On the way to digital literacy for everyone: A user interface agent to assist (not only) blind computer users

Martina Weicht[1] and Alke Martens[2]

[1]IT Science Center Ruegen gGmbH, Germany
weicht@it-science-center.de

[2]Dept. of Computer Science, University of Rostock, Germany
martens@informatik.uni-rostock.de

"Interest in design and development of interactive software applications has increased considerably over recent years. The underlying reason for this interest is the need to allow the greatest number of people access to software applications for the largest number of purposes and in the widest number of contexts." (Paternò, 2001)

1 Introduction

While today's screen readers are quite well–evolved and support a variety of applications, difficulties still arise when processing complex documents, especially those not providing adequate metadata, i.e. when they are not well structured or annotated. Especially large websites tend to be very challenging, as they are optimized to their graphical representation rather than to their contents' accessibility. Exploring those websites takes a lot of time and still they may be too hard to understand or to use. In cases where a blind user 'gets stuck' while doing a particular task, they ask for sighted help wherever available. This observation leads to the idea of providing a context–sensitive help in the form of an assistant that knows the application's problem domain and guides the user through the desired task. To achieve this, tasks are modelled using task models that describe a hierarchy of subtasks and their relations to each other. A user interface agent processes these task models and interacts with both the user and the application.
Employing task models to guide users through complex (unfamiliar) tasks will improve usability for the end–user — in our case to visually impaired people. However, this approach can also be used to support novice users, children, elderly people, and people with other kinds of disabilities.
In the following, we will first sketch the state of the art in automated help generation and task models. Afterwards, we will throw a short glance at the ideas and the current development of our project. The paper closes with an outlook.

2 State of the Art

Task Models: The concept of task models originates from model–based software development and is mainly used for designing user interfaces, for usability testing, and for adapting user interfaces to different target devices [1]. All tasks including user interaction and system reaction are modelled and expressed in a task tree leading from the most abstract task down to atomic tasks in the tree's leaves. This hierarchical description can also be used to assist computer users as it corresponds to our general reception of tasks and to the "Divide and Conquer" principle. In his paper entitled "Task models in interactive software systems", Paternò lists a number of purposes in which task models may be useful [2], among them "supporting a user during a session" and "documenting interactive software". It remains to be seen whether task models really provide the optimal solution. Concurrent approaches, which will be investigated during project development, are Petri nets, UML activity diagrams, and workflows.
Automatic Help Generation: During the past years, many approaches on automatic generation of help as well as work on pedagogical agents have been introduced. Today pedagogical agents and avatars are often part of specific, mainly web–based applications. Regarding the large amount of e–learning

systems, in most cases integrated help and user support is realized in a rather coarse way. Besides traditional manuals delivered with applications which hardly consider their context, no overall help is currently available. Exceptions can be found in the area of Intelligent Tutoring Systems, where intelligent help and support is an integral part of system development (see e.g. [3]). Context sensitive help in other than e–learning systems can rarely be found.
One example of an early context–sensitive help system is the paper clip in the MS Office Suite. It pops up during interaction and informs about certain aspects of the application or suggests specific procedures to achieve a certain goal. While this assistance in certain cases may be considered task–oriented, it is perceived as annoying and hardly helpful [4].
The Hymanoid Hyper Help (H3), introduced by Moriyon [5], provides four types of help messages: a description of the item that was selected for help, information on what kind of commands are available, what is displayed, and where the user may click along with the click's consequences. The system does not provide task–oriented help to specific questions like "How can I delete a file?". To enforce explicit interaction with the help system, H3 uses a separate window to show its hypertext help messages. It can also refer to parts of the window, commands, and data displayed by highlighting their location.
The FUSE–System (Formal User interface Specification Environment [6]), a model–based tool for automatic UI generation, provides tool–based support during all phases in user interface design. One of its four components, Plug–In (Plan–based User Guidance for Intelligent Navigation), generates a context–sensitive hypertext help on the current state of the user interface supported by animation sequences that demonstrate complex procedures. This approach is based on state–transition–diagrams, Petri nets, and takes the user's motivation and knowledge into account. Its user model contains descriptions of static and dynamic properties of both entire user groups and indiviual users influencing the user interface itself and the complexitiy of help that is offered to the user. Linking the user interface creation process and the agent module as tightly as done in the FUSE system promises a better agent quality. This, however, is not always the case in software development and does not support existing applications. Therefore, effective means for retroactive creation of user interface agents are needed.
Eisenstein and Rich [7] describe a similar approach along with specific tools to create GUIs and user interface agents from the same task model. Their work is based on Collagen (Collaborative Interface Agent), a Java–based middleware for the creation of those agents. A code generator combines the XML–based task model, task–GUI mapping and GUI specification to create the agent interface to use with Collagen. One version of this agent is PACO (Pedagogical Agent in Collagen), which offers step–by-step guidance along the task model, encourages and corrects the user. This approach, too, is focused on creating both the interface and its agent at the same time. Paco is meant to teach procedural knowledge rather than whole concepts which the user is assumed to be familiar with.

3 Preliminary thoughts on an interface agent (not only) for blind users

The necessity for the user interface agent sketched in this paper emerged from the project SUE (Screenreader & Usability Extensions [8]) at the IT Science Center Ruegen. This open source project is funded by the German Federal Ministry for Labor and Social Affairs and runs from January 2007 until December 2009. Within the project a screen reader for Linux–based graphical user interfaces is developed that supports at least the most common tasks in employment scenarios, including desktop interaction, email processing, web browsing, text and spreadsheet processing. The resulting screen reader will be published under a free license and ongoing support will be provided.
A user interface agent within this screen reader will improve the quality of interaction with the applications listed above by providing context–sensitive help and instruction. While being motivated from a blind user's web browsing experience, the SUE screen reader architecture allows the agent to be customized according to other users' needs, i.e. sighted users can easily turn off any Braille support, but keep the speech output. As the SUE agent is directly integrated into the screen reader environment,

no additional software is needed. The task models are meant to be generic for a certain general task, such as email processing. They are separately mapped to a specific application and its components, e.g. customized to a specific email client software such as Mozilla Thunderbird. A specific voice will allow the user to clearly distinguish the screen reader's voices from the agent's. Additionally, the agent may provide information on key commands to trigger interface elements and therefore help the user to learn those by heart as they are an important means of interaction especially for blind users. It may provide alternative ways to perform a task and even background information on why or how a certain action is performed or which consequences an action leads to.

In the context of visually impaired computer users, a graphical avatar is obviously not very accessible, but it may help sighted users, especially children. Instead, blind users may profit from an appealing, friendly sounding voice. Within their screen readers, blind users often prefer to use synthesized voices as those are very responsive and allow for an efficient workflow. A synthesized voice's major drawback when compared to a natural one is its speech quality, which is often conceived as mechanical or tinny. Within the assistance scenario, working speed is less important as the user may need some time to recapitulate which steps have been taken or may be taken next. Therefore, a natural voice may be the right choice for a user interface agent in SUE. This voice should motivate and affirm the user to continue, creating some kind of "persona effect" [9] — it is questionable whether a synthetic voice could even achieve this.

During assistance the user interface agent should provide a list of the sub–tasks that have already been performed, e.g. for later reconstruction of the particular steps taken, and of those tasks that can be performed at the current state. Those lists adhere to the way many visually impaired people like to navigate in documents: via lists of objects of a certain type such as headlines, links, etc.

The user may exit the help system at any point to continue on their own. This allows them to only ask for support for particular parts of a complex task and continue autonomously on familiar sub–tasks. Also, the reverse approach should be offered: the user may ask for assistance at any point in the system where they 'get stuck'. At this point the agent searches for available task models at or around the current view or control and guides the user through the remainder of the task. That way the user can start on their own until they need assistance.

In addition, the user interface agent for SUE will need user profiles containing information on the user's knowledge, their age, their special needs and even the assistive technology used to automatically adapt the type and amout of information given to them. The agent should adapt to the user's skills and reflect their level of mastery, but also return to a lower level when the user repeatedly fails on tasks they claimed to already know.

In SUE we will re–use experiences from the advances sketched in the previous section and extend them for the special needs of visually impaired people. Expanding the H3 approach, SUE will implicitly answer questions, e.g. as she guides the user towards the deletion of a file. In addition to information pop–up–windows in H3, SUE will also provide its output via audio and Braille which are the most important means of interaction for blind computer users. Visually impaired users may configure the help system according to their needs, i.e. to receiving help messages via speech output while their current location within the application remains visible on their Braille display. The use of hypertext links within the explanatory text may confuse the user and move their attention from the task to be performed to following links that may not directly relate to the task. For SUE pointing to GUI elements related to tasks is considered very useful. A sole color highlighting is of course of little help when dealing with blind users, but may support other types of users or those in collaboration scenarios. It remains part of research to find out whether a description of the component's location should be given or whether the cursor should directly be positioned to that location (or a combination of both). When offering additional information on each step, SUE can combine both procedural and conceptual knowledge for a more effective use of the respective application.

Contents for the user interface agent may derive from the training course material compiled by the Study Centre for Visually Impaired Students (SZS [10]) at Karlsruhe University, one of the SUE project partners. This material is based on the syllabus for the European Computer Driving License (ECDL [11]),

adapted to Linux operating systems and to screen reader use. While it is surely not comprehensive, it provides a basic repository to get the agent started. All content should of course adhere to current e–learning standards for easy authoring, handling, and re–use.

4 Conclusion

On the way to digital literacy for everyone via SUE's user interface agent there are still many unanswered questions. While a highly configurable agent is clearly needed, and extensive user tests are required for verification, the sketched agent may indeed help different kinds of users and provide them with additional usability and support. However, its success largely depends upon the quality of the agent's interface and the availability of task models and mappings to specific applications, so those aspects need to be considered carefully when designing the user interface agent for SUE.

References

[1] Paternò, F.: Model–Based Design and Evaluation of Interactive Applications, Springer, 2000.

[2] Paternò, F.: Task Models in Interactive Software Systems. In Handbook of Software Engineering & Knowledge Engineering, 2001.

[3] Martens, A. Ein Tutoring Prozess Modell fuer fallbasierte Intelligente Tutoring Systeme AKA Verlag infix, 2004.

[4] Swartz, L.: Why People Hate the Paperclip: Labels, Appearance, Behavior and Social Responces to User Interface Agents. Bachelor thesis, Stanford University, 2003.

[5] Moriyon, R., Szekely, P. Neches, R.: Automatic Generation of Help from Interface Design Models. In Celebrating interdependence: CHI '94 conference proceedings, 1994.

[6] Lonczewski, F., Schreiber, S.: Generating User Interfaces with the FUSE–System, 1996.

[7] Eisenstein, J., Rich, C.: Agents and GUIs from task models. In Proceedings of the Sixth International Conference on Intelligent User Interfaces, pages 47–54, 2002. ACM Press.

[8] SUE: Screenreader & Usability Extensions. `http://sue.sourceforge.net`

[9] Lester, J., Converse, S., Kahler, S., Barlow, T., Stone, B., Bhogal, R.: The Persona Effect: Affective Impact of Animated Pedagogical Agents. In CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, 1997.

[10] Study Centre for Visually Impaired Students (SZS). `http://www.szs.uni-karlsruhe.de`

[11] DLGI mbH: European Computer Driving License. `http://www.ecdl.de`