Towards Interactionflows for Smart Products

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

ABSTRACT

Nowadays, the interaction between a product and the user is described using different methods than for product to product communication. This makes it difficult to replace users and products mutually to create really dynamical environments, capable of reducing the amount of interactions, if possible. To advance the design of interactive smart environments, we introduce a concept for describing product-initiated interaction with users and demonstrate, how they can be applied in practice. This allows to combine both, automated procedures and interaction with the user, to a new concept called interactionflows.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Systems; D.2.11 [**Software Engineering**]: Software Architectures

Keywords

Smart Products, Interactionflows, Interaction Types, Workflows, Context

1. INTRODUCTION

In our everyday environment we encounter more and more smart products. A smart product is a physical product with capabilities to communicate with other smart products and with users [4]. Several smart products together form a smart environment.

One way to make smart products "smart" is to enable them to fulfill complex procedures together with other smart products in the environment and the users. Developers of smart products must be provided with adequate tools to model such procedures. There are plenty of approaches, how this can be done, started from using task models like GOMS [2] or CTT [5] to modeling with process definition languages like BPEL [3] or XPDL [7].

In all existing approaches, developers need to take care of three aspects: (i) they need to be able to describe context-aware procedures, (ii) those procedures need to be able to include the local product and remote smart products, and finally (iii) they need to be able to specify how to approach the user. However, existing approaches only pay attention to one or two of those aspects.

In this paper, we address the question, how the three aspects can be handled in an integrated way and give a brief summary of our categorization of product-initiated interaction. Thereby we focus on a high level description of user interaction, which is suitable for being included into such procedures. Further we show our system design and how we use these categories, context information and a common process definition language to describe procedures including different products and users.

SAC'10 March 22-26, 2010, Sierre, Switzerland.

2. ABSTRACTING INTERACTION

Users expect to interact naturally with their smart environment and whenever a product needs to approach her, this must also happen in a very natural way.

Based on the speech act theory and communicative acts, we derived a set of interaction types between smart products and users. While the basic theory tries to explain psychological meanings of speech acts, we focus on practical product-initiated computer to human interaction in smart environments. We differ between *warnings*, *phrases*, *notifications*, *advices*, *acknowledgements*, *prompts*, *chats* and *commands*. More details about the different types can be found in [6], here we illustrate only two types exemplary.

(i) One of the most important interaction types is the *warning*. If the user does not recognize a possibly critical situation this might lead to dangerous situations. Thus, a Warning has to be recognized by the user as soon as possible. Example: "*The coffee maker has a critical hardware malfunction. Please remove its power cord and call a service technician.*"

(ii) A *phrase* describes the kind of interaction that has no remarkable content like greetings or gratitudes. They are mainly used to make the interaction more natural and let systems appear more friendly. Since it is only of low importance to the user, there is no need to ensure that she noticed it. Example: "*Thank you for descaling the coffee maker.*"

Each of these types has different implications on how the interaction with the user should take place during runtime. Obviously a warning must be communicated in a different way than a phrase, which the user even might decide to ignore.

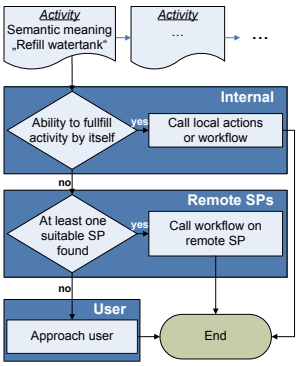

Figure 1: Flowchart of activity handling

3. TOWARDS INTERACTIONFLOWS

These interaction types form a key building block for our architecture. In the currently started work we develop a concept that includes context awareness, product orchestration and interaction with the user into workflows and call this approach *interactionflows*.

XPDL [7] is well established and supports many features we require, like defining activities, participants, resources or the usage of applications, while still being simple to use. It is well supported by graphical workflow editors, enabling product developers to easily create workflows for our system. Due to these facts, we use the process definition language XPDL to model procedures as workflows.

3.1 Including Context

Using XPDL conform workflows allows using logical expressions to check if a transition should be triggered. We base our system on XML representations of context information, like it is sent by the communication middleware MundoCore [1] developed at our department. Applying usual XPath expressions to those context representations, we select the appropriate values, which are compared to given values. Thus, we are for example able to check if the value of a temperature exceeds a certain limit. By writing the details about the comparison directly into the workflow description, we are able to include all required data.

3.2 Including Remote Products

Figure 1 shows a flowchart example demonstrating how the system handles an activity. If a product is not capable of executing an activity by itself, it first tries to identify other smart products in the environment which can provide a workflow that performs the task. Therefore we add additional data to the activity, allowing to create a product choreography. Beneath a semantic description of the task, we also provide a mapping for data exchange, using standard constructs of XPDL. If the product has discovered several suitable products, it hast to choose the best one. The product does this by evaluating additional information about the other products, like their location or currently running workflows.

3.3 Including Users

If the product cannot find another product that matches the requirements, it has to approach the user, so he can take over that part. Therefore the product searches the environment for products that provide output capabilities, like screens or speakers. As before, it needs to choose the most suitable device by evaluating which device fits best to the reported context information like location of the user and the devices, brightness or noise level in the room. The result is compared with the last used output device to decide if it is worth changing.

The interaction types can be included into the workflows as extended attributes, containing details about the content of the interaction. To improve modeling, we think that the number of participants of an average workflows should be reduced to two: the product itself and everything / everyone else. Thus, the activities running on the local product, having detailed execution information, can be separated from the activities that have to be executed by other products or the user.

4. SUMMARY AND OUTLOOK

Coming back to the three aspects from the introduction, we plan to (i) allow triggering the workflows with events, (ii) describe procedures using the process description language XPDL with some extensions to allow including remote products and (iii) include abstract definitions of interaction into those workflows.

The interaction types can be used as meta information for the activities a product has to execute. If the activity cannot be executed automatically, the type can be used to provide interaction suitable to the situation. However, if the user is asked to "*refill the watertank*", the feedback can be a spoken phrase ("*ok*", "*next step*", ...), a button pushed on a provided user interface, or it can even be automatically provided by a water sensor. The problem of incorporating different kinds of feedback will be addressed in our future work.

Acknowledgments.

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

5. REFERENCES

[1] E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser. MundoCore: a light-weight infrastructure for pervasive computing. *Pervasive Mob. Comput.*, 3(4):332–361, 2007.

[2] S. K. Card, T. P. Moran, and A. Newell. *The psychology of human-computer interaction*. Erlbaum, Hillsdale, N.J. [u.a.], 1983.

[3] IBM, B. Systems, Microsoft, SAP, and Siebel. Web services business process execution language (WS-BPEL, BPEL), version 2.0 specification, 2007.

[4] M. Mühlhäuser. Smart products: An introduction. In *Constructing Ambient Intelligence*, pages 158–164. 2008.

[5] F. Paternò, C. Mancini, and S. Meniconi. ConcurTaskTrees: a diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 362–369. Chapman \ Hall, Ltd., 1997.

[6] M. Ständer. Bridging the Gap between Users and Smart Products. 2009.

[7] W. M. C. WfMC. XML Process Definition Language (XPDL), 2009.