Improving Intelligibility and Control in Ubicomp

Jo Vermeulen
Hasselt University – tUL – IBBT
Wetenschapspark 2,
B-3590 Diepenbeek, Belgium
jo.vermeulen@uhasselt.be

ABSTRACT
Users often become frustrated when they are unable to understand and control a ubicomp environment. Previous work has suggested that ubicomp systems should be *intelligible* to allow users to understand how the system works and *controllable* to let users intervene when the system makes a mistake. In my thesis, I focus on novel user interfaces and interaction techniques to support intelligibility and control.

Author Keywords intelligibility; explanations; control; end-user configuration; ubicomp; feedback; feedforward.

ACM Classification Keywords H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms Human Factors; Algorithms; Design.

INTRODUCTION
Ubiquitous computing (ubicomp) systems are generally *context-aware*, which means they act based on *context* [8]: implicit input collected from the environment. These systems thus often act without explicitly involving the user, which may leave users surprised as to why the system behaves in a certain way. Moreover, system actions are usually a result of complex reasoning about context data which might be hard for users to grasp [11].

However, being difficult to understand is only part of the problem. Context-aware systems have been shown not be infallible. They are bound to sometimes make mistakes because of the inevitable incompleteness of context information [3,5]. It is therefore important that users are able to correct the system if it makes a mistake. Failing to do so will eventually result in users who feel out of control, and might result in them losing trust in the system [2].

Bellotti and Edwards [3] proposed two design principles to tackle these problems: *intelligibility* (what others have called *scrutability* [4]) and *control*. They argued that context-aware systems should be *intelligible* by informing users about the system's understanding of the world and should offer users *control* in order to recover from possible mistakes.

My thesis focuses on novel user interfaces and interactions for supporting intelligibility and control in ubicomp. This work encompasses two primary research questions:

1. How can we effectively allow users to understand how ubicomp systems work?
2. How can non-technical users be enabled to configure and correct the system's behavior?

In this thesis proposal, I describe both current and proposed work to further the theory, design, tools and techniques for effectively providing intelligibility and control to end-users.

RELATED WORK
There are a number of ubicomp systems and architectures that extend intelligibility and control to end-users. Cheverst's IOS system [4] shows confidence levels and visualizations of decision tree rules and allows end-users to manipulate system parameters. Situations [9] automatically supports simple intelligibility and control user interfaces and also allows designers to create application-specific user interfaces.

Lim, Dey and Avrahami [15] investigated if why (not) questions could be used to improve the intelligibility of context-aware systems. Their results suggest that allowing users to pose why (not) questions about the behavior of a context-aware system would result in better understanding and stronger feelings of trust. In a later study, Lim and Dey [14] investigated the different information needs users have for context-aware applications under various situations, recommending amongst others that why questions should be made available for all context-aware applications.

Others have explored providing intelligibility through visualizations. Rehman et al. [16] describe how a location-aware ubicomp application was enhanced with augmented reality visualisations. An initial user study suggested that the visual feedback supports users in forming a better mental model. A few studies have been performed to investigate the effect of displaying the level of uncertainty when systems make decisions based on context data [1,18]. However, further research is needed as these studies have reported contradicting results.

UbiComp'10, September 26–29, 2010, Copenhagen, Denmark.

Ju, Lee and Klemmer [13] describe a design framework for reasoning about transitions between implicit and explicit interaction. They discuss three interaction techniques that allow users to overcome errors in system's proactive behaviour: *user reflection*, *system demonstration*, and *override*. The first two can be seen as interaction techniques for improving intelligibility, while the latter is a technique for providing control.

Coutaz [6] proposed the *meta-User Interface* (meta-UI) concept, which is nothing more than a generic user interface to support intelligibility and control in smart spaces. Coutaz analyzed several existing systems and argues that there should be more attention towards control by end-users and to embedding meta-UIs within domain-specific applications.

Several systems have been developed to support end-users in controlling, configuring or programming their ubicomp environment (*e.g.*, [7,17]). Further investigation is necessary, however, to explore which interaction techniques and user interfaces are best suited for this purpose.

APPROACH AND METHODOLOGY

Many systems I discussed in the previous section only represent a single point in the larger design space of possible techniques to provide intelligibility and control. I have outlined a number of design decisions that I considered during my work:

Timing: Intelligibility and control can be supported at different times during the interaction: *before* an event will take place (*e.g.*, Ju, Lee and Klemmer's [13] system demonstration and override techniques), *during* the event (real-time) or *after* the event occurred (*e.g.*, why questions).

Generality: User interfaces and interaction techniques for intelligibility and control can be *general* (*e.g.*, why questions) or *domain-specific* (*e.g.*, techniques for visualizing location errors in navigation systems).

Degree of co-location: Coutaz [6] refers to this dimension as the level of integration. The intelligibility or control technique might be *embedded* or integrated with the rest of the system (*e.g*, Ju, Lee and Klemmer's [13] techniques) versus *external*, when they require users to switch to a separate mode.

Initiative: Users may need to explicitly request intelligibility information or invoke control techniques manually (*user*), or might automatically be presented with these features when necessary (*system*: *e.g.*, Ju, Lee and Klemmer's [13] techniques).

Modality: Several modalities can be used to help users to understand or control the system (e.g. *visual*, *auditory*, *haptic*). Depending on the domain, context and the system, different modalities might be preferred.

Level of control: The level of control end-users can exert over the system varies from *intelligibility*, where no additional control is added beyond intelligibility[1], over *counteract*, where users can perform the opposite action (*e.g.*, undo), to *configuration,* where users can tweak predefined system parameters, and *programmability* where users can themselves (re-)define how the system works.

Even though this design space is non-exhaustive, it provides valuable insights into different areas that are still left to be explored, both in the literature and in my own work. I currently developed two prototypes at different positions in the spectrum. Additionally, I investigated different ways to support *feedforward*, a specific type of intelligibility information which is presented *before* an event takes place.

In future work, I will extend the current prototypes and build new ones to further explore the design space. I will continue to study the different dimensions and perform larger user studies to validate my results. I am aware of the potential challenges and pitfalls in building these prototypes and will rely on existing tools and approaches for rapid prototyping (*e.g.*, Wizard of Oz testing) if necessary.

CURRENT RESULTS

PervasiveCrystal

As a first step, I extended the existing ubicomp framework ReWiRe [19] to allow systems built with ReWiRe to reason about the causes and consequences of system and user actions. As ReWiRe employs a simple rule-based behavior model it is a good candidate for exploring different interaction techniques for intelligibility and control.

I annotated ReWiRe's behavior model with additional metadata to link different rules together. These annotations are then processed at runtime to build up a model of the system's behavior that can be easily queried. Additionally, the behavior model provides simple control mechanisms over executed actions (*e.g.*, undo). Based on this extension, I built *PervasiveCrystal* [22], a system that answers *why* and *why not* questions posed by users about unexpected events that occurred and expected events that did not occur respectively (see Figure 1).

A pilot study was conducted with a first iteration of PervasiveCrystal to get an idea of its ease of use. Five volunteers were asked to use this system on a mobile device to understand and control the behavior of an interactive room in different situations. All subjects were able to use the questions to find the cause of events. One of the major problems users faced was the fact that the why-menu quickly became cluttered when many events were firing in a short time span (*e.g.*, events from a motion sensor). Even though the majority of participants was happy with the fine-grained control user interfaces (see Figure 1-B.4), subjects found it difficult to predict the result of invoking the *undo*

[1] This still allows users to intervene by changing their own behavior based on their understanding of how the system works.

and *do* commands. We later changed our implementation to use more specific labels based on the label for the corresponding action (*e.g.*, "Turn lights on"), as shown in Figure 1-B.

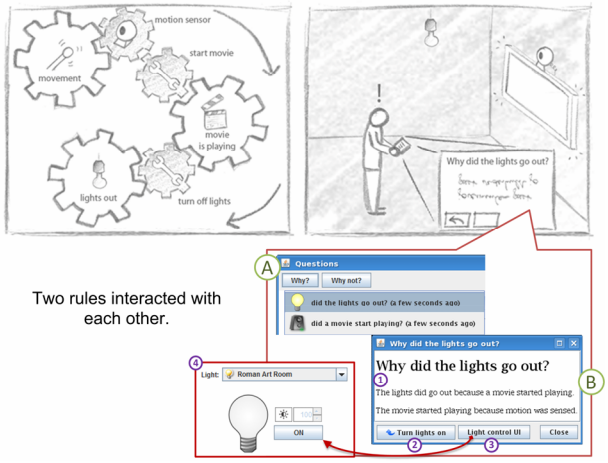

Figure 1. PervasiveCrystal shows a list of available questions, based on recent events (*A*). Answers are generated by linking events to what caused them to happen (*B.1*). Additionally users have access to two control mechanisms: they can *undo* operations (*B.2*) or invoke fine-grained control user interfaces, in this case: a light control UI (*B.4*).

A visual representation of system behavior

Because of the heterogeneous nature of ubicomp environments — which often employ several displays, speakers, sensors, embedded devices — users might require co-located information that tells them what the system is doing, *when* and *where* it is doing this, and allows them to intervene without leaving their current task.

To explore this idea, I developed a prototype [21] that uses steerable projectors to overlay the environment with real-time visualizations of actions occurring in the environment (*e.g.* turning off the lights)[2]. Figure 2 shows how we used a simple graphical language to visualize the relationships between sensors or devices and system actions. When an action is executed, an animation is shown to visualize the cause(s) and consequence(s) of this action. In addition, users can issue a voice command to cancel (or *undo*) the most recent action.

We ran an informal study with five participants to gather feedback about the suitability of this approach. Subjects were asked to explain how the system worked in three different situations. Four out of five subjects described the system's behavior correctly for each of the three situations. Participants were generally happy with our visualizations, but sometimes had difficulties with keeping track of visualizations across multiple surfaces. One participant mentioned she received too much information, leaving her overwhelmed. This might indicate that it we should be careful when automatically providing explanations (the *initiative* dimension). Finally, several subjects experienced difficulties with invoking the cancel feature, possibly because they were not familiar with speech interaction.

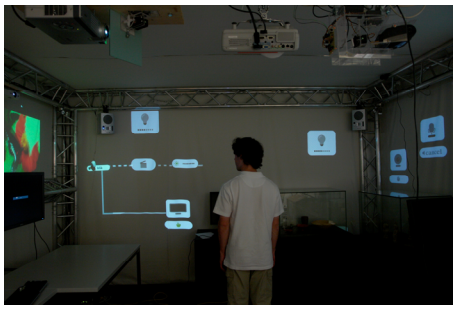

Figure 2. A user looks at an animation that links sensors and devices with system actions to explain the system's behavior.

Feedforward

Feedback and affordances are two of the most well-known principles in human-computer interaction, and considered to be essential ingredients of good user interface design. The related notion of *feedforward* [10], however, does not seem to have been given as much consideration. Feedforward informs the user about *what the result of an action will be*. If we consider the timing dimension, feedforward is thus intelligibility information that is provided *before* an event takes place.

Feedforward might be conveyed in several ways: through form-giving (*e.g.*, when using physical objects in combination with digital information), using different modalities (*e.g.*, auditory or haptic cues), ranging from a minimal hint such as a label to the full details of what might happen.

Recently, I have investigated [20] the use of feedforward in the field of tangible interaction. The specific characteristics of tangible interaction [12] (*e.g.*, little reliance on displays) make it an ideal area for exploring rich ways to support feedforward (and intelligibility in general). I identified several promising directions for future work on feedforward. Surprisingly, designers mostly rely on the visual modality and, to a lesser degree, on the haptic modality. This provides opportunities for auditory feedforward, which is only seldomly used. Second, we noticed that very few systems make feedforward continuously available for each step of the interaction process. Continuous feedforward might be important for context-aware systems to guide the user (*e.g.*, by making *what if* questions [15] available throughout the system). We hope to use the results from this survey on feedforward to refine our design space and extract opportunities for feedforward in the general field of ubiquitous computing.

[2] A video of the system is available at http://www.youtube.com/watch?v=gztUqtvvMHs

NEXT STEPS

Figure 3 shows how my current work fits into the proposed design space for intelligibility and control. In the next months, I will continue to study the dimensions I touched upon and perform larger user studies to confirm my initial results. In addition, I would like to delve deeper into the dimensions *generality*, *timing* and *level of control* by improving upon current prototypes and building new ones.

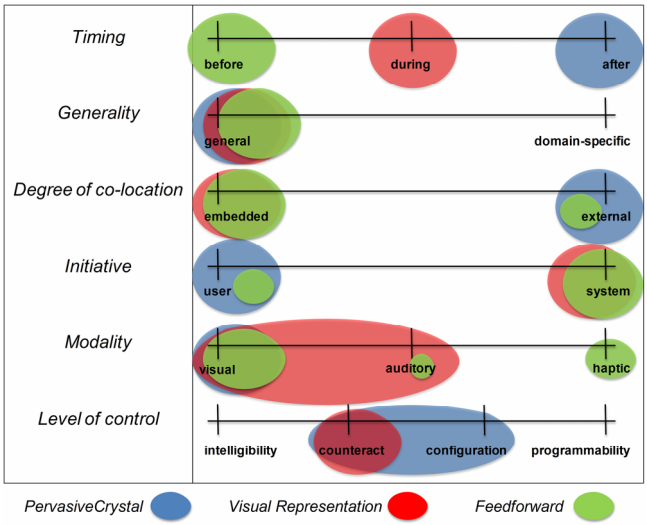

Figure 3. Framing my current work into the proposed design space for intelligibility and control.

More specifically, I plan to investigate how specific application domains (*e.g.*, pervasive healthcare, mobile guides) influence the suitability of different techniques. I hope to build a prototype that spans the full *timing* scope, providing intelligibility and control before, during and after events, applying the insights I gained from the feedforward survey. Finally, I am looking into allowing end-users to configure and program their environments in real-time.

EXPECTED OUTCOME

With my thesis, I aim to develop guidelines into effectively supporting intelligibility and control which future ubicomp systems can rely on, and develop novel user interfaces, tools and interaction techniques for intelligibility and control that are applicable in a wide range of different situations.

ACKNOWLEDGMENTS

I would like to thank my advisors Karin Coninx and Kris Luyten for their support and guidance.

REFERENCES

1. Antifakos, S., Schwaninger, A., and Schiele, B. Evaluating the Effects of Displaying Uncertainty in Context-Aware Applications. *Proc. Ubicomp '04.*, (2004), 54-69.
2. Barkhuus, L. and Dey, A.K. Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined. *Proc. Ubicomp '03*, Springer (2003), 149–156.
3. Bellotti, V. and Edwards, W.K. Intelligibility and accountability: human considerations in context-aware systems. *Hum.-Comput. Interact. 16*, 2 (2001), 193–212.
4. Cheverst, K., Byun, H.E., Fitton, D., Sas, C., Kray, C., and Villar, N. Exploring Issues of User Model Transparency and Proactive Behaviour in an Office Environment Control System. *User Modeling and User-Adapted Interaction 15*, 3-4 (2005), 235–273.
5. Cheverst, K., Davies, N., Mitchell, K., and Efstratiou, C. Using Context as a Crystal Ball: Rewards and Pitfalls. *Personal Ubiquitous Comput. 5*, 1 (2001), 8–11.
6. Coutaz, J. Meta-User Interfaces for Ambient Spaces. *Proc. TAMODIA '06*, (2006), 1-15.
7. Dey, A.K., Hamid, R., Beckmann, C., Li, I., and Hsu, D. a CAPpella: programming by demonstration of context-aware applications. *Proc. CHI '04*, ACM (2004), 33–40.
8. Dey, A.K. Understanding and Using Context. *Personal Ubiquitous Comput. 5*, 1 (2001), 4–7.
9. Dey, A.K. and Newberger, A. Support for Context Intelligibility and Control. *Proc. CHI '09*, ACM (2009).
10. Djajadiningrat, T., Overbeeke, K., and Wensveen, S. But how, Donald, tell us how?: on the creation of meaning in interaction design through feedforward and inherent feedback. *Proc. DIS '02*, ACM (2002), 285-291.
11. Edwards, W.K. and Grinter, R.E. At Home with Ubiquitous Computing: Seven Challenges. *Proc. UbiComp '01*, Springer-Verlag (2001), 256–272.
12. Ishii, H. and Ullmer, B. Tangible bits: towards seamless interfaces between people, bits and atoms. *Proc. CHI '97*, ACM (1997), 234-241.
13. Ju, Lee, and Klemmer. Range: exploring implicit interaction through electronic whiteboard design. *Proc. CSCW '08*, ACM (2008), 17–26.
14. Lim, B.Y. and Dey, A.K. Assessing Demand for Intelligibility in Context-Aware Applications. *Proc. Ubicomp '09*, ACM (2009), 195-204.
15. Lim, B.Y., Dey, A.K., and Avrahami, D. Why and Why Not Explanations Improve the Intelligibility of Context-Aware Intelligent Systems. *Proc. CHI '09*, ACM (2009), 2119–2128.
16. Rehman, K., Stajano, F., and Coulouris, G. Visually Interactive Location-Aware Computing. *Proc. Ubicomp '05.*, (2005), 177-194.
17. Rodden, T., Crabtree, A., Hemmings, T., et al. Configuring the Ubiquitous Home. *Proc. COOP '04*, (2004), 227-242.
18. Rukzio, E., Hamard, J., Noda, C., and De Luca, A. Visualization of uncertainty in context aware mobile applications. *Proc. MobileHCI '06*, ACM (2006), 247–250.
19. Vanderhulst, G., Luyten, K., and Coninx, K. ReWiRe: Creating interactive pervasive systems that cope with changing environments by rewiring. *Proc. IE '08*, (2008), 1-8.
20. Vermeulen, J., and Hoven, E. van den. Feedforward in tangible interaction. *Unpublished*.
21. Vermeulen, J., Slenders, J., Luyten, K., and Coninx, K. I Bet You Look Good on the Wall: Making the Invisible Computer Visible. *Proc. AmI '09*, Springer-Verlag (2009), 196–205.
22. Vermeulen, J., Vanderhulst, G., Luyten, K., and Coninx, K. PervasiveCrystal: Asking and Answering Why and Why Not Questions about Pervasive Computing Applications. *Proc. IE '10*, (2010).