User-Oriented Accessibility Patterns for Smart Environments

Michael Zaki and Peter Forbrig

University of Rostock, Department of Computer Science,
Albert Einstein Str. 21, 18059 Rostock, Germany
{michael.zaky,peter.forbrig}@uni-rostock.de

Abstract. One of the main indicators concerning the usability of an application is the corresponding level of accessibility provided by this application. Although a lot of work has been done in the software engineering domain, the accessibility problem has not been enough tackled in the HCI area. In this paper we present an idea to resolve the user-related accessibility problems since the modeling stage of smart environment applications while being assisted by patterns. The proposed idea is to provide two generic patterns used for any accessibility modeling problem, and additionally the creation of two pattern libraries presenting concrete solutions for the most common user-based accessibility problems.

Keywords: Smart Environments**,** Bidirectional Information Accessibility, Task Models, Patterns.

1 Introduction and Motivation

The need for applications taking user accessibility problems into account has obviously increased over the last few decades. 1 out of 5 people in the United States has some kind of disability. The number of people with disabilities is still increasing, as it has increased with about 25 % in the last ten years. Consequently, the fact that an application is universally accessible has become one of the top concerns of this application's usability evaluation. A very common understanding of the word 'Accessibility' defines it as "*Authorization, opportunity or right to access records or retrieve information from an archive, computer system or a website*". In our point of view, the 'Accessibility' must be defined in a broader way while speaking about ubiquitous computing environments and especially smart environments for two main reasons:

a) In [1] smart environments have been defined as "*physical environments where actors are performing tasks cooperatively while being assisted by mobile and stationary devices*". From this definition, one can easily see that a given actor or user in a smart environment is receiving information not only from the existing devices and domain objects but also from the other actors, which implicitly means that in order to solve the accessibility problem we have to guarantee that this user is able to receive and understand the ambient information sent by the other actors as well as the devices and the computing systems in the environment.

J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2011, LNCS 6761, pp. 319–327, 2011.

b) The user needs to perform tasks so that he can successfully accomplish his role and achieve the goal for which he is in the smart environment. But actually in order to accomplish his role, he may also need to deliver information to the other entities in the environments as actors, devices and computing systems. Consequently here we can see another dimension of the accessibility problem, as the user must also be able to deliver information to the environment. This new requirement increases the complexity of the accessibility problem, as we have to guarantee as well that the information provided by this user is accessible to any other entity in the environment.

Taking these requirements into account, we define the word 'Accessibility' in the domain of smart environments in the following manner:

"*The opportunity for all the users to receive and to deliver all kinds of information, regardless of the information format or the type of user impairment*".And so here we are dealing with a need for bidirectional accessibility concerning any information coming from or going towards any user in the smart environment.In Fig.1 the information flow for the role listener in a presentation scenario is depicted, and the direction of the arrows represents the direction of the information. In this paper we present the solution of using patterns for the integration of the accessibility needs into the modeling process of smart environment applications.

The remainder of the paper is structured as follows: In Section 2, the related work and a background about the topic are presented. After that, the approach of using patterns to deal with the accessibility problems is discussed while in Section 4 the presented approach is fostered by an example which illustrates and puts into evidence the use of these patterns. Finally we summarize and give a brief overview of the future research avenues in this domain.

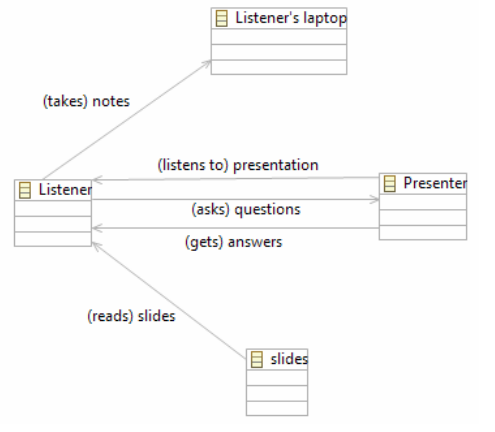

Fig. 1. Information flow for the role 'Listener' in a presentation scenario

2 Background and Related Work

In spite of the fact that patterns were first invented in urban architecture by Christopher Alexander in 1977[2], their influence has spread and reached the software engineering as well as the HCI area. In [3] The Gang of Four have introduced patterns

as recurring solutions for common software design problems. These patterns realized a brilliant success in the software engineering field, and consequently the idea of using patterns has also conquered the HCI domain. In [4] Borchers defines HCI design patterns as "*a structured textual and graphical description of a proven solution to a recurring design problem*". Also in [5] Tidwell defines a collection of interaction design patterns where the solution is expressed in terms of suggested perceivable interaction behavior.

The accessibility is one of the fields where HCI patterns have already contributed. However, a lot of work needs to be done in this direction as the existing patterns are unable to deal with this problem in environments with high complexity like smart environments where the user is exchanging information with all the surrounding entities. For example in [6] accessibility patterns solving the web design and content accessibility problems were discussed, while in [7] an approach merging accessibility and usability UI needs by the use of patterns is presented. The main shortcomings are that these approaches are just coping with Human-web accessibility problems and they present solutions in order to make the application's UI accessible to every user, while in smart environments the user can receive information from other sources and consequently one can have other types of accessibility problems such as Human-human accessibility problems for example. An illustration of such an accessibility problem can be clarified if we imagine the case of a normal lecture taking place in a smart meeting room. If one of the existing students is having troubles to get information from the professor, so in fact this student is having a Human-human accessibility problem. Another major disadvantage is that the solution is usually presented on the UI design level, while it would be better and more effective if we can integrate a solution earlier in the application's modeling stage.

Task models proved a great success as a starting point for modeling applications in the HCI area. In [8] the idea of using task models for smart environment is tackled. The main argumentation why task models are suitable for these environments is the high complexity of task performance, and since the main goal behind smart environments is to assist the user while performing his tasks and to give him a nice experience about the environment, so a successful modeling method should stress on the good understanding of the user tasks in order to be able to deliver a high level of assistance. Several notations for task modeling exist, however the most common one is CTT [9]. It provides a set of temporal operators that express the precise order in which the tasks have to be performed in order to achieve the desired goal.

From this background one can infer the need for patterns being capable to model solutions for any user-related accessibility problem and furthermore these patterns have to be task-based in order to fit into the HCI application domain. In the next section we present our task-based patterns to deal with the user-related accessibility problems in ubiquitous computing and smart environment areas. The notation used for our patterns is the CTT notation.

3 User-Oriented Accessibility Patterns

As already mentioned, in smart environments by using the term 'Accessibility' we mean the capability of any user to receive/deliver any kind of information from/to any

existing entity in the environment without being restricted by the information format or any kind of user impairment. In our point of view, any user-related accessibility problem can be categorized as one of the two following types:

1. ***Input accessibility problem***: The user is not able to get information from one or some entities in the environment.
2. ***Output accessibility problem***: The user is not able to express himself and to deliver information to one or some entities in the environment.

Let us take the example of a presentation scenario in a smart meeting room. In order to be able to follow a presentation the user has to be able to listen to the presenter and to read the slides shown on the canvas. Then, let us assume we have a blind user in this meeting room trying to follow the presentation. This blind user is able to get information from the talker while it is impossible for him to read the slides, and so he is having a serious problem to get information from the slides (The information presented on the slides are considered as input for the user).We can describe this situation by telling that our blind user is having an input accessibility problem.

Now, what if the same user is trying to take notes while listening to the presentation? Actually to take notes, one has to write some points either by normal handwriting or using one of his devices (e.g. laptop). Evidently, our blind user is also unable to take notes in the normal and usual way, but this time he fails to deliver information to one of the environment entities and consequently such a problem is categorized as an output accessibility problem, as the environment is unable to receive information from this user.

If we look to the situation from the modeling point of view, how does the problem look like for a developer who is trying to take the accessibility needs into account while constructing his model for a smart environment application? The problem is that there are some tasks which are modeled as atomic actions in the case of normal users, but now these tasks have to be extended in the case of impaired users. So the developer has to integrate an accessibility mechanism or model for each task the user is unable to perform in the usual manner. To overcome this problem and in order to assist the developer to integrate the accessibility needs into his model we present our idea which is composed of two parts. The first step is to construct the input and output generic accessibility patterns, while the second step is to provide two pattern libraries with some concrete instantiations of the generic patterns for the most common accessibility problems. In the following these two ideas are discussed in further details.

3.1 The Input and Output Accessibility Patterns

Any user-related accessibility problem we can imagine is based on the fact that the current type of the information is not suitable for this user, or in other terms our specific user is unable to receive the information in its actual format. That is why our approach relies on the fact that a typical accessibility problem can be solved by providing a mechanism changing the current format of the information to another suitable and understandable format to the user. On a concrete level, this mechanism is a software performing the information's transformation process from the former to the

recent format. Such softwares already exist, as an example to change the information format from text to speech the program "*verbose*" [10] can be used, while "*Dragon Naturally Speaking 11*"[11] can be used for the inverse process. The basic idea behind the input and output accessibility patterns is to provide a core model which is able to describe the information format adaptation mechanism regardless of the type of information or the kind of user impairment, and consequently the developer can take any kind of user impairment or accessibility problem into account by just instantiating these patterns and integrating them into his model. The existence of users having some kind of disability can take place in several types of smart environments. For example the probability of having impaired users or elderly people in smart homes is relatively high, as smart homes are able to offer them real assistance while performing their daily life tasks. Consequently, it is crucial that those people be taken into account by the developer while constructing his model for such environments. Moreover smart offices and smart meeting rooms are places where the existence of impaired users is also probable. Therefore we present here our patterns which help the developer to consider these accessibility problems within his model.

As a result, in Fig. 2 we have the generic input accessibility pattern containing the input accessibility mechanism which behaves like a black box having as input the information received by the environment in the usual format and as output the information in the new convenient format processed and understood by the user.

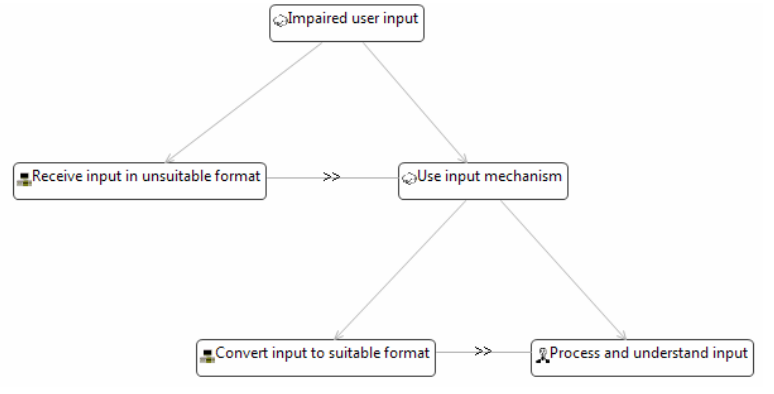

Fig. 2. User-oriented input accessibility pattern

Also in Fig.3 we have the generic output accessibility pattern containing the output accessibility mechanism having as input the information received by the impaired user in the format he can provide it with, and as output the information broadcasted to the environment in the normal format.

3.2 User Impairment-Based Accessibility Patterns Libraries

There is a huge set of possible user impairment types. However, in our point of view the two already mentioned generic patterns can be employed to model any accessibility problem after being instantiated and adapted to the context of use which

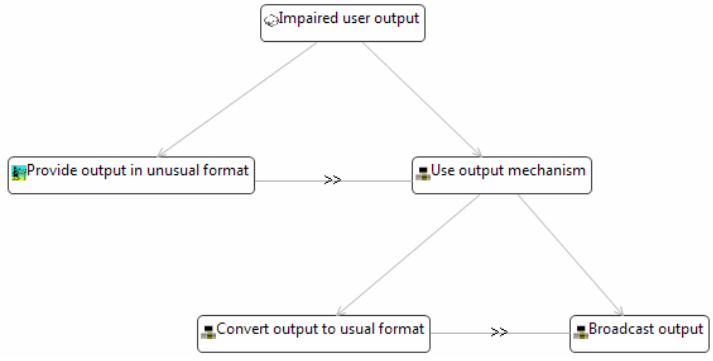

Fig. 3. User-oriented output accessibility pattern

depends on the modality or format of the information and the type of user handicap. In order to maximize the benefit of our patterns we provide here two pattern libraries which contain concrete instantiations out of these generic patterns for the most common and known user impairment types (e.g. blind user, deaf user, paralyzed user…). By having this set of patterns the level of assistance offered to the developer is increased, as for the most expected situations he is able to directly load the convenient pattern from the pattern library without going through any adaptation process. The first library is responsible for any input accessibility problem concerning one of the listed impairment types, and the second library is responsible for the corresponding output accessibility problem. As an example of such patterns we present here the detailed structure of the blind input accessibility pattern. In Fig.4 the solution proposed by the pattern is depicted. Some blind users use screen readers for which the functionality is also based on the idea of changing the modality of the information from text to speech or Braille formats, and so the usage of these screen readers is modeled in the same manner.

Name: Blind input accessibility
Context: A blind user fails to receive information from the environment because this information is presented in unsuitable format (text format).The information type must be changed to a new suitable format in order to be processed and understood by the user.
Problem: How to integrate the information adaptation process into the model.
Solution: Receive the information in its usual format. Use a mechanism (software) in order to convert the information type from the text to the speech format.

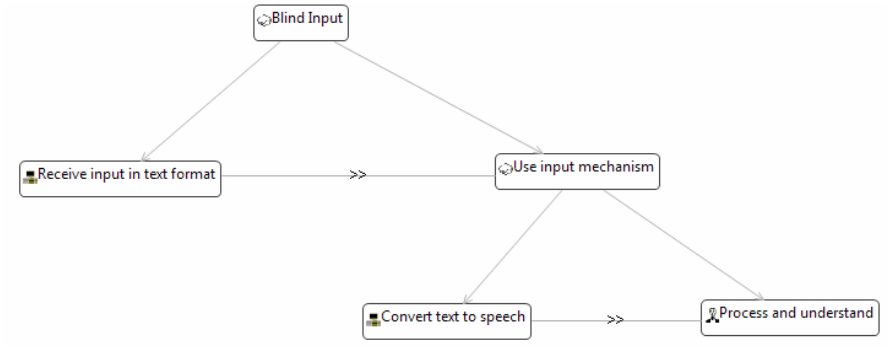

Fig. 4. Blind input accessibility pattern

4 Patterns Application Example

In this section we show an example illustrating the use of our patterns in smart environment application modeling process. In [12] the suitability of CTML (Collaborative Task Modeling Language) as a modeling language for smart environments applications is discussed. Briefly, CTML defines five main steps the developer has to follow in order to model any scenario in smart environment. The first step is to define the roles which are associated to the existing actors in the environment. Let us take the scenario of giving a presentation in a smart meeting room as an example. In this scenario we have three possible roles as we have the chairman, the presenter and the listener. For the sake of brevity, we just focus here on the role 'listener'. In the normal case the role 'listener' is a very simple role to model. A listener has to listen to the talk, take notes and he may ask questions at the end. In Fig.5 this simple case is presented. Normally the listener takes some notes every while during the presentation, and that is why the "Suspend/Resume" temporal operator was chosen before the task 'Takes notes'. Also at the end of the talk this listener may or may not ask questions, and so this task is not mandatory for the accomplishment of his role and consequently the "Choice" temporal operator comes before the task 'Asks questions'.

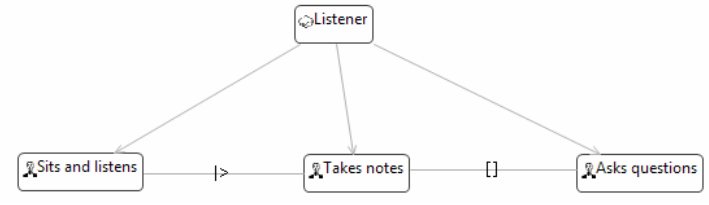

Fig. 5. 'Listener' role for normal user

Now let us think about the same role but in the case of a deaf user. How can the developer use our patterns in order to extend this model and adapt it to the deaf user needs? To have a valid transformation, the developer has to iterate over all the

existing tasks and has to substantiate every atomic action the impaired user cannot perform by one of our patterns. To be able to choose the right pattern, the developer has to decide whether the problem for the current task is an input or an output one. So in this example, we will begin by the 'Sits and listens' task. Listening to a presentation implies getting information from the slides shown on the canvas and meanwhile getting information from the presenter. Our deaf user does not have any problem to read the slides but he is incapable of listening to the presenter and consequently it is impossible to get input from the talker. This is an input accessibility problem and so here we need the input accessibility pattern. If we come to the next task 'Takes notes' we figure out that our impaired user does not have any problem to take notes either by handwriting or even using his notebook. So this task does not need to be extended and it remains just the same. Finally when we come to the last task 'Asks questions', most of the deaf users are not able to speak as well. So our user is not able to raise his hand and just begin asking his questions in the usual way, and here we say that the user is having troubles to express his ideas and to deliver information to the environment and so it's an output accessibility problem and the solution is provided by the output accessibility pattern. In Fig.6 the corresponding role 'Listener' for a deaf user is depicted. In fact the developer is able to load the mentioned patterns from the pattern libraries directly as the deaf user case is an expected one. For other kinds of impairment which are not so common he has to use the generic input and output accessibility patterns by instantiating and adapting them to his context of use.

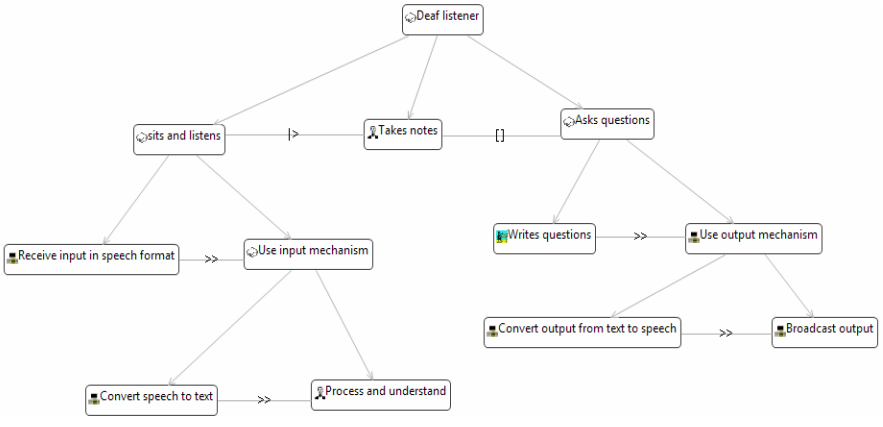

Fig. 6. 'Listener' role for deaf user

5 Conclusion and Future Research

In this paper we discussed the user-related accessibility problems from a different point of view, as we provided a new definition of what accessibility means in the context of smart environments. Based on this definition bidirectional accessibility should be guaranteed to the user, and so we tried to discuss this requirement from modeling point of view. We illustrated the problem as it is seen by the developer by

arguing that some atomic actions in the normal user case have to be extended in the case of an impaired one in order to model his accessibility needs. We provided our input and output generic accessibility patterns which can be employed as a core model to be instantiated and adapted to any user-based accessibility modeling situation. Furthermore, to offer a higher level of assistance we suggested two pattern libraries containing instantiations of the two main generic patterns for the most recurrent user-impairment cases. These pattern libraries save the effort and the time consumed by the developer in order to adapt the generic patterns into his context. To illustrate the use of our patterns we presented a simple case study which implements the way these patterns can help the developer in the transformation of his model from the normal to the impaired user case. Finally, we would like to mention that although the scenario tackled in this paper was always the presentation scenario inside a smart meeting room, but the presented patterns can also offer the same level of assistance for modeling applications in other smart environment domains and scenarios as in smart homes for example. Actually the existence of impaired users can even be more common in smart homes, and so the intervention of our patterns can have much more influence in the corresponding scenarios.

In the future we would like to answer the question whether it is possible or not to provide device-oriented accessibility patterns as well and consequently we can build a pattern language addressing any kind of accessibility problem in the whole environment.

Acknowledgments. This work is supported by a grant of the German National Research Foundation (DFG), Graduate School 1424, Multimodal Smart Appliance Ensembles for Mobile Applications (MuSAMA).

References

1. Wurdel, M.: Making Task Modeling Suitable for Smart Environment. In: ICUMT, pp. 1–6 (2009)
2. Alexander, C., Ishikawa, S., Silverstien, M.: A Pattern Language. In: Towns, Buildings, Construction. Oxford University Press, Oxford (1977)
3. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software (1994)
4. Borchers, J.: A pattern approach to interaction design. In: DIS 2000 (2001)
5. Tidwell, J.: Interaction Design Patterns: Twelve Theses. In: PLoP 1998 (1998)
6. Jeschke, S., Pfeiffer, O., Vieritz, H.: Using web accessibility patterns for web application development. In: SAC 2009 (2009)
7. Vieritz, H., Schilberg, D., Jeschke, S.: Merging web Accessibility and Usability by patterns. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) ICCHP 2010. LNCS, vol. 6179, pp. 336–342. Springer, Heidelberg (2010)
8. Wurdel, M., Propp, S., Forbrig, P.: HCI Task Models and Smart Environments (2008)
9. Paterno, F., Mancini, C., Meniconi, S.: ConcurTaskTrees: A diagrammatic Notation for Specifying Task Models. In: IFIP TC13, pp. 362–369 (1997)
10. http://www.nch.com.au/verbose/index.html
11. http://www.pcworld.com/reviews/product/587538/review/dragon_naturallyspeaking_11_professional.html
12. Wurdel, M., Sinnig, D., Forbrig, P.: CTML: Domain and Task Modeling for Collaborative Environments. J. UCS 14, 3188–3201 (2008)