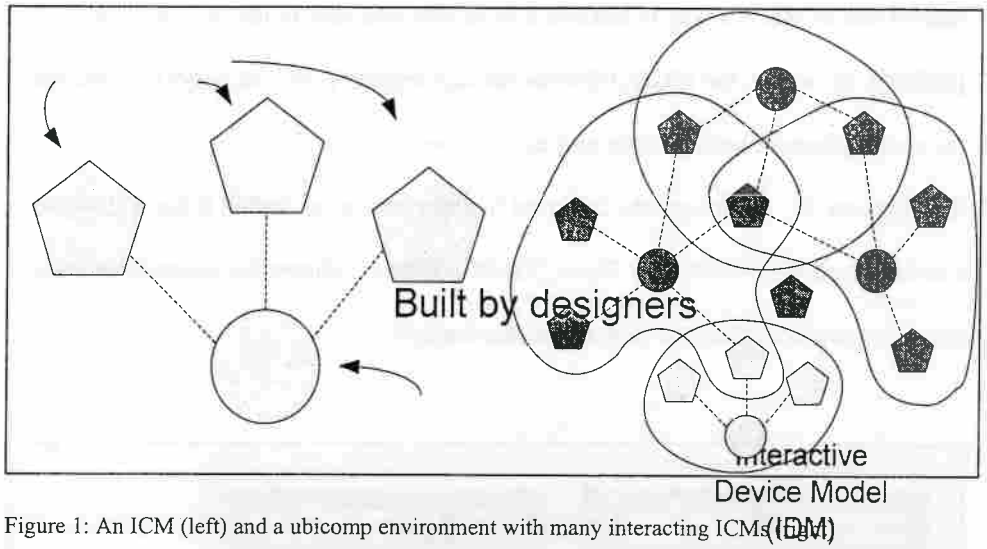

Figure 1: An ICM (left) and a ubicomp environment with many interacting ICMs (right)

Interactive
Device Model
(IDM)

Both IDMs and ISMs are EM models, but the differences between them are summarised as follows:

Interactive
Device Model
(IDM)

- An IDM is built by designers; an ISM is built and maintained by users.
- An IDM corresponds to a particular device for generic application; an ISM corresponds to a particular situation for individual use.
- An IDM assists users to gain a conceptual understanding of the device; an ISM assists users to configure devices through creating definitions to establish dependencies between the IDMs that reflect their observation of the use situation.

Interactive
Situation
Model
(ISM)

Maintained
by a user

A soft-interface approach

We argued above that conventional context-awareness and user modelling techniques are not sufficient to meet the system adaptation requirement for a ubicomp environment. We also need ways for the user to customise the system. Ideally, these should be flexible and easy to use. Usually there are trade-offs between flexibility and

ease of use so that if a way is flexible it is usually not easy to use and *vice versa* (cf. [Odl99]). However, the SICOD framework can arguably provide ways to customise the system that are both flexible and easy to use.

To illustrate this point consider a central heating controller, called Balmoral, based on a real-life model described by Green [Gre99]. Figure 2 shows the control panel and a summary of instructions on how to use the controls.

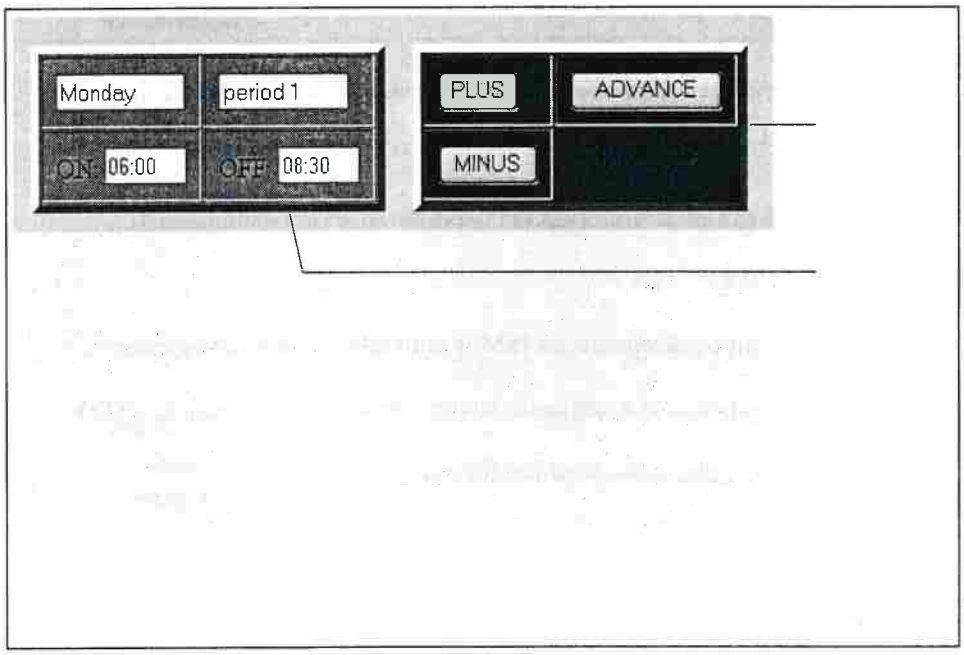

Figure 2: A layout of a central heating control panel and its instructions (adapted from [Gre99])

By pressing buttons on this control panel, a user can set up three periods of heating for each weekday. The operation of the control panel is highly dependent on the mode switching button called 'ADVANCE'. Mode switching buttons like this also exist in most programmable VCRs, washing machines, digital watches and desk clocks. They are also popular in conventional windows-based GUIs. Mode switching makes a system difficult to comprehend because a certain user action has the perception of

Summary Instructions

These controls determine three periods of heating
clicking the ADVANCE button you can move fro

the system accordingly. With a conventional 'press-button' interface, however, it is usually unavoidable because of the physical constraints on the number of buttons that we can put on a control panel. For example, it would be inconvenient to have a separate pair of 'PLUS' and 'MINUS' buttons for every weekday and every heating period. We refer to this kind of interface as a 'hard-interface'.

The design of a hard-interface not only prescribes the functionality of the system but also the possible observables and their interpretation by the user in the situations of use. The inflexibility of this interaction and interpretation is a potential barrier to providing a system view that has conceptual integrity for the user (e.g. consider the different roles that the 'PLUS' and 'MINUS' play according to the current mode of operation). It also affects the flexibility of the resulting system. For example, the central heating interface only allows the user to enter three heating periods for every weekday – a rather arbitrary prescription imposed by the designers.

The SICOD framework provides a different way to configure the system. We can regard an ICM as a 'soft-interface' to a system of ubicomp devices. Within the SICOD framework, an ICM for the central heating control would be as depicted in Figure 3. A visual modelling software prototype that enables the modeller to manipulate the definitions on the right of Figure 3 via the graph on the left has been developed to support the interactive construction of an ICM [Won03].

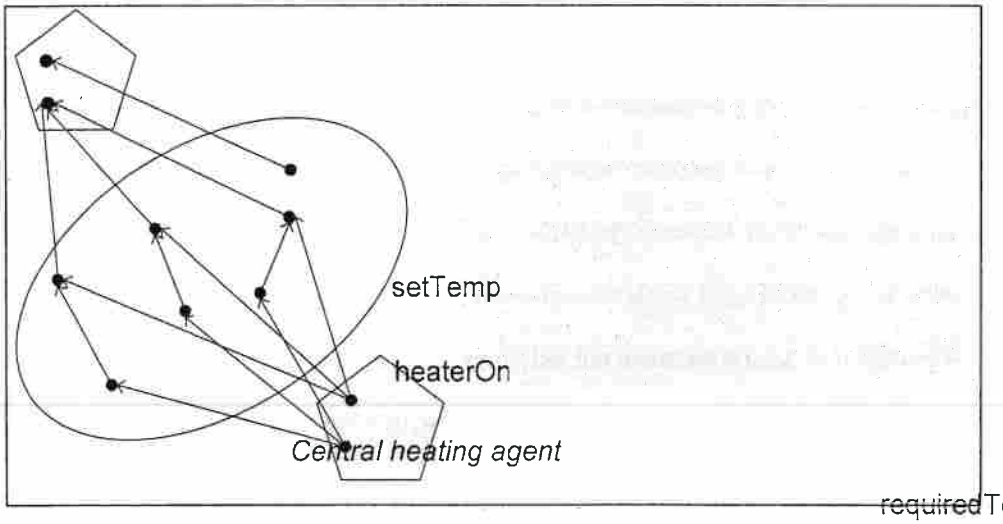

Figure 3: An ICM for central heating control (left) and its underlying definitions (right)

The observables `sunday`, `saturday` and `weekday` specify the user's configuration of the heating period. The observables `sundayOn`, `saturdayOn` and `weekdayOn` evaluated to `True` only if they can match their own day and heating period with the current time and day information provided by the `Clock` agent. The central heating will be turned on only if `heaterOn` evaluates to `True`.

With the ICM, extension of the system becomes very easy. For example, suppose that the user has bought sensors to detect if there is any person in the house. The central heating ICM can make use of this information to allow more efficient use of central heating – the user can change the `heaterOn` definition to:

```
heaterOn is (sundayOn or saturdayOn or weekdayOn) and houseNotEmpty
```

A conventional central heating interface will surely have difficulty in embracing this requirement without replacing the whole control panel and developing mechanisms to link to the sensors.

The SICOD framework improves both the flexibility and the ease of use of the resulting system:

- the system becomes more flexible as its ICM is open to change and extension;
- the system becomes easier to use as the SICOD framework provides a simple interface through which to customise devices. The user only needs to learn the underlying script syntax to be able to control a variety of different devices.

The idea of allowing the user to customise most parts of the ubicomp system is an alternative approach to the adaptation through auto-learning described in [Byu01]. Byun *et al.* propose several scenarios in which a ubicomp system can learn patterns of use. One of them is "If a user participates in a meeting at 10 am every fourth Monday, the PDS [their proposed system] might learn this behaviour and suggest that the user might have to go to a meeting today (the fourth Monday) at 10 am. However, a more sophisticated level of learning would enable the system to realize when such a notification is inappropriate, for example when the user is on holiday." [Scenario D in Byu01]. Simple problems of user customisation become complex problems of artificial intelligence. Even if we do eventually have systems which are smart enough to act on behalf of users, system predictability will become an issue.

In discussing possible extensions to his Context Toolkit, Dey [Dey01] describes the struggle involved in negotiating the tradeoff between supporting a complex situation and providing a simple method for describing a situation. He adds that "while designers who have domain-specific expertise can determine part of the solution [to a ubicomp problem], they will obviously not think of everything that is needed to support individual users. It is the end user who is in the best position to further specialize a context-aware application to meet their individual needs." [Dey01]. The application of the SICOD framework is potentially a simple way to allow users to represent situations by building ISMs.

Conclusion

The SICOD framework transforms the role of the designer from 'prescribing use situations' to 'developing reusable functionalities that allow users to specify use situations by themselves'. An analogy can be made here with the spreadsheet framework, in which designers provide a library of domain-specific functions but the actual use of these functions is for spreadsheet users with their specific tasks to determine.

With the SICOD framework, users can develop their understanding of use situations specific to them. This understanding can be animated and visualised by building ICMs. The result is a ubicomp system of devices without a fixed system boundary, capable of better adaptation to use situations and open to evolution.

Acknowledgements

Many thanks are due to our colleagues in the Empirical Modelling research group for many stimulating discussions in seminars and meetings.

References

[Abo00] Gregory D. Abowd, Elizabeth D. Mynatt, Charting Past, Present and Future Research in Ubiquitous Computing, *ACM Transactions on Computer-Human Interaction*, 7(1):29-58, March 2000.

[Adz94] V.D.Adzhiev, W.M.Beynon, A.J.Cartwright and Y.P.Yung, A Computational Model for Multiagent Interaction in Concurrent Engineering, In *Proceedings of CEEDA'94*, Bournemouth Univ., 227-232, 1994.

[Bey00] W. M. Beynon, S. Rasmequan, S. Russ, The Use of Interactive Situation Models for the Development of Business Solutions, *Workshop on Perspective in Business Informatics Research*, March 31-April 1, 2000.

[Bey01] W. M. Beynon, C. Roe, A. Ward, A. Wong, Interactive Situation Models for cognitive aspects of user-artefact interaction, In *Proceedings of Cognitive Technology 2001: Instruments of Mind*, 2001.

[Bey99] W. M. Beynon, P-H. Sun, Computer-mediated communication: a Distributed Empirical Modelling perspective, In *Proceedings of CT'99*, San Francisco, August 1999.

[Byu01] Hee Eon Byun, Keith Cheverst, Exploiting User Models and Context-Awareness to Support Personal Daily Activities, *Workshop on User Modelling for Context-aware Applications*, User Modelling 2001 Conference, 2001.

[Che01] Keith Cheverst, Nigel Davies, Keith Mitchell, Christos Efstratiou, Using Context as a Crystal Ball: Rewards and Pitfalls, In *Personal and Ubiquitous Computing*, v5, Springer-Verlag London, 2001.

[Dey01] Anind K. Dey, Understanding and Using Context, In *Personal and Ubiquitous Computing 2001*, Volume 5 Issue 1, Springer-Verlag London, 2001.

[Edw01] W. Keith Edwards, At Home with Ubiquitous Computing: Seven Challenges, In G. D. Abowd, B. Brumitt, S. A. N. Shafer (Eds): *Ubicomp 2001*, LNCS 2201, p256-272, 2001.

[EMWeb] *The Homepage of Empirical Modelling*, Online at http://www.dcs.warwick.ac.uk/modelling.

[Fis00] Gerhard Fischer, User Modeling in Human-Computer Interaction, in *10th Anniversary Issue of "User Modeling and User-Adapted Interaction(UMUAI)*, 2000.

[Goo90] D. Gooding, *Experiment and the Making of Meaning: Human Agency in Scientific Observation and Experiment*, Kluwer Academic Publishers, 1990.

[Gre99] Thomas Green, *The BALMORAL Central Heating Controls*, Online at http://www.ndirect.co.uk/~thomas.green/workStuff/devices/controllers/HeatingB.html, December 1999.

[Jos02] Mini K. Joseph, Now, it's autonomic computing, *Times News Network*, July 2002.

[Kul00] Bill Kules, *User Modelling for Adaptive and Adaptable Software Systems*, Department of Computer Science, University of Maryland, April 2000.

[Lan02] M. Langheinrich, V. Coroama, J. Bohn, M. Rohs, *As we may live – Real-world implications of ubiquitous computing*, Online at www.inf.ethz.ch/vs/publ/papers/implications-uc2002.pdf, April 2002.

[Neg96] Nicholas Negroponte, *Being Digital*, Coronet London, 1996.

[Odl99] Andrew Odlyzko, The Visible Problems of the Invisible Computer: A Skeptical Look at Information Appliances, First Monday, 1999.

[Run02] Jaratsri Rungrattanaubol, *A Treatise on Modelling with Definitive Scripts*, PhD Thesis, Department of Computer Science, University of Warwick, April 2002.

[Spi00] Dag Spicer, *If You Can't Stand the Coding, Stay Out of the Kitchen*, History of Computing #3, Dr. Dobb's Journal, Online at http://www.ddj.com/columns/history/2000/200006hc/do200006hc001.htm, 2000.

[Sun99a] Pi-Hwa Sun, *Distributed Empirical Modelling and its Application to Software System Development*, PhD Thesis, Department of Computer Science, University of Warwick, UK, July 1999.

[Ten00] David Tennenhouse, Proactive Computing, *Communications of the ACM*, v43n5, May 2000.

[Won03] Allan Wong, Before and Beyond Systems: An Empirical Modelling Approach, PhD Thesis (submitted), Department of Computer Science, University of Warwick, January 2003.