Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 208 (2008) 41–55
www.elsevier.com/locate/entcs

Exploring an Option Space to Engineer a Ubiquitous Computing System

Michael D. Harrison[a,1] Christian Kray[a] José Creissac Campos[b]

[a] *Informatics Research Institute, Newcastle University, U.K.*

[b] *Departamento de Informática/CCTC, Universidade do Minho, Braga, Portugal*

Abstract

Engineering natural and appropriate interactive behaviour in ubiquitous computing systems presents new challenges to their developers. This paper explores formal models of interactive behaviour in ubiquitous systems. Of particular interest is the way that these models may help engineers to visualise the consequences of different designs. Design options based on a dynamic signage system (GAUDI) are explored using different instances of a generic model of the system.

Keywords: interactive systems, model checking, ubiquitous computing, human computer interaction

1 Introduction

Making built environments smart by using public displays, location sensors and mobile devices is becoming increasingly feasible in a range of environments. Such environments have the potential to modify the behaviour and experience of their inhabitants. They provide information and services to support the various roles and activities of the people within these environments. The development of such systems presents interesting human and system engineering challenges.

Over the last few years there has been significant development of technologies, platforms and middlewares that will support these systems. There have also been valuable ethnographic studies designed to observe the effects that these technologies have in transforming the working or living arrangements of those who occupy these spaces. There is however little research devoted to the issues of engineering these systems, particularly in relation to human engagement and experience. While some researchers focus on the technological challenges of building sensors, actuators and

[1] Email: michael.harrison@ncl.ac.uk

1571-0661
doi:10.1016/j.entcs.2008.03.106

computing devices that can be integrated into 'real' environments, an equally important aspect relates to investigating and improving how people engage with these technologies and how these issues can be understood in the design stages. Users can still find it difficult to engage and appropriate the supported activities. It is not surprising therefore that discussion about the definition and the utility of *ubiquitous computing* (consider for example [1,2,16]) continues despite many substantial and significant achievements.

Users of ubiquitous computing systems are typically immersed within them, surrounded by and affected by their consequences. Although the user might engage with the environment, the system in which she is embedded makes assumptions based on where the user is, what she is doing and her personal tastes. Weiser describes "calm computing" that "weave themselves into the fabric of everyday life until they are indistinguishable from it" [19]. Others have used terms like implicit interaction [17] or proactive systems [18] to describe the potential of these systems to make assumptions about the user. More recently authors like Rogers [16] have reminded the community of the importance of engagement commenting that, in reality, systems that anticipate the users' wishes or desires can be controlling or confusing.

Because of the potential complexities of these systems, models are required to enable designers to consider the implications of choices about styles of interaction and the effects on action of context. Analysis based on modelling is required before fielding the system so that design decisions may be considered *early* before expensive commitments have been made. This analysis should consider interaction issues associated with the usability of the system, the device and the experience that users have of the system as well as exploring the potential for modifying the behaviour of these users. This paper explores these issues by addressing: how to understand novel requirements relating to ubiquitous systems in built environments; how these are mapped into the design; and how modelling techniques can be used to explore an option space for the design of the system. These ideas will be explored by considering a system to provide navigation support through situated displays.

2 Related work

There are a variety of ways that usability can be explored in ubiquitous computing systems. Analysis may use scenarios aimed at capturing typical or extreme situations. It may use personas aimed at challenging designer assumptions about who might be using the system. When the user engages with the system, the effects of the interaction both personally in terms of personal devices, and publically in terms of shared displays in the built environment, should be transparent and intuitive. The system should not take the initiative away from the user in a controlling way. Informal analysis techniques such as cognitive walkthrough, co-operative evaluation or heuristic evaluation can provide useful early feedback of some aspects of the design. However new and challenging characteristics are of particular relevance in these systems:

(i) how context can change the meaning of the actions [12] in ways that might confuse users

(ii) how context might trigger changes in the information that flows around the system

(iii) how the right information reaches the right person at the right time, without controlling or confusing the user

(iv) how this information and the actions that can be carried out will change the behaviour of users.

A well designed system can create a designed experience for the user. For example, consider designing an environment in a hospital to allay the anxiety of an out-patient visiting an unfamiliar hospital, aiming to reach a specified consulting room for a specific appointment time. Properties such as those relating to experience can be difficult to analyse. They are critically dependent on physical and textural characteristics only available in the live target environment. A prototype system explored in the live environment may have safety or commercial consequences, consider for example a prototype running in a busy airport. Deploying a system that is close to product when many downstream design commitments have already been made will be expensive to redesign.

An important part of requirements elicitation is to establish what scenarios or personas represent the important characteristics of the design to form the basis for analysis. This process can be carried out through interview. However other techniques help to elicit the scenarios that are used. For example, a richer understanding of the experience that users have may be obtained by using cultural probes [7] to elicit snapshot experiences. These probes may be used to discover properties. Tools can be used to check these models to produce counter-examples that may themselves form the basis for valuable scenarios. These cultural probes are fragments provided by users that can be used to help understand experience of an existing system. The aim is that these snapshots establish what is required of a new design. Eliciting snapshots involves subjects collecting material: photographs, notes, sound recordings, that they believe capture important features of their environment. These snippets may make sense as part of a story. The information gleaned may help understand characteristics of the current system that cut across a range of scenarios.

A range of formal models can be developed capturing different characteristics of ubiquitous systems that can be explored either through simulation or through property checking. This can happen prior to implementing a system and therefore provide the means to explore, relatively cheaply, alternative design options. Two key concerns in the modelling process are to identify relevant properties and to capture aspects relating to the deployment of information, scheduling of activities, and issues of location.

This paper explores a particular kind of model that describes aspects of a user's implicit action while moving through a built environment. It aims to characterise those properties that relate to experience or user behaviour. This space includes

public displays, mobile devices and sensors [8]. Interaction between user and activity may be impacted unexpectedly by location, the current status of the task at hand and other external effects generated by the ubiquitous environment. It may be affected by other users carrying out different tasks. A similar analysis [8] focussed on timing issues, in particular the timeliness and relevance to person and location of message delivery, using the Uppaal model checking tool [3]. In most previous research in formal methods in interactive systems models are used to explore the interaction between user and device. For example these models are used to characterise and analyse mode confusion problems, see for example [12]. It may also be appropriate to explore the way in which information in the environment provides resources for the user as they carry out their activities in the style discussed by [4]. To illustrate the analysis of whole built environments and implicit action an example is used based on a dynamic signage system installed at Lancaster University. Alternative designs and properties of the system are explored. The models and properties incorporate simple user assumptions.

3 The GAUDI system

The GAUDI system [10] is designed to help people find their way in a complex built environment such as a large office building. Static signage and directories are replaced with dynamic situated displays. The user interface of the implemented system consists of two parts. The interface provided by GAUDI shows labelled photographs of all the office owners as well as the meeting rooms in the building on a large touchscreen. A visitor can touch the photograph or name of the person she wants to visit. This will initiate navigation support. Situated displays are arranged throughout the building (similar to the interactive doorplates described in [9]). When a visitor selects the person they wish to visit on the large touchscreen, all the displays in the building will show arrows that point towards their office. Together they indicate the path a visitor needs to follow. GAUDI will continue to show these arrows either until the visitor touches the display at her target location, or until a time out period has expired. GAUDI, in its original form, may be considered to be calm technology:

- it does not interfere with people's 'everyday life', that is it (subtly) enhances rather than replaces an existing analogue system
- it can be used without thinking about it, i.e., the only explicit interaction required is touching a destination (and possibly the display at the target location)
- it provides support when and where it is needed, e.g., if displays are present at all decision points.

A number of properties of this system reflect these characteristics of calmness.

P1: the directions provided will always get the visitor to their required destination subject to simple assumptions about the passenger following the directions shown on the nearest dynamic sign

P2: the throughput of new visitors who need this facility will be sufficiently low

that the arrival of a new visitor will not have the effect of changing the route and therefore confusing the visitor currently following the path.

The derivation of properties that are relevant to a particular system may be achieved through a variety of processes including obtaining snapshot experiences from users of existing systems [8]. Consider the following plausible example:

S1: photographs produced by subjects of anonymous and narrow corridors with comments like:
- "It is very easy to lose my bearings in spaces such as this one"
- "When I get lost who do I ask, all the doors are closed?"
- "How near am I to my destination?"

In practice the variety of elicited snapshot experience requires organization to ensure that subsets of facilities are not neglected. Snapshot experiences may be used to trigger further narratives. The analyst might enquire of a user who has generated a snapshot: "Can you think of situations where this particular feature has been important?" By these means they may inspire a scenario that would not otherwise have been gathered. These documented experiences can also be converted into properties that the new design should satisfy. Hence the comment relating to S1: "It is very easy to lose my bearings in spaces such as this one" could lead to properties such as:

P3: when a visitor moves into a location where there is more than one possible exit the sign should indicate the direction which the visitor should travel

P4: it should always be possible to see the next sign in the path

P5: the user should be able to visualise where they have come from and where they are going to and how far they have got in their journey

P6: if the visitor makes a wrong turn then signs should continue to lead the visitor to their destination.

One problem with the current version of GAUDI is that if the visitor is too slow getting to the destination, they will either lose the route and therefore be stranded or they will follow signs to the office selected by the next visitor to the building. Another problem is that a new visitor will have to be told to wait until the delay has completed or the previous visitor has reached their destination before choosing the room that they wish to reach. In practice these problems will not be an issue if the delay is long enough and visitors who are unfamiliar with the environment arrive infrequently enough. However visitors may arrive more frequently than anticipated or arrive in bursts. The next section explores a model of GAUDI like systems that can be used to explore possible alternatives that avoid some of these problems. An aim is to develop a family of models that can be used easily to explore alternatives.

4 Modelling the System

The Uppaal system [3] is used to explore alternative designs. This is partly because it was envisaged that timing issues would be important in understanding the system

and partly because Uppaal provides useful and robust facilities for animating the system model. In Uppaal, processes are modelled as communicating timed automata and can be animated to explore specific paths based on derived scenarios. Properties of the system, such as P1-P6 in the previous section, can be investigated using model checking techniques. An important focus of these models is how users interact with the system in terms of the implicit interactions that take place in the built environment. This will be discussed further in later sections. It must be emphasised that a variety of approaches could have been used to explore models of these system alternatives. Particular approaches would have made it easier to capture other aspects of the models. For example, KLAIM [6] or π calculus [14] would have been more expressive of location and mobility. While others would have allowed an alternative style of analysis. For example, PRISM [11] would have made it possible to explore statistical properties of the system.

The common characteristics of these systems are that they all involve public displays, sensors and mobile devices. In principle a generic form of the model may be tailored to appropriate versions of the system. In this example the model's structure is based on a publish-subscribe communication mechanism. A central distributor broadcasts messages to listening processes (in this system the processes are either visitors or sensor/displays). The physical and human capabilities are also defined in the model. The model emulates a two storey building with a staircase linking the two floors. The space is defined using co-ordinates. The co-ordinates describe regions of the room rather than points — that area around the co-ordinate in which the sensor / display recognises the presence of another device. These regions may be rooms, corridors or staircases. The model captures the visitor or visitor's mobile device reading the information on the sensor / display when in the region of a particular co-ordinate position.

Three versions of the system were modelled indicative of a range of possible designs. Each of the designs involves a different degree of explicit interaction. In the implemented version of the system the only user engagement that can occur takes place at reception when the visitor selects a destination and at the destination itself when they press the display at the doorplate having reached the destination office. The two variations reflect changes in the proactive nature of the interaction and the degree of engagement of the users. Version 2 makes the situated displays sensitive to the presence of visitors to avoid the problem that arises because the frequency of completely novice visitors will make the one at a time approach of the existing system unacceptable. Version 3 acknowledges the possibility that more than one visitor can be in the same space at the same time and so supports user initiated action to obtain further information updates directly to the visitor's personal device. To simplify the model for exposition purposes some of the aspects of timing that are required to ensire that reachability is achieved are not included in the models.

4.1 The distributor

Two models for the the distributor were developed. They are presented in Figure 1 and Figure 2 (state transitions have been labelled to help the description that

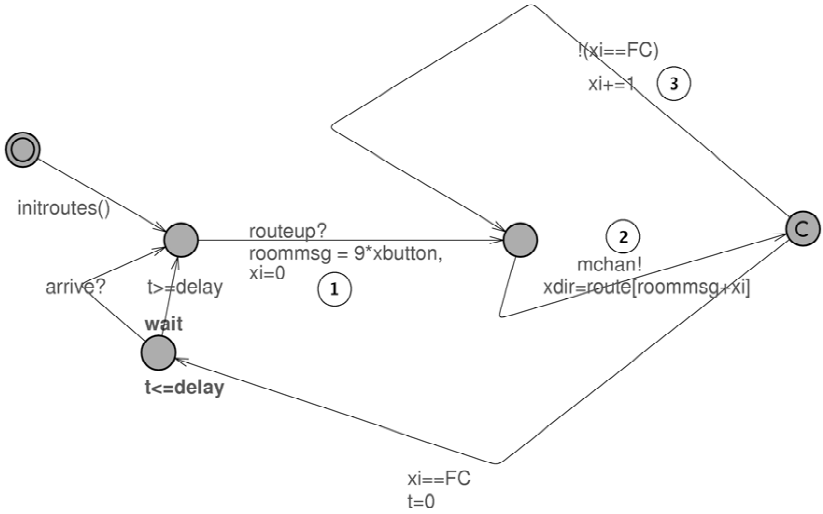

Fig. 1. The GAUDI distributor

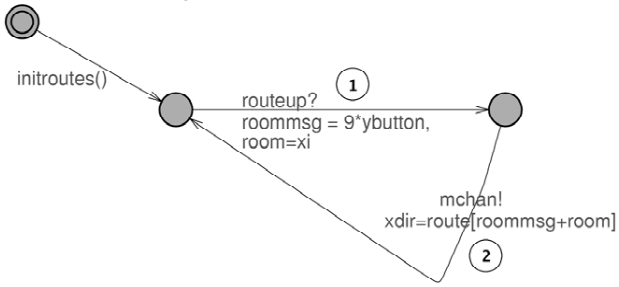

Fig. 2. The sensitive distributor

follows). The first model (Figure 1) is used to characterise the distributor of the original version of the GAUDI system (which will be referenced as version 1). The second model (Figure 2) is used to characterise the distributor of the two additional versions that have been developed (versions 2 and 3).

The distributor has a general format. After initialisation, its behaviour is triggered by a request (`routeup?` – ① in both Figure 1 and Figure 2). In version 1 the request to the distributor is made when the visitor is in reception. In the case of versions 2 and 3 (Figure 2) the triggering request occurs when the sensor / display recognises the presence of the visitor in its space. The information sent with the request tells the distributor which room number the visitor is looking for (the room identifier `roommsg` is used to index `route` which is an array of directions holding all the paths along the space – and select the appropriate route), and which display to use (needed in versions 2 and 3).

As a result of being triggered the distributor broadcasts one or many messages using the channel `mchan` ②. In the case of version 1, messages are sent to all locations (the process counts through the location values using the variable `xi` ③). On each increment the direction associated with the route at that location is sent (using the shared variable `xdir = route[roomsg+xi]`). In the case of version 2 and

version 3 the `routeup?` request carries information regarding the current position of the visitor using the shared variable `xi` (`room = xi`), and a single message is broadcast using `xdir = route[roommsg+room]`. These messages are broadcast to all the combined sensor / displays in the system. Through shared variables, the messages contain both the direction that is to be displayed by the display / sensor and a tag (in this case the room) that is used by the display / sensor to decide whether the message is relevant.

When the distributor has completed distribution (in version 1 Figure 1) it either waits until a timeout has elapsed (`t>=delay`) or it is notified by the visitor that it has arrived `arrive?` before allowing another to request a destination. This handover is not required for versions 2 and 3.

The direction information is absolute (`North`, `South`, `East`, `West`, `Up` or `Down`). It is assumed that the hand-held device of the visitor could be used to adjust direction to the orientation of the visitor. This aspect of the design is not considered in more detail in these models. The implemented version of GAUDI differs from these models in that each sensor/display holds the whole route so that if the display is moved to a new location it picks up the new direction and adjusts itself accordingly. Maintaining multiple copies of arrays of directions is not an option using the sort of modelling technique described. Because the time to distribute all the messages for the space is insignificant, this can be seen as practically equivalent.

4.2 The sensor / display process

A single process is modelled that combines sensor and display. In all versions of the process, the visitor sends a request to read information (using `route?` ①) and the sensor / display then sends a result (using `direct!` ②) which includes the direction to be displayed contained in shared variable `xsign`. Version 1 (Figure 3) receives messages from the distributor (via `mchan` ③) and updates its own display using the function `updatedisplay()`. This function checks whether the message is relevant to the current location. The visitor can only read the relevant display when they are "within range of the sensor". This is modelled by linking the visitor's co-ordinate position to a channel that can be used to communicate with the sensor / display in the visitor's location if there is one (see section 4.3).

Version 2 and 3 of the sensor / display includes a further trigger (Figure 4) that requests an update from the distributor `routeup!`. The requests include a message that contains the route number and location of the visitor who requires the information (`ybutt` and `xloc`). As a result of the request this version of the sensor / display receives a specific broadcast message via `mchan` as before.

In summary the display/sensor has three important characteristics.

(i) It knows its location. The relevant direction and route channels that are used in the mapping of co-ordinates are part of the instantiation of the process.

(ii) It receives messages from a central server. It filters those messages that relate to its location.

(iii) It represents an access point for a "mobile" process which communicates with

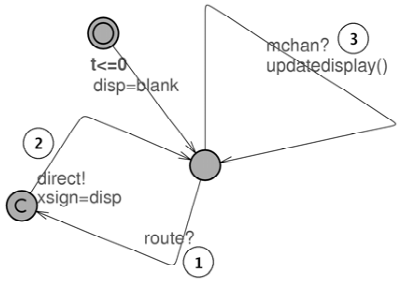

Fig. 3. The combined sensor and display

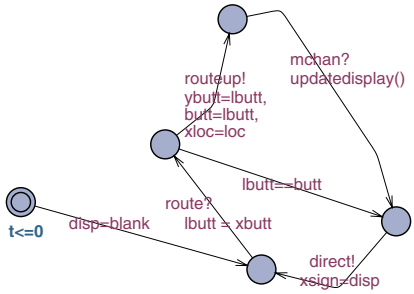

Fig. 4. The sensitive sensor and display

it. The mobile process calculates the channels from its location.

4.3 The visitor

The visitor process describes simple human characteristics of the visitor as well as the visitor's personal device. It knows:

- its position as defined by an (x, y) co-ordinate
- when it is in the "neighbourhood" of a sensor / display

In version 1 (Figure 5) the visitor communicates its requested route to the dispatcher using `press`. This initiates the download of the whole route to the displays in the space. In all versions the visitor process detects its position using a function `sensorlink()` that maps its co-ordinates (initially the visitor is in the reception area which is on the ground floor, co-ordinates $(0, 0)$). This function returns the channel `loc` that is to be used to communicate with the nearest sensor / display. If the value returned is `NS` then there is no nearest sensor but if there is a sensor, a request is made to the sensor / display using `route[loc]!` and the value returned via `direct[loc]?` is stored in `dir`. This direction is used to update the co-ordinates using the function `updatecoord()`. The visitor process "remembers" the direction last read if there is no sensor / display in a given location (i.e., `loc` has the value `NS`). The visitor process follows the direction indicated by the sign (using the function `updatecoord()`).

This description of a visitor process captures assumptions about how a visitor will behave in the environment. Clearly different user assumptions could be made. In the present case further assumptions are made about the visitor's movement. The

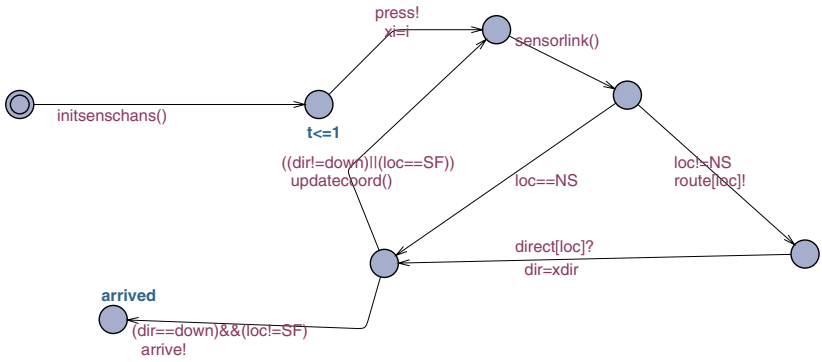

Fig. 5. The visitor

visitor reads and updates the direction of travel correctly in response to the display that is read. If the direction is "North" then the y co-ordinate is incremented, if "West" then the x co-ordinate is decremented. When the visitor reaches a direction "Down" and is not in a stairwell then it has reached its destination. If it is in the stairwell on the first floor and it reads "Down" then it descends to the ground floor and so on.

The model makes further assumptions about how the visitor responds to possible collisions. In the event that movement would cause collision, hitting a wall, then the co-ordinate that would have been updated is unchanged and the other co-ordinate is updated in the same direction (if the x co-ordinate was to be incremented then the y co-ordinate would be incremented and so on). This policy works in the present case because sensor / displays are located in the corners and therefore a visitor always eventually reaches a location where there is a display.

Each of these assumptions may be questioned. They capture hypotheses about how people might behave in the smart environment. For example if a stochastic model like PRISM [11] were used instead further assumptions might be made about how the visitor remembers the direction with a given probability and in so doing estimates of likely error behaviour could be explored.

In summary the model supports a generic notion of mobility achieved by relating space to the channels that are used to communicate with the display / sensor. The visitor process in addition captures a set of simple assumptions about the user which can be explored further:

(i) It can read the direction from the sensor display.

(ii) It remembers the direction it has read and continues to use it even when the display is not visible.

(iii) It moves correctly in response to the direction that has been read.

Each of these elements could be manipulated to explore types of user error.

All models use the same floor layout which is different from Lancaster University but is of similar complexity. Version 2 of the system assumes that the sensor is sensitive to the visitor's position and therefore the visitor variant sends the route each time (Figure 6). The third version of the model performs the same activity

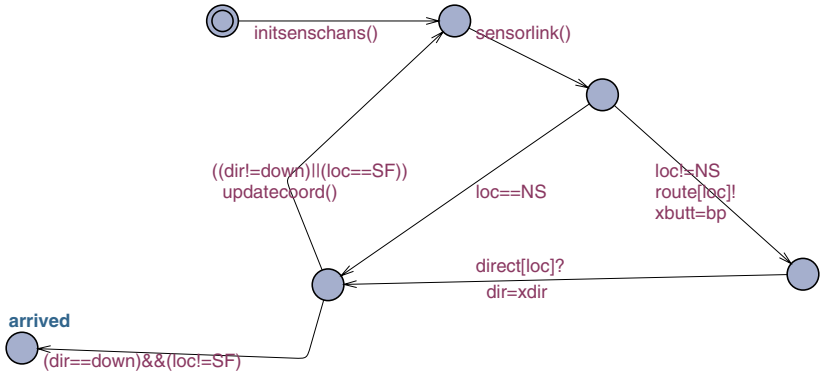

Fig. 6. The visitor who can elect to receive a direction

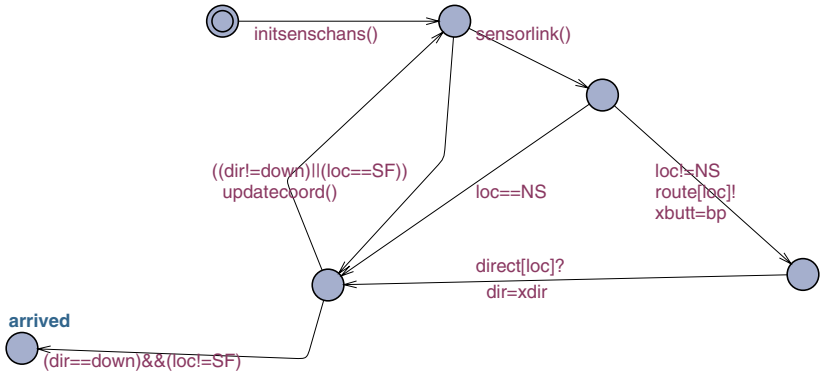

Fig. 7. The explicit visitor process

as the second except that the elective refreshing of the visitor's handheld display is also modelled. An additional unlabelled branch in the model specifies the possibility that the visitor does not request an update to its display. In this case direction information used by visitor is determined by the last request that was made by the visitor (this happens non-deterministically in the model). Hence if the user "forgets" to update the display it will continue to use a prior direction even though it could have been updated through a visitor's request.

5 Evaluation

Modelling the system allows exploration of the implications of alternative designs by exploring sequences of modelled states. These state sequences can be thought of as paths that provide the basis for scenarios. There is no space for detailed analysis of the system. Instead the three types of activity that were carried out are briefly described.

5.1 Scenario exploration

The model provides a framework within which specific scenarios can be explored. Scenarios that may have been gathered through a process of interviewing users can

be "stripped down" to steps in the model. When animated, the steps taken within the model may be understood and visualised by the analyst in the broader context of the physical textures, siting of displays and other features of the real system. Hence the model animation is used as a trigger for the domain specialist to rehearse and to visualise what the designed system would be like. The model generates questions for the designer about the choices that have been made. These questions can be explored and discussed with a usability specialist or even re-expressed as a narrative that could be of value to explore with users and to allow them to experience the properties that are important.

Additionally, when properties fail in Uppaal, a trace is provided by the tool. These traces need to be explored (sometimes extended) through animation in order to build an understanding of the root of the problem being highlighted.

5.2 Property exploration

The model provides a basis for exploring properties. These properties could be identified through a process such as the one described for eliciting snapshot experiences. They may be automatically checked against the model. This process of property checking may generate traces that would form the basis for further scenarios [5]. Hence for example the version of these models that were used for checking properties included specific timing information. This was to ensure that the system would not dwell endlessly at a particular state. The following properties have been checked.

- Reachability: checking that the model behaves in a way that is consistent with the proposed system. For example that the user will arrive at the required destination if they follow the signs. This can be achieved using properties such as `A<>(P1.dir==down)&&(P1.loc!=SF)`, that the visitor (`P1`) will always reach a location that is not the stairwell where the direction of the display is down. Note that timing in the particular model used was designed so that the model always progresses because without it it is always possible that the process can dwell in a state forever and therefore fail to reach its destination.
- Visibility: checking that there is always another sign in the line of sight of the current display in the direction that the visitor should travel. This is achieved by adding an observer process, see below, and checking that at each location (loc) the visitor reaches where there is a display then there is a further display visible (that is in line and not concealed by walls) in the direction defined by the display in loc.

It is also possible to use this model to check security and privacy properties, for example to check that the public display closest to the user can never combine with information displayed on the user's personal device to betray private information about another user.

Situations where a property breaks can be at least as interesting as whether the property is satisfied. Counter-examples may demonstrate failures in the system but they may also indicate situations where properties of the information may not

be satisfied but may be unproblematic from a usability point of view. Alternative models that make different assumptions about visitor dwell time will enable the exploration of counter-examples where visitors might spend longer than expected reading displays and travelling between locations. These counter-examples to properties may be used as the basis for scenarios.

5.3 Observer processes

Observer processes provide further opportunities to check properties of the system. They can be used to act like observational scientists spotting interesting events and calculating whether these events have properties in the context of the ongoing execution of the system. Hence for example an observer was used to check whether a visitor can always see the next sign in the direction of the arrow that is on the nearest sensor / display. In order to achieve this the observer process is signalled each time a visitor reaches a new location, and a message is sent to the observer giving the direction it reads from the sensor / display and the visitor's destination. The observer uses this information to calculate from the current location whether there is a straight line in terms of the (x, y) co-ordinates to another sensor / display. The observer sets a variable depending on whether the display can be found. Other examples of observer processes are described in [8].

6 Discussion and Conclusions

An important benefit of formal modelling of human interactions in ubiquitous systems is the ability to evaluate user characteristics *prior to building* the system. This applies both to testing whether user requirements are met in a particular model and to improving a model iteratively so that it eventually meets the set requirements. This may enable developers to modify the current design/model in a targetted way and speed up the development process. Another benefit is that the elicited traces are gathered by a formal process and therefore the designer can be quite clear why the scenario based on the trace is of interest.

One potential disadvantage of the proposed approach relates to scalability. While there are no inherent limitations, it is possible that models become too complex to fully check them within reasonable time or memory limits. This can result either from poor modelling (e.g., using the wrong abstractions) or the fact that the model requires multiple instances of the same process. Ways to overcome this particular problem include the use of multiple, smaller models (potentially at the expense of being able to prove certain properties for the entire system) or multi-tiered models with several layers of abstraction.

A second issue is the mapping between informal usability requirements, the model and the implementation of the system. In order for the model to have any value, it needs to be tightly linked to the system implementation, i.e., the system needs to implement the model. Similarly, the formal requirements used to check against the formal model need to accurately represent the requirements identified in the elicitation phase. Potential solutions to these issues include formal and/or

(semi-)automatic processes to derive an executable implementation from the formal model or tools supporting both processes (compare for example, [13,15]).

Finally, creating, evaluating and modifying a formal model of a ubiquitous system introduces a further step into the development process, which can incur increased costs and lengthen the time to completion of a project. In addition, formal modelling may require a different set of skills from developers, which may have the same effect. This issue is however counterbalanced by the advantages listed above as well as by the potential savings resulting from identifying problems early on in the development process.

The main value of Uppaal was through its provision of animation facilities and time. The modelling technique makes it necessary to use rather complicated "patterns" to capture communication and to support mobility. Another problem with the techniques is that there are no clear pathways towards stochastic modelling. Questions remain about whether modelling such as this will require not only consistent families of model but also a variety of modelling techniques.

By making interaction requirements explicit, modelling can also pave the way to define more clearly what is meant by 'calm computing' in the context of a particular system. The next step is to integrate and use the approach described in this paper in the context of a larger and more complex project.

Acknowledgement

Michael Harrison and José Creissac Campos acknowledge with thanks EPSRC grant EP/F01404X/1 and FCT/FEDER grant POSC/EIA/56646/2004.

References

[1] Abowd, G. and E. Mynatt, *Charting past, present and future research in ubiquitous computing*, ACM Transactions on Computer-Human Interaction **7** (2000), pp. 29–58.

[2] Abowd, G. D., E. D. Mynatt and T. Rodden, *The human experience*, Pervasive Computing Magazine **1** (2002), pp. 48–57.

[3] Behrmann, G., A. David and K. Larsen, *A tutorial on uppaal*, in: M. Bernardo and F. Corradini, editors, *Formal methods for the design of real-time systems*, number 3185 in Springer Lecture Notes in Computer Science, Springer-Verlag, 2004 pp. 200–236.

[4] Campos, J. and G. Doherty, *Supporting resource-based analysis of task information needs.*, in: S. Gilroy and M. Harrison, editors, *Interactive Systems: Design, Specification and Verification 12th International Workshop, DSVIS 2005, Newcastle upon Tyne, UK, July 2005*, number 3941 in Springer Lecture Notes in Computer Science (2006), pp. 188–200.

[5] Campos, J. and M. Harrison, *Model checking interactor specifications*, Automated Software Engineering **8** (2001), pp. 275–310.

[6] De, R., Nicola, G. L. Ferrari and R. Pugliese, *KLAIM: a kernel language for agents interaction and mobility*, IEEE Transactions on Software Engineering **24** (1998), pp. 315–330.

[7] Gaver, W., T. Dunne and E. Pacenti, *Design: cultural probes*, ACM Interactions **6** (1999), pp. 21–29.

[8] Harrison, M., C. Kray, Z. Sun and H. Zhang, *Factoring user experience into the design of ambient and mobile systems*, in: G. van de Veer, P. Palanque and J. Wesson, editors, *Engineering Interactive Systems*, 2007, accepted for publication, Springer Lecture Notes in Computer Science.

[9] Kray, C., K. Cheverst, D. Fitton, C. Sas, M. Patterson, J.and Rouncefield and C. Stahl, *Sharing control of dispersed situated displays between nomadic and residential users*, in: *Proceedings of MobileHCI'06*, Espoo, Finland, 2006.

[10] Kray, C., G. Kortuem and A. Krüger, *Adaptive navigation support with public displays*, in: R. S. Amant, J. Riedl and A. Jameson, editors, *Proceedings of IUI 2005* (2005), pp. 326–328.

[11] Kwiatowska, M., G. Norman and D. Parker, *PRISM: Probabilistic symbolic model checker*, in: T. Field, P. Harrison, J. Bradley and U. Harder, editors, *Computer Performance Evaluation: Modelling Techniques and Tools Proceedings 12th International Conference, TOOLS 2002*, number 2324 in Springer Lecture Notes in Computer Science (2002), p. 200.

[12] Loer, K. and M. Harrison, *Analysing user confusion in context aware mobile applications*, in: M. Constabile and F. Paternò, editors, *INTERACT 2005*, 3585 (2005), pp. 184–197.

[13] Mellor, S. J. and M. J. Balcer, "Executable UML: A Foundation for Model-driven Architecture," Addison Wesley, 2002.

[14] Milner, R., "Communicating and mobile systems: the pi-calculus," Cambridge University Press, 1999.

[15] Murphy, G. C., D. Notkin and K. Sullivan, *Software reflexion models: bridging the gap between source and high-level models*, SIGSOFT Softw. Eng. Notes **20** (1995), pp. 18–28.

[16] Rogers, Y., *Moving on from Weiser's vision of calm computing: Engaging UbiComp experiences*, in: P. Dourish and A. Friday, editors, *Proceedings of Ubicomp 2006* (2006), pp. 404–421.

[17] Schmidt, A., *Implicit human computer interaction through context*, Personal and Ubiquitous Technologies **4** (2000), pp. 191–199.

[18] Tennenhouse, D., *Proactive computing*, Communications of the ACM **43** (2000), pp. 43–50.

[19] Weiser, M., R. Gold and J. S. Brown, *The origins of ubiquitous computing research at PARC in the late 1980s.*, IBM Systems Journal **38** (1999), pp. 693–696.