Gathering knowledge for supporting interaction with smart products

Melanie Hartmann
Technische Universität Darmstadt
Hochschulstr. 10
64289 Darmstadt, Germany
+49 6151 164752
melanie@tk.informatik.tu-darmstadt.de

Victoria Uren
University of Sheffield
Department of Computer Science,
211 Portobello,
Sheffield, S1 4DP, UK
v.uren@dcs.shef.ac.uk

Elena Vildjiounaite
Technical Research Centre of Finland
Kaitoväylä 1
90571 Oulu, Finland
+358 20 722 2470
elena.vildjiounaite@vtt.fi

ABSTRACT
The number of smart products in our daily life and thus the interaction with them is constantly increasing. In order to facilitate the interaction, smart products require different types of knowledge: knowledge about the user model, workflows, the domain and the context. In this paper, we point out how these different types of knowledge can enhance the interaction and how they can be gathered.

Keywords
Smart products, knowledge acquisition, interaction, ontologies

1. INTRODUCTION
In our everyday life we encounter an increasing number of smart products. Thereby, we refer to smart products as physical products that are able to communicate with users and other smart products in the environment without relying on a central infrastructure. Smart products assist and guide the user in executing her tasks. For that purpose, smart products require different types of knowledge that are stored in their digital object memories. In this paper, we present the different knowledge types that we consider essential and how this knowledge can be gathered. To the best of our knowledge, we are the first to focus thereby on the interaction aspect of smart products. Related approaches focus on the data gathering and sharing [1] or on supporting the interoperability between smart products [2] without directly exploiting the knowledge for enhancing the interaction.

2. SMART PRODUCTS' KNOWLEDGE
The knowledge required by a smart product for supporting the interaction is stored on the smart product itself in order to keep it autonomous. For example, the smart oven which is able to guide the user how to bake comes along with all the required knowledge so that it does not rely on additional internet connections that might not be available in every kitchen. In this section, we provide an overview of the different types of knowledge to be stored on a smart product which are essential from our point of view for the user interaction.

In order to adapt the interaction to the user's specific needs, a smart product requires knowledge about the user (**user model**) For example, no audio alarms should be used for hearing-impaired persons or a user may prefer textual to audio instructions. Thus, the user model should include *user characteristics* and *long-term user preferences*, taking into account that some preferences may be valid only in certain contexts (like "no audio alarms at night"). The user model should also include *user experience* in performing different operations. For example, a user baking a cake for the first time welcomes detailed instructions, in contrast to an experienced cook.

For assisting the user in executing a task, the smart products need to know which steps are required to complete it. It has been shown that **workflows** are a good means for modelling these instructional interactions. They enable smart products to control the sequence in which instructions are passed to the user. Further, they can consider context data gathered from sensors to adapt the interaction accordingly. We propose to use a semantically enhanced version of the widespread process definition language XPDL [3] for that purpose. Using this language, elements of workflows can be given semantic annotations that allow them to be related automatically to context and to other products in the environment during runtime.

This requires that the smart products have knowledge about the capabilities of other products in the environment, as well as being able to announce their own capabilities. For example, for instructing the user to mix the dough, the smart oven should automatically choose and recommend the best available tool for that purpose, e.g., an electric whisk. This knowledge is modelled in the **domain model.** It concerns which concepts exist in a specific domain and how they are connected. In the cooking domain, concepts might include physical objects like ovens, flour and cakes, but also abstract concepts, like actions such as baking or chopping. Connections between objects include inheritance (fruit cake is a type of cake), mereological relations (flour is an ingredient of a cake) and domain specific relations (baking uses an oven). Besides ensuring the interoperability between smart products, the domain knowledge enables the smart product to present information in human terms, e.g., sending messages such as "the oven is ready" rather than "sensor 7538b reads 181 deg.C".

The execution of workflows usually involves a great deal of user interaction. However, this can be rather tedious as the input and output capabilities of a smart product are often rather limited. For example, a coffee machine usually only has a few buttons and a two line display. For that reason, the required interaction should be minimized and adapted to the current situation of the user. Thus, smart products need to be aware of the current **context**. Besides adapting the interaction, this allows smart products to support implicit interaction and reduce the amount of information that has to be queried from the user.

To sum it up, we see user model, workflows, domain and context knowledge as the key enablers for a meaningful interaction between user and smart product. In the following subsections, we describe how the different types of knowledge can be gathered.

2.1 User Model

The user model can be partially acquired via questionnaires, but it is not feasible to ask the users to fill in long questionnaires as it may be annoying. The most important things to acquire manually are personal *characteristics* of the user, such as health problems. *Long-term user preferences* (along with the situations in which they are valid) can also be specified by the user. However, smart products can also gather knowledge about each user's preferences by observation and analysis of the user behaviour. For example, if a user has switched off audio instructions several times in a row when she was listening to music, this can be done automatically next time when the environmental sensors detect music.

Knowledge about *user experience* should be constantly updated via observations of the user behaviour. For example, a smart product may consider a user an expert in doing something simple, like beating eggs, after she has done it several times. In order to conclude that the user is an expert in more complex things, smart product may additionally check whether a user handles the task quickly or scrolls the instructions back and forth repeatedly.

2.2 Workflows

As mentioned before, we propose to use a semantically enriched process description language. The elements of workflows, such as Activities or Participants, can be given semantic annotations. For example, the baking activity in a recipe workflow is assigned to a Participant with the semantic annotation "Oven", meaning the task can be carried out by any product which is an oven. If the user's kitchen contains a double oven, the workflow can be improved by changing the annotation to specify which of its two ovens is best for baking.

These semantic annotations are intended to support smart behaviour required to carry out a particular workflow step, such as choosing the appropriate smart product for a subtask like mixing dough. When products are being designed, workflows can be written by product developers or partially learned from instructional texts, such as recipe books, then modified. In use, altering the memory of a product to personalise the annotations could be done manually, with the user being asked to nominate which oven they prefer, but this could be annoying. Some changes could instead be observed from the context, e.g., by noting that the user often uses one oven not the other for baking. It is also possible that workflows could be learned by demonstration, with the system observing the changes to context as an expert user carries out a task and transforming these into semantically annotated workflows that could support a novice.

2.3 Domain Knowledge

We have said that, in the cooking domain, concepts might include physical objects, abstract concepts and the relations between them. The completeness of a domain model can never be guaranteed, and therefore we know that domain ontologies shipped with smart products will need to grow and adapt to describe a particular user's environment and interests, in order to better support their interactions. There are two levels of the model description we need to consider. The upper level, known as the ontology level or schema level, is where the structure of the model is defined: the classes and permitted relations between them. Changes at this level are termed *ontology extension*. The lower level, known as the instance level or sometimes the knowledge base, is where specific individuals and their relations are defined. Changes at this level are termed *ontology population*.

Ontology population is more common. Consider the case that the user purchases a new smart product for her kitchen that is equipped with a noise sensor. However, it is not described with the same ontology as the smart oven. For that reason, the noise sensor has to be assigned to the existing classes in the ontology of the smart oven to enable the oven to adapt to environmental noise, e.g., by using visual output instead of audio in noisy environments. This ontology population can be performed manually or (at least partially) automatically by detecting similarities to other noise sensors already linked to the ontology.

Ontology extension should be required less often. Consider the case that the smart kitchen environment is initialized with a generic domain ontology for cooking, which contains the class Cake. For an enthusiastic cook, the ontology may be extended, adding subclasses of Cake such as FruitCake and SpongeCake that differ in their baking process. This enables the oven to automatically adapt the temperature and baking time and thus reduce the amount of interaction required by the user.

2.4 Context Knowledge

The context information is gathered from sensors attached to the smart product itself and also from sensors and smart products in the environment [4]. The context information is semantically described to facilitate the information exchange between different products. Thus, new sensors can be easily used by a smart product if its ontology contains the context information provided by the sensor. Otherwise, the corresponding ontology needs to be populated as described in the previous section. Higher-level context is automatically inferred by a smart product from the basic data gathered from physical sensors, e.g., by using rule based reasoning.

3. SUMMARY

In this paper, we argued why knowledge about the user, workflows, the domain and the current context are important for smart products for supporting the interaction with users. Further, we pointed out how to gather these different types of knowledge.

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).

4. REFERENCES

[1] Wahlster, W., Kröner, A., Schneider, M., and Baus, J. 2008: Sharing Memories of Smart Products and Their Consumers in Instrumented Environments. In: *it - Information Technology* 50(1), Special Issue on Ambient Intelligence

[2] Kawsar, F. 2009. *A Document based Framework for User Centric Smart Object Systems*. Ph.D. Dissertation, Waseda University, Tokyo.

[3] Webster, P., Uren, V., and Ständer, M. 2010. Shaken not Stirred, Mixing Semantics into XPDL. In: *5th International Workshop on Semantic Business Process Modelling (SBPM)* at ESWC.

[4] Miche, M., Schreiber, D., and Hartmann, M.. 2009. Core Services for Smart Products. *In*: *AmI-Blocks'09*, at AmI'09