Answering Why and Why Not Questions in Ubiquitous Computing

Peer-reviewed author version

VERMEULEN, Jo; VANDERHULST, Geert; LUYTEN, Kris & CONINX, Karin (2009) Answering Why and Why Not Questions in Ubiquitous Computing. In: 11th International Conference on Ubiquitous Computing (Ubicomp 2009). p. 210-213..

Handle: http://hdl.handle.net/1942/10369

Answering Why and Why Not Questions in Ubiquitous Computing

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

Karin Coninx
Hasselt University – tUL – IBBT
Expertise Centre for Digital Media
Wetenschapspark 2,
B-3590 Diepenbeek, Belgium
karin.coninx@uhasselt.be

Geert Vanderhulst
Hasselt University – tUL – IBBT
Expertise Centre for Digital Media
Wetenschapspark 2,
B-3590 Diepenbeek, Belgium
geert.vanderhulst@uhasselt.be

Kris Luyten
Hasselt University – tUL – IBBT
Expertise Centre for Digital Media
Wetenschapspark 2,
B-3590 Diepenbeek, Belgium
kris.luyten@uhasselt.be

UbiComp 2009, Sep 30 – Oct 3, 2009, Orlando, FL, USA

Abstract
Users often find it hard to understand and control the behavior of a Ubicomp system. This gives rise to usability problems and can lead to loss of user trust, which may hamper the acceptance of these systems. We are extending an existing Ubicomp framework to allow users to pose *why* and *why not* questions about its behavior. Initial experiments suggest that these questions are easy to use and could help users in understanding how Ubicomp systems work.

Keywords
Why, questions, explanations, intelligibility, context.

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

Introduction
The combination of implicit input and complex decision making algorithms poses difficulties for users in understanding how a ubiquitous computing (Ubicomp) system works [1]. Previous studies have demonstrated the potential of allowing users to ask *why* and *why not* questions about the behavior of complex [2] and context-aware software [3]. To the best of the authors' knowledge, however, there is to date no Ubicomp

Figure 1. The scenario: a visitor of an interactive museum notices that a documentary stops playing on a nearby display, and poses a why question to figure out why this happened.

Figure 2. The why menu, the timeline view and the answer to the question in the scenario. The why menu shows a sorted list of questions relevant to end-users (most recent first), while the timeline shows all events (including system events).

framework available that supports these questions. This paper describes our ongoing effort of extending the ReWiRe framework [4] with support for why and why not questions, arising respectively from *unexpected* events that occurred or *expected* events that did *not* occur. Our system mainly differs from existing implementations such as Crystal [2] in that it can trace events and actions across distributed applications and supports questions about the interplay of these networked components. A first experiment with our prototype suggested that these questions are easy to use and can help in understanding the behavior of a Ubicomp system.

Scenario

Consider a user walking around in an interactive museum (Figure 1). The user carries a personal device (e.g. an Ultra-Mobile) for easy access to the services in the museum and for asking why (not) questions. When she walks up to a display case containing Roman pottery, a related documentary movie starts playing on a nearby screen. When she moves a few steps backwards to get a better look at the screen, the movie suddenly stops. She is surprised by this behavior, and decides to use the why questions to find out why this happened. As shown in Figure 2, the why menu shows her a list of questions she can ask. She selects the first question and learns that the movie stopped because she was too far away from the display case. Having understood how the system works, she then moves a bit closer to get the movie to play again.

Implementation

Generating Questions and Answers

We built our implementation of why (not) questions on top of the existing ReWiRe framework [4]. Context-aware behavior in ReWiRe is defined by means of Event-Condition-Action (ECA) rules. To present users with possible why (not) questions and to generate corresponding answers to these questions, we have annotated these ECA rules in three ways. First, events that affect end-users are annotated with a *why* label for generating the questions in the why menu. Secondly, actions need to provide a list of events they can possibly trigger, in order to create a list of why *not* questions about events that did not occur when these actions were not executed. Finally, each event, action and condition is provided with a short descriptive label. These short labels are used together with the available information in the ECA rules (e.g. causal dependencies) to automatically generate answers to the questions.

Allowing Users to Intervene

Our system provides basic control mechanisms that allow users to correct unwanted behavior. Users might want to reverse an effect — or *undo* — when asking a why question (as they did not expect it to happen). Similarly, users may want to achieve an effect — or *do* — when asking a why not question (as they expected the effect to occur while it did not). Answer dialogs therefore include an *undo* or *do* button depending on the type of question, as seen in Figure 2. Do is implemented by simply executing the action part of the corresponding ECA rule that would have achieved the desired effect, but which was not executed. Undo is realized by supplying each rule with an inverse action that reverses the effects of the rule's action. When the user undoes an effect, the system will execute the inverse action of the corresponding rule which caused the effect to happen. While our implementation of undo and do is relatively straightforward, we believe it is

Figure 3. The control UI for the "*Play media*" task. This UI is shown when users click the "*How can I play media*?" button in the answer dialog of Figure 2. It provides fine-grained control over playing of music or video in the environment. Similar UIs exist for other tasks (e.g. controlling lighting).

sufficient for getting a sense of the potential of these control mechanisms in a first user experiment.

Besides these basic control operations, users can also invoke a specific user interface (UI) related to the question they asked. This UI provides more fine-grained control over the system and is activated by clicking on an "*How can I … ?"* button in the explanation dialog. To make this possible, actions are annotated with related user tasks (e.g. playing media, controlling lighting, etc.). Each of these user tasks has an associated control UI which can be invoked at any time. In the answer dialog of Figure 2, for example, clicking the "*How can I play media?*" button will result in the control UI of Figure 3.

Preliminary User Study

We conducted a pilot study to get an idea of the ease of use of our prototype. Five voluntary researchers from our lab were asked to use our techniques to understand and control the behavior of an interactive Ubicomp room in three situations. Participants used a networked Ultra-Mobile PC (UMPC) to ask why questions and view the corresponding answers. All subjects were able use the questions to find the cause of events in these tasks. After the test, we conducted a semi-structured interview in which participants generally indicated that they found our technique useful and easy to use.

One of the main problems users faced is that the why-menu quickly became cluttered when many events were firing in a short time span. This made it hard for subjects to find the question they wanted to ask. Participants also found it hard to predict the effect of invoking undo and do, which might suggest that more specific labels (e.g. "Play video") are needed. On a positive note, the majority of participants were able to use the specific control UIs (see Figure 3).

Ongoing and Future Work

Our immediate ongoing work consists of improving our prototype based on feedback from the informal study. At a later stage, we plan to conduct a more formal evaluation and investigate the required developer effort to make existing ReWiRe applications "why question"-ready.

Acknowledgements

The authors would like to thank the study participants for their time and other researchers at our lab for their feedback and advice on this work. Part of the research at EDM is funded by ERDF (European Regional Development Fund), the Flemish Government and the Flemish Interdisciplinary Institute for BroadBand Technology (IBBT).

References

[1] V. Bellotti and W.K. Edwards, "Intelligibility and accountability: human considerations in context-aware systems," *Hum.-Comput. Interact.*, vol. 16, 2001, pp. 193-212.

[2] Myers, Weitzman, Ko, and Chau, "Answering why and why not questions in user interfaces," *Proc. CHI '06*, ACM, 2006, pp. 397-406.

[3] B.Y. Lim, A.K. Dey, and D. Avrahami, "Why and Why Not Explanations Improve the Intelligibility of Context-Aware Intelligent Systems," *Proc. CHI '09*, ACM, 2009, pp. 2119-2128.

[4] G. Vanderhulst, K. Luyten, and K. Coninx, "ReWiRe: Creating interactive pervasive systems that cope with changing environments by rewiring," *Proc. IE '08*, IET, 2008, pp. 1-8.