Bridging the Gap between Users and Smart Products

Marcus Ständer
Telecooperation Department
Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt, Germany
Email: staender@tk.informatik.tu-darmstadt.de

***Abstract*—The increasing number of interactive products in our environment leads to an accession of interaction between products and users. Nowadays, this interaction is described with different concepts than the interaction between the products. This makes it difficult to mutually replace users and products to create smart environments that are able to automatically perform tasks for the user, if suitable products are available. To advance the design of interactive smart environments and to bridge this gap, we introduce a concept for describing product-initiated interaction with users.**

I. Introduction

In our everyday life we encounter an increasing number of smart products. A smart product is a physical product, software or service with capabilities to communicate with other smart products and with users. Several smart products together form a smart environment. A more detailed definition can be found in [1].

In the vision of ubiquitous computing [2], users naturally interact with these environments and the vast number of smart products is used to ease everyday work. The products can dynamically find other products in the environment to fulfill complex procedures together. In the vision users and products are interchangeable: A user can take over the work of a smart product and vice versa.

The question how the behavior of smart environments can be handled has been in the focus of research for some time now. The three major aspects thereby are (i) how to solve the technical interplay of products, (ii) how the model for the behavior has to be designed, and (iii) how to design interaction with the user. Unfortunately there seems to be a gap between the different research directions and only a few meta-concepts exist. In this short paper, we introduce the first element of our approach by proposing a categorization of product-initiated interaction with the user. We model the processes in the environment as workflows and use these categories to annotate the different steps. These annotations enable the smart product to figure out at runtime, how to approach the user in a suitable way.

II. Abstracting Interaction

Users expect to interact naturally with their environment which also holds for product-initiated interaction. Obviously a warning must be communicated differently than just some phrase, which the user even might decide to ignore. If a product does not adapt, the user would instantly get the feeling of a very static and "dumb" environment. Another issue to be considered is the explicit or implicit character of the user's feedback. Whenever a product tells a user to take a certain action, the user could press some button on a user interface, providing explicit feedback, or sensors could generate events and provide implicit feedback. Thus, it is a very challenging task to design the interaction behavior of smart products.

A. Representation

Based on concepts derived from *speech act theory* [3] *communicative acts* [4], and the analysis of different scenarios provided by the research project *SmartProducts*, we created a set of interaction types describing the interaction between smart products and users. We thereby focus on practical product-initiated computer to human interaction in smart environments not only from the view of the end-user but also from the view of developers. The different types can be arranged according to the *importance* of the interaction for the user and the expected *feedback*, as shown in Fig. 1. Concerning the importance, we distinguish between interaction that *can be omitted*, that *can be deferred* and that *cannot be deferred*. The expected feedback can be split up into *no expected feedback*, *simple predicate feedback* (yes/no) and *complex feedback*. To enhance the adaptation during runtime, we also include an optional field in each type, describing if the user will be occupied for a rather long time or not. The resulting interaction types can then be used to determine when and how to interact with the user in the most suitable way. Further, it might be necessary to allow changing the type during runtime. The subject of a notification, which is disregarded for too long, might become a critical issue after some time.

In the following, we list the different types we identified so far and give a brief description for each of them. We illustrate each type with the example of our smart coffee maker [5].

Phrase *Phrases* only convey information of low importance which can also be easily ignored, like greetings. Example: "*Thank you for descaling the coffee maker.*'

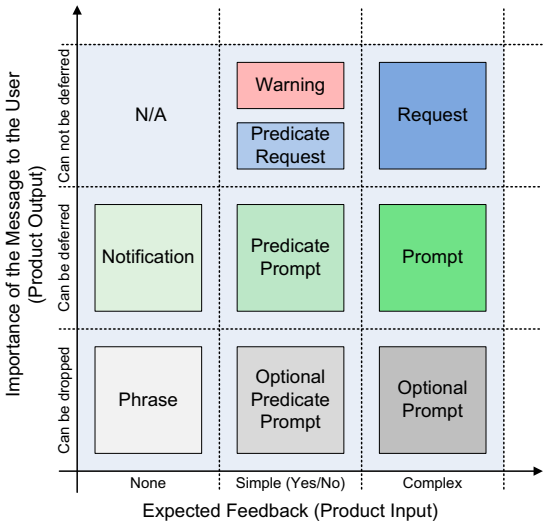

Figure 1: Overview of the interaction types.

Optional Predicate Prompt *Optional Predicate Prompts* can be used to get yes/no values. If no response is recognized, a predefined default value will be used.
Example: "*Do you want two pieces of sugar as usual? (Default: yes)*"

Optional Prompt *Optional Prompts* allow more complex feedback. They also contain default values that are used, if no user feedback is received for some time.
Example: "*Which web site should be loaded while you are waiting for your coffee? (Default: smartproducts-project.eu)*"

Notification The content of *Notifications* is related to all general information available and can be deferred for some time.
Example: "*The approximated time for running the descaling program is 20 minutes.*"

Predicate Prompt *Predicate Prompts* are deferrable and expect simple yes/no feedback.
Example: "*Shall the descaling be started now?*"

Prompt *Prompts* can be used to ask for more complex feedback. This type is also deferrable.
Example: "*Descaling is required. When should it be planned?*"

Predicate Request *Predicate Requests* describe interaction where a smart product asks for non deferrable, simple feedback.
Example: "*To continue, please place a water pot under the water dispenser.*"

Request Like the *Predicate Request*, the *Requests* require immediate feedback. The expected feedback can be more complex.
Example: "*To continue, please fill 100ml of the descaling solution into the water tank.*"

Warning It is crucial that *Warnings* are recognized by the user as soon as possible and thus cannot be delayed.
Example: "*The coffee maker has a critical malfunction. Please turn it off and call a service technician.*"

III. Towards Interactionflows

One way to make smart products "*smart*" is to enable them to fulfill complex procedures together with other smart products in the environment and with the users. Developers of smart products must be provided with adequate tools to model such procedures. There are plenty of approaches defining how this can be done, ranging from task models to process definition [6].

In the currently started work we will develop a concept that includes context-aware interaction descriptions into workflows. Using the interaction types we will reduce the description required for the interaction with the user. Including possible feedback into these types will allow us to track the current progress of workflows more accurately and flexible. The system should for example recognize and allow if the user does not follow the modeled workflow exactly. This advances the otherwise rather static characteristics of workflows. With these annotations, we enable developers of smart products to create workflows which result in a natural interaction at runtime.

Acknowledgments

My thanks go to my supervisor Professor *Max Mühlhäuser* for his ideas and his constructive feedback.

Part of this research was conducted within the *SmartProducts* project funded as part of the Seventh Framework Programme of the EU (grant number 231204).

References

[1] Mühlhäuser, M.: Smart products: An introduction. In: Constructing Ambient Intelligence. (2008) 158–164

[2] Weiser, M.: The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. **3**(3) (1999) 3–11

[3] Austin, J.L.: How to do things with words. Harvard University Press (2000)

[4] FIPA, T.: Fipa communicative act library specification. Change 2000 (2000)

[5] Aitenbichler, E., Lyardet, F., Austaller, G., Kangasharju, J., Mühlhäuser, M.: Engineering intuitive and self-explanatory smart products. In: Proceedings of the 2007 ACM symposium on Applied computing, Seoul, Korea, ACM (2007) 1632–1637

[6] Limbourg, C. and Pribeanu, C. and Vanderdonckt, J.: Towards uniformed task models in a model-based approach Lecture notes in computer science, 164182. Springer