User-Centered Task Modeling for Context-Aware Systems

Tobias Klug
Telecooperation Group
Technical University of Darmstadt
Hochschulstrasse 10
Darmstadt, Germany
lastname@tk.informatik.tu-darmstadt.de

1. INTRODUCTION

Ubiquitous computing envisions computers as such an integral part of our environment that we cease to notice them as computers. One part of this vision are context aware applications that know about their user's context and are able to intelligently support him doing his work. However, actually achieving context awareness in computer systems is an extremely complex matter.

Compared to traditional software, the context awareness idea is relatively new. Therefore existing software engineering methodologies focus on WIMP (Windows, Icons, Mouse, Pointer) interfaces. For these methodologies it is sufficient to concentrate on the system itself, because there are few if any external influences. As a result these applications are always in control of what happens. With context aware systems this is different. They focus on the real world and the system is just an add-on. Therefore, the designer needs some understanding of what is going on in the real world that might influence the system. In other words s/he needs to understand the user's context. In most cases the relevant context can be defined as the sum of all user's tasks and everything that influences their execution. In the following we will refer to this as the user's work environment.

Understanding a user's work environment is one of the most important steps when designing a context aware application. Users need to be involved in this step to obtain authentic data on how users perform their work, what their environment looks like and what their needs are. User centered design [4] is a paradigm that fits naturally in this application area as it increases the chances of user acceptance.

An important step towards understanding a user's work environment is gathering knowledge about his task and goal structures. This step is already common practice in the HCI community. HCI practitioners use techniques of the family of task analysis and modeling to analyze work processes. The resulting models are used to communicate the user's needs to the design and development teams building the application.

At runtime the system also needs to be aware of the user's environment, because it wants to support him/her in reaching a goal. To achieve that, the system needs to anticipate this goal and to know what needs to be done to achieve it. But goals and the related task structures can be very complex which makes it hard if not outright impossible to infer a user's goal without additional knowledge. Therefore, the system needs a model of the relevant tasks and goals.

It is obvious that models of the user's work environment are of help or necessary during a number of development stages from the initial concept development phase to the runtime phase. The remainder of this paper shows why adapted task models are interesting candidates for the above mentioned purposes.

2. TASK MODELING

The introduction shows that task modeling can be a valuable tool for the development of context aware systems. However the requirements for a successful integration are difficult to meet. The model must be powerful enough to be useful at runtime and at the same time simple enough to be understood by non-technical stakeholders that use it during development.

As task modeling has its origins in HCI, most existing methods originate in this discipline. Depending on their purpose the models vary in their degree of formality. Rather informal models like Hierarchical Task Analysis (HTA) [1] are used for analysis and concept development. These models are easy to understand and can be used as communication artifacts between different stakeholders during the system design. Other models like GOMS [2] and CTT [3] are more formal. These are often used for performance and offline usability evaluations. They are machine readable, but their formalisms make them unusable for any work with non-experts in task modeling.

The latter models have also been used by the model based user interface development community to (semi-)automatically construct user interfaces based on task models. So far these models have mostly been used to develop WIMP and mobile applications. They only capture a user's tasks and hardly any information about the context they are executed in. Integrating context into these models has been attempted, but these approaches describe how the task model changes in a specific context rather than describing why and how a task is influenced by this context.

Both, usability and expressiveness have not yet been achieved in one model because existing models have all been developed with a single purpose in mind. Theoretically it is possible to achieve both in a single model, because user and system are dealing with different representations of a model.

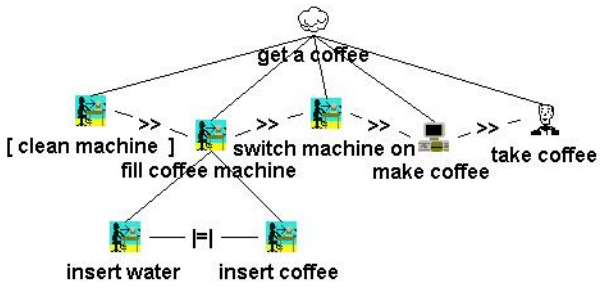

Figure 1: Coffee machine example in CTT notation.

A human is only interested in its graphical representation, whereas the system only understands the formal specification. The challenge is to find suitable representations for both parties and a mapping between these two.

3. PROPOSED MODEL

Task models and their associated context are hard to model because they form a very complex system with many relationships and constraints. Previous approaches have tried to integrate information about the task context into the same representation as the task model itself. The results are monolithic representations too bloated to be useful for any human, especially if end users are concerned. These kinds of models answer the question how tasks and context are interrelated, but not why. Let's consider the example of getting a coffee:

Alice wants to get a coffee from the coffee machine. But before she can get the coffee, she needs to fill in water and coffee. Then she starts the coffee machine and waits until the coffee is ready. If the machine was not clean before she might also need to clean the machine first.

Using CTT to model this process might look like in Figure 1. The process actually involves two interacting parties, Alice and the coffee machine. Although this model is supposed to show Alice's view on the coffee problem, it mixes her tasks with details inherent to the way the coffee machine works. This approach has several disadvantages. If the coffee machine were to be exchanged with a new device, the user process might have to be remodeled, although only the interaction depending on the coffee machine changed. Another problem is the readability of the representation. The only task Alice actually wants to perform is taking the coffee from the machine. All other tasks are only necessary because of the way the machine works.

In contrast splitting the model into one part for each entity involved in the process offers many advantages. Figure 2 shows such a model using a notation similar to UML. The upper part shows Alice's workflow from her point of view. The lower part shows a state machine modeling the behavior of the coffee machine. If the coffee machine needs external input to change state, this is indicated with the name of the task at the transition arrow.

The only link between the two models is the association of the "take coffee" task in Alice's workflow with the "take coffee" transition in the coffee machine's state machine. When Alice wants to perform this task, the coffee machine needs to be in the "coffee ready" state. If this is not the case, Alice needs to perform additional other steps like "insert water" to bring the machine into this state.

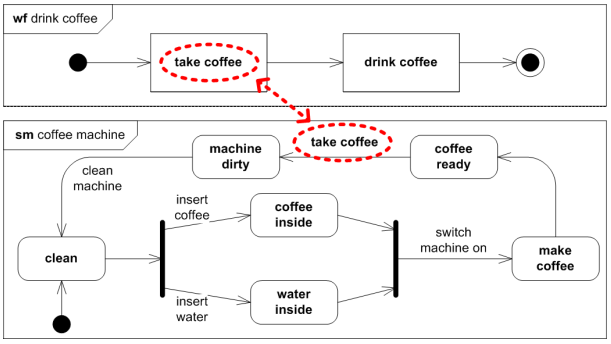

Figure 2: Coffee machine example with separate models for user and machine based on UML.

Using this approach logically separates the different entities within the process. Now, if a new coffee machine were to be purchased, only one part of the model would have to be changed and the link between the parts might have to be adapted. Additionally, the model is much more flexible, because a variety of coffee machine states is handled transparently. If the previous user forgot to clean the machine or already filled in water, The model shows what to do.

This example clearly shows that a lot of dependencies within one work environment can be deducted from its parts. As Figure 2 shows, the only manual dependency that was specified is the dependency between the "take coffee" task and the respective state of the coffee machine. All other dependencies can be derived from the combination of all parts. Together with the separation of concerns within the model the readability of the representation is improved while still maintaining the complexity for the computing system.

4. REFERENCES

[1] J. Annett. *Hierarchical Task Analysis*, chapter 2, pages 17–35. Lawrence Erlbaum, 2003.

[2] B. E. John and D. E. Kieras. The goms family of user interface analysis techniques: comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351, 1996.

[3] G. Mori, F. Paterno, and C. Santoro. CTTE: support for developing and analyzing task models for interactive system design. *IEEE Trans. Softw. Eng.*, 28(8):797–813, 2002.

[4] K. Vredenburg, S. Isensee, and C. Righi. *User-Centered Design: An Integrated Approach.* Prentice Hall, 2001.