I Bet You Look Good on the Wall: Making the Invisible Computer Visible

Jo Vermeulen, Jonathan Slenders, Kris Luyten, and Karin Coninx

Hasselt University - tUL - IBBT,
Expertise Centre for Digital Media,
Wetenschapspark 2, B-3590 Diepenbeek, Belgium.
[jo.vermeulen,kris.luyten,karin.coninx]@uhasselt.be
jonathan.slenders@student.uhasselt.be

Abstract. The design ideal of the *invisible* computer, prevalent in the vision of ambient intelligence (AmI), has led to a number of interaction challenges. The complex nature of AmI environments together with limited feedback and insufficient means to override the system can result in users who feel frustrated and out of control. In this paper, we explore the potential of visualising the system state to improve user understanding. We use projectors to overlay the environment with a graphical representation that connects sensors and devices with the actions they trigger and the effects those actions produce. We also provided users with a simple voice-controlled command to cancel the last action. A small first-use study suggested that our technique might indeed improve understanding and support users in forming a reliable mental model.

1 Introduction

The visions of ambient intelligence (AmI) and ubiquitous computing (Ubicomp) share the goal of moving computers into the background, thereby making them effectively *invisible* to end-users. This design ambition is clearly present in Mark Weiser's vision of Ubicomp [14] as well as in AmI-oriented efforts such as the EU-funded Disappearing Computer Initiative [12].

If computers are to be so natural that they become invisible in use, they will often need to function on the periphery of human awareness and react on implicit input. This kind of system is called context-aware [11]: it is able to interpret and adapt to the user's current situation or *context*. These systems often react on a context change by taking a (presumably desired) automatic action on behalf of the user. In an ideal world, where the sensed context would be 100% accurate, users would then indeed not "notice" the computers embedded in their environment, but would only experience the right actions being "magically" performed at the right time.

The previous assumption is unrealistic, however. There are many aspects of context (e.g. human aspects such as our mood) that cannot reliably be sensed or inferred by machines [4]. Moreover, our behaviour is unpredictable and impossible to model accurately by computers [13]. From these arguments, it can be

concluded that it is infeasible to allow context-aware computer systems in Ubicomp or AmI environments to act *without user intervention*. However, before users are able to intervene, they should first understand how the system works and what it is trying to do. When something goes wrong, the system needs to present itself and the way it works to end-users – or essentially become *visible*.

Making clear how an AmI environment functions is not always easy to achieve because of the heterogeneous nature of these environments (they often contain several displays, speakers, sensors and computers of different sizes) and their complex behaviour. Adding to this problem is the fact that due to the focus on the invisible computer, Ubicomp and AmI systems often have little support for traditional user interface concerns such as feedback, control, and indeed "visibility" [3]. We are not the first to make these observations: a number of researchers have pointed out problems along these lines such as Bellotti et al. [4,3], Rehman et al. [9] and Barkhuus and Dey [2]. The heart of the problem lies in the fact that the lack of visibility inhibits users from forming a correct mental model of the system and exacerbates the Gulf of Execution and the Gulf of Evaluation [7]. As a consequence, users have difficulties predicting the behaviour or even the available features of the system [9]. Moreover, there is often no way for the user to override the actions taken by the system, which results in users who feel out of control [2]. Bellotti et al. [4] propose two key principles that are necessary to improve the usability of context-aware applications: *intelligibility* (or the system's capability of being understood) and *control*.

In this paper, we present a technique to make the invisible computer *visible* for end-users. We use projectors to overlay the environment with a graphical representation that shows the location and state of the different sensors and input/output devices such as displays or speakers. When the system acts on behalf of the user, an animation is shown that connects a system action with its cause and effect. Of course, constant visualisations might be distracting and contrary to Weiser's idea of calm computing [14]. We therefore believe our technique is useful mainly as a "debug mode" for end-users. The visualisations might be hidden when users have gained more experience with the system, and be called upon again whenever users have difficulties understanding the system's behaviour or when they want to know more about the reasoning behind a certain system action. Our technique allows users to consult the system state whenever necessary, thereby improving the system's intelligibility. Users receive real-time feedback about actions that happen, when they happen. In addition, a primitive control mechanism is provided that allows users to cancel an action in progress. We explored the usefulness of our technique in an informal first-use study. Results suggested that it might indeed improve understanding and support users in forming a reliable mental model.

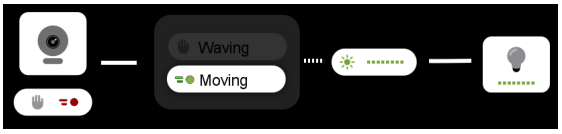
2 A Graphical Representation of Behaviour

A simple graphical language was developed to visualize the relationships between sensors or devices and the actions executed by the system. This allows users to

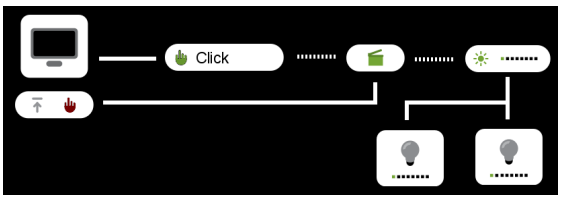
get an overview of the system state at a glance. When an action is executed by the system, an animation is shown to reveal the links between this action and the different devices or sensors in the environment.

2.1 Visualising the Environment and Its Behaviour

Each sensor or input/output device (e.g. a camera, speaker or display) is visualised at its physical location in the environment with an icon and a label . These icons allow users to get a view of the devices that are present in their environment. Below the icon of each *input* device or sensor, a separate label is drawn that displays the possibilities of the device and its current state using smaller icons. *Output* devices feature only an icon and no separate label. The icon of an output device embeds its current state. For example, Fig. 1(a) shows an icon and label for a webcam (an input device) on the left and an icon for a light (an output device) on the right. In this (fictional) example, the webcam can detect the events "waving" and "moving", as indicated by the small icons in the label. In Fig. 1(a), the motion detection state is active and therefore highlighted. The light's state corresponds to its current intensity and is displayed as a horizontal bar.



(a) Motion in front of the webcam (input) triggers the light (output). The event "waving" of the webcam is now inactive, but could trigger another action



(b) A chain of events: touching the screen results in a movie being played (on the same screen). This, in turn, results in the lights being dimmed.

Fig. 1. Mockups of example trajectory visualisations.

We define a *trajectory* as a visualisation between two or more objects in the environment. A trajectory consists of four parts: a source device; the event that happened at this device; an action to be executed; and one or more target devices that are impacted by the action. Between each of these, lines are drawn. Dotted

lines are used between events and actions, while connections between devices and other objects use solid lines.

An example trajectory is shown in Fig. 1(a). Here the webcam detects motion, which triggers an action that turns on the lights. This action, in turn, impacts the light on the right side of the figure. Note that the small state icons are repeated together with a textual description. The "Waving" state is shown semi-transparently to indicate that it is not active. A bit further to the right, a graphical representation of the action is shown, connected to the light it turns on. The lines in a trajectory will be animated from source to effect, thereby possibly spanning multiple surfaces. Device icons and labels will always be shown, even if they are not active (in which case they are displayed semi-transparently). Other labels (e.g. action labels) only become visible when the connecting line crosses them. Animations will slowly fade out after they have been completed.

Trajectories can also visualize multiple actions which are triggered in sequence. Fig. 1(b) shows a trajectory with two sequential actions. In this situation, touching the screen causes a movie to be played on this screen. The action of playing a movie will itself cause another action to be executed: one that dims the lights for a better viewing experience. Likewise, it is possible to visualize more complex rules that combine multiple sensors using boolean operators (e.g. AND, OR).

2.2 Overriding System Actions: The Cancel Feature

Fig. 2.2 shows a mockup of the cancel command in action. Since the cancel feature is voice-controlled, it is displayed as a microphone sensor icon. The only possible state is an invocation of the cancel feature when it recognizes the word "cancel", as indicated in the corresponding label. When an action is cancelled the microphone will turn and shoot at the icon corresponding to the effect of the action, resulting in a hole in this icon. The shooting animation might again span different surfaces to reach its target. This kind of visual feedback shows users in a playful way that the effect that the action had on the environment has been undone.

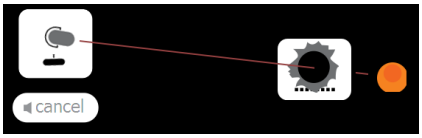


Fig. 2. When the action "light off" is cancelled, the microphone shoots a hole in the light icon.

2.3 Expressiveness and Limitations

The graphical notation was deliberately kept simple. It mainly targets systems that encode their behaviour as a list of if-then rules, which is a common approach to realizing context-awareness [5]. Our behaviour representation suffers from two main shortcomings. First, we are currently unable to visualize the reasoning behind machine learning algorithms, another frequently used approach to realize context-aware systems. Secondly, as with any visual language, scalability is an issue. When the notation would be used to visualize a very complex network of connected sensors and devices, the result could become too complex to comprehend for users. Despite these limitations, we believe that our notation is useful for exploring the potential of visualising the behaviour of AmI environments.

3 Implementation

We use several static and steerable projectors to overlay the physical environment with our graphical representation. The advantage of using projectors is that annotations can be displayed on any surface in the environment without requiring users to wear head-mounted displays or carry specialized devices. For more details on how to set up this kind of system, we refer to the existing literature (e.g. [8]). An overview of our implementation is given in Fig. 3.

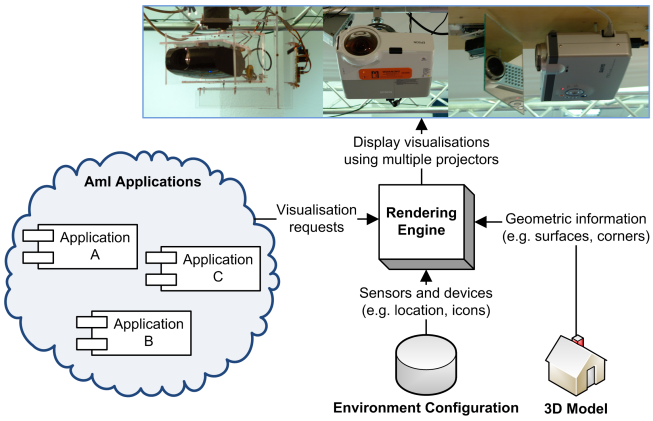


Fig. 3. Software applications in the AmI environment can send requests to the rendering engine to make their behaviour visible to end-users.

Because we were mainly interested in exploring the potential of our technique, our current implementation was deliberately kept simple. The most important software component in our system is the *rendering engine*. It is implemented as a central service that allows applications to make their behaviour visible to end-users. The rendering engine is responsible for overlaying the environment

with a visualisation of all devices and sensors, and for showing animations (or *trajectories*) between these elements when a software application executes an action. For this, it relies on a 3D model of the environment and an environment configuration file describing the sensors and devices in the environment. The 3D model of the environment is used to determine which of several steerable and static projectors need to be used and what image corrections need to be applied to display the annotations. The configuration file encodes the position of each device and sensor in the environment, together with their icons, possible states and a number of predefined trajectories. When software applications need to visualize a change of state in a device or the execution of a certain action, they send a request to the rendering engine containing an XML description of the required state change or trajectory.

4 Evaluation

4.1 Participants and Method

We ran an informal first-use study to investigate the suitability of our technique for understanding the behaviour of an AmI environment. Note that the aim of the study was to identify major usability problems and to drive design iteration, rather than to formally validate specific claims. The experiment was carried out in a realistic Ubicomp environment: an interactive room which features different kinds of sensors, and various means to provide users with information. We deployed a number of applications on the room's server which used sensors to steer other devices in the environment (e.g. motion detection with a webcam for controlling the lights). Applications were developed with Processing[1] and communicated with each other and the ambient projection system over the network.

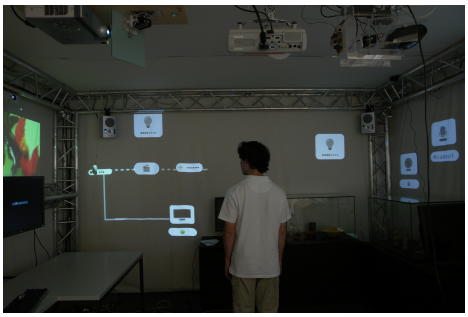


Fig. 4. A user looks at an ongoing animation.

The study group comprised 5 voluntary participants from our lab whose ages ranged from 24 to 31 (mean = 27.8); three were male, two female. All

[1] `http://www.processing.org/`

subjects had general experience with computers. Four out of five had experience in programming, while the fifth participant was a historian. Each individual study session lasted about 40 minutes. First, subjects were asked to read a document explaining our technique. Afterwards, subjects were presented with three situations in which they had to understand the environment's behaviour using our technique. After completing the test, participants had to fill out a post-test survey. The three tasks subjects had to perform during the study were:

- **Task 1**: Subjects were asked to press a play button on a touch screen, after which a movie would start to play in the environment. This, in turn, triggered an action that turned off the lights for better viewing experience.
- **Task 2**: Subjects were given the same instructions as in the first task, but were also told to find a way to turn the lights back on afterwards. They were expected to use the *cancel* functionality to achieve this effect, which was explained in the introductory document.
- **Task 3**: In the last task, subjects were asked to walk up to a display case and were told that they would notice something changing in the environment. The display case was equipped with a webcam for motion detection, which would turn the lights on or off depending on the user's presence.

Subjects were allowed to explore the system and perform each task several times until they felt that they had a good understanding of what was happening. After completing a task, participants received a blank page on which they had to explain how they thought that the different sensors and devices were connected. This allowed us to get an idea of each participant's mental model. Subjects were free to use drawings or prose (or a combination of both).

Two of the sensors used during the test were implemented using the Wizard of Oz technique: the voice-controlled cancel feature and the webcam motion detection sensor. The other applications and devices were fully functional.

4.2 Study Results

In our post-test survey, participants ranked our technique highly for being useful to understand and control what happens in an AmI environment (Q7, mean = 4.2, median = 4 on a 5-point Likert scale, $\sigma = 0.447$); and for *not* being confusing (Q8, mean = 4.2, median = 5, $\sigma = 1.095$). In general, participants indicated that they understood how to use our visualisation technique (Q1, mean = 4.6, median = 5, $\sigma = 0.548$); that they found the visualisation easy to understand (Q3, mean = 4, median = 4, $\sigma = 0.707$); and that it provided them with the information they wanted to know (Q4, mean = 4, median = 4, $\sigma = 1.0$). However, responses were less conclusive about the cancel feature (Q5 and Q6, $\sigma > 1.7$ in each case), where one participant (P5) gave the lowest score twice. Detailed results are presented in Fig. 4.2. Note that the small sample size ($n = 5$) causes the standard deviation (σ) to be relatively high overall. Four out of five participants described the system's behaviour correctly for each of the three tasks. The fifth participant (P5) described the first and third task correctly, but experienced difficulties with the second task.

Q1	I understand how to use the visualisation of the environment.
Q2	I found the visualisation of the environment easy to use.
Q3	The visualisation of the environment was easy to understand.
Q4	The visualisation of the environment was what I wanted to know.
Q5	I understand how to use the cancel feature.
Q6	I found the cancel feature easy to use.
Q7	I find these techniques useful to allow users to understand what happens in a smart environment, and to allow them to exert control over this behaviour.
Q8	I was not confused by the visualisation of the environment.

(a) Questions used in the survey.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
P1	5	5	4	5	5	5	5	5
P2	4	4	3	3	4	5	4	3
P3	5	4	5	3	5	5	4	5
P4	5	3	4	4	5	3	4	5
P5	4	3	4	5	1	1	4	3
Mean	4.6	3.8	4	4	4	3.8	4.2	4.2
Median	5	4	4	4	5	5	4	5
σ (Std. dev)	0.548	0.837	0.707	1.000	1.732	1.789	0.447	1.095

(b) Post-test questionnaire results. Participants are numbered from P1 to P5, questions from Q1 to Q8.

Fig. 5. Post-test questionnaire.

4.3 Discussion

Subjects were generally happy with our visualisations. One of the test participants mentioned that he found it "convenient to follow the lines to see what is happening", while another said: "it was clear to see which action had which effect". As mentioned before in Sect. 4.2, four out of five subjects were able to correctly describe the system's behaviour. We feel that this is a promising result, especially since the participant without a technical background (P2) was among these four. It might indicate that visualising the behaviour of an AmI environment can help users to form a correct mental model, which is in line with the findings of Rehman et al. [10]. However, further investigation is necessary to validate this claim.

The study also revealed a few shortcomings in our current prototype. Three subjects reported problems with recognizing the features of devices or sensors using its icons. Both the touch screen and cancel icons were found to be unclear.

During the study, we noticed that several participants experienced difficulties with keeping track of visualisations across multiple surfaces. Sometimes the visualisation would start outside subjects' field of view, which caused them to miss parts of the visualisation. A possible solution might be to use spatial audio to guide users' attention to the area of interest.

One participant (P2) commented that she sometimes received too much information, which confused her (as indicated by the neutral score on questions Q3, Q4 and Q8). She referred to the first task, in which a "click" on the touch

screen was visualised as causing the movie to start playing. It might be useful to disable visualisations for actions which occur often and are obvious to users, or to implement a generic filtering mechanism.

Finally, several subjects had difficulty in invoking the cancel feature. This issue might be ascribed to two causes: an unclear icon (as mentioned before) and the unfamiliarity of participants with speech interaction. One user (P4) mentioned that he felt "uneasy using a voice-controlled command", because "he was used to clicking". Both the relatively low score by participant P4 on question Q6; and the low scores on questions Q5 and Q6 by participant P5 and his incorrect explanation of the system's behaviour, might be attributed to the difficulty of invoking the cancel command. However, further studies will be necessary to identify the exact problems that subjects face when using the cancel feature.

5 Related Work

In recent years, increasing awareness of the difficulties users encounter in AmI or Ubicomp environments gave rise to a number of techniques that try to address these issues. In what follows, we discuss interaction techniques related to the ones presented in this paper. Rehman et al. [10] describe how a location-aware Ubicomp application was enhanced with augmented reality visualisations to provide users with real-time feedback. An initial user study compared the augmented version of the application with the original one. Results suggested that the visual feedback makes for a more pleasant user experience, and allows users to form a better mental model, which is in line with our findings. The main difference with our work is that the visualisations of Rehman et al. are application-specific, while ours could be used for any application. There have been a number of other studies that deal with issues of intelligibility, control and trust. For example, Antifakos et al. [1] found that displaying the system's confidence increases the user's trust, while Lim et al. [6] suggested that answering why (not) questions posed by users could improve the intelligibility of context-aware systems. We feel that these techniques could be combined with our approach. Further investigation will be necessary to determine the ideal level of user involvement and the most suitable feedback mechanisms in different situations.

We are not the first to visualise the behaviour of context-aware systems. iCAP [5], a design tool that allows end-users to prototype context-aware applications, also represents context-aware behaviour rules visually. With our system, however, users see a visualisation of the system's behaviour in real-time and in-situ, when and where the events take place.

6 Conclusions and Future Work

The implicit nature of interaction and the invisibility of the system in AmI and Ubicomp environments have led to a number of interaction challenges. In this paper, we presented a technique that overlays the environment with a graphical representation of its behaviour. This allows users to view the system state at

a glance and receive real-time feedback about events and actions that occur in the environment. Additionally, we provided users with a basic control feature that allowed them to cancel the last action. A small first-use study suggested that our visualisation might indeed improve understanding and support users in forming a reliable mental model. The study also revealed a few shortcomings of our system which we plan to address in a future design iteration. Finally, we are aware of the limitations of this study and plan to conduct further experiments to validate our findings.

References

1. Stavros Antifakos, Nicky Kern, Bernt Schiele, and Adrian Schwaninger. Towards improving trust in context-aware systems by displaying system confidence. In *Proc. MobileHCI '05*, pages 9–14. ACM, 2005.
2. Louise Barkhuus and Anind K. Dey. Is context-aware computing taking control away from the user? three levels of interactivity examined. In *Proc. Ubicomp '03*, pages 149–156. Springer, 2003.
3. Victoria Bellotti, Maribeth Back, W. Keith Edwards, Rebecca E. Grinter, Austin Henderson, and Cristina Lopes. Making sense of sensing systems: five questions for designers and researchers. In *Proc. CHI '02*, pages 415–422. ACM, 2002.
4. Victoria Bellotti and W. Keith Edwards. Intelligibility and accountability: human considerations in context-aware systems. *Hum.-Comput. Interact.*, 16(2):193–212, 2001.
5. Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. iCAP: Interactive Prototyping of Context-Aware Applications. In *Proc. Pervasive '06*, pages 254–271. Springer, 2006.
6. Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proc. CHI '09*. ACM, 2009.
7. Donald A. Norman. *The Design of Everyday Things*. Basic Books, September 2002.
8. Claudio S. Pinhanez. The everywhere displays projector: A device to create ubiquitous graphical interfaces. In *Proc. UbiComp '01*, pages 315–331. Springer-Verlag, 2001.
9. Kasim Rehman, Frank Stajano, and George Coulouris. Interfacing with the invisible computer. In *Proc. NordiCHI '02*, pages 213–216. ACM, 2002.
10. Kasim Rehman, Frank Stajano, and George Coulouris. Visually interactive location-aware computing. In *Proc. Ubicomp '05*, pages 177–194. Springer, 2005.
11. B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proc. WMCSA '94*, pages 85–90. IEEE Computer Society, 1994.
12. Norbert Streitz, Achilles Kameas, and Irene Mavrommati. *The Disappearing Computer: Interaction Design, System Infrastructures and Applications for Smart Environments*. Springer-Verlag, 2007.
13. Lucy A. Suchman. *Plans and situated actions: the problem of human-machine communication*. Cambridge University Press, 1987.
14. Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.