Accessible Screenshots for Blind and Visually Impaired People

Christin Engel[1], Denise Bornschein[1], Gerhard Weber[1]

Chair of Human-Computer Interaction, TU Dresden[1]

[christin.engel,denise.bornschein,gerhard.weber]@tu-dresden.de

Abstract

Training material to learn new applications mostly includes screenshots to describe the structure of an application or to represent required interaction steps. Blind people have no access to such graphics that are needed to learn handling an application. However, screen reader users need adapted information to learn effective interaction strategies in GUI environments. We propose a concept to make screenshots accessible for blind screen reader users by mapping UI elements and states to HTML elements. Based on a pilot study with two blind people, we found requirements that ended up in a tool that automates the generation of interactive, accessible screenshots.

1 Introduction

According to the German Federal Statistical Office (Statistisches Bundesamt, 2017), 94 % of the companies make use of computers. Overall, 55 % of employees use a computer with internet access relating to business. The use of a computer is a main requirement for many jobs. Blind people frequently work with computer systems (Barnicle, 2000). Learning the effective usage of applications is linked with training and tutorials. Training materials – in particular handbooks and web-based training – frequently make use of screenshots to explain the handling with a specific application. They present, for instance, step-by-step instructions, navigation sequences, the user interface (UI) and its regions as well as widgets and icons. In addition, screenshots contain additional elements, such as annotations or arrows. For a sighted computer user, it is easy to identify relevant information from such complex graphics. Blind users have no access to a screenshot's content, which hinders them from learning the usage of applications independently. As not only training materials, but also examinations can base on screenshots, for blind people it is also difficult to get a certificate. In the end, enabling blind people to access screenshots could improve their opportunities of education and training.
An established way to make the content of screenshots accessible for blind people is to provide a

Veröffentlicht durch die Gesellschaft für Informatik e. V. 2018 in
R. Dachselt, G. Weber (Hrsg.):
Mensch und Computer 2018 – Tagungsband, 02.–05. September 2018, Dresden.
 https://doi.org/10.18420/muc2018-mci-0306

verbal image description. There is a lack of guidelines and studies concerning the description of screenshots. If the blind user is familiar with an application, it may be suitable to provide a short description of the screenshot (e.g. an explanation of the path to show navigation within menus). Furthermore, generating and reading image descriptions is time-consuming and requires high cognitive resources. Screen reader users must switch between the description that is read out by the screen reader and the application. Embedding instructions concerning specific interaction steps, could support the simultaneous work with the application and the training material. To learn a new application, blind people do not only need information about the structure, but also strategies to handle the application with a screen reader. In addition, an accessible screenshot must provide the content of additional elements, which were shown on the graphical screenshot (e.g. frames, annotations, states of interaction objects). Furthermore, the accessible screenshot must include all information that is represented visually exclusively, e.g. layout information, colored elements, grouped elements or icons.
Current research mostly focuses on how to train blind people in the usage of Graphical User Interfaces (GUIs), but there is no specific research that explores accessible screenshots for blind and visually impaired people. In this paper, we present a concept for transcribing graphical screenshots into HTML and corresponding requirements. Furthermore, we show a first prototype that automates the generation of such accessible screenshots in HTML.

2 Related Work

The BITV (Barrier-free Information Technology Ordinance)[1] demands the support of text alternatives to enable the access to graphics in general. Currently, no guidelines how to describe screenshots exist. Additionally, it is unclear whether a textual description is suitable to teach the handling of applications effectively. On the one hand, generating effective text alternatives for screenshots is time-consuming and requires knowledge about non-visual computer interaction. On the other hand, it takes a high effort from the user to follow the description while interacting with the application simultaneously. Otherwise, the user could listen to the description first, and remember the content while handling the application.
Blind people use assistive technologies to handle applications. Screen readers read out the content of the screen sequentially. The user focuses windows, regions and widgets one after another by navigating through keyboard input (Kochanek, 1994). The screen reader outputs detailed information about focused elements, such as the name, the status (selected/not selected) or the functionality. However, the majority of training material just bases on interaction with the computer mouse. Consequently, blind and visually impaired users may need adapted information to learn the use of applications.
There is initial research concerning the non-visual use of computers. Barnicle (Barnicle, 2000) explored strategies that blind users apply while interacting with GUIs. He pointed out that blind users need efficient search strategies or direct keyboard input to avoid searching item by item sequentially. He stated that blind users have to identify the meaning and intent of a control item just by the name and type. They do not have access to the appropriate labeling of the

[1] http://www.bmas.de/DE/Service/Gesetze/barrierefreie-informationstechnik-verordnung-2-0.html (zuletzt besucht am 03.07.2018)

control item. Following Barnicle, blind users miss important information that sighted users get through visual indicators. However, screen reader users meet several barriers when interacting with software – in particular they miss contextual information of UI elements, they were not notified when changes appear and they have to reach dozens of elements item by item if no efficient search strategies or special keyboard access are supported (Barnicle, 2000).
Requirements for the training of blind people in the use of GUIs were stated by Weber et al. (Weber et al., 1994). Icons and other visual concepts are not intuitive for blind people. That is why these concepts have to be introduced while train blind people to handle GUIs. Weber et al. also stated to support the blind user by providing a tactile representation of the GUI. In another publication, Mynatt and Weber (Mynatt and Weber, 1994) discuss approaches to built intuitive and efficient non-visual interfaces. They defined four design issues for GUI access:

1. **Coherence:** The visual and the non-visual interface should be coherent to support the collaboration between sighted and blind users. Both should support the same mental model.
2. **Exploration:** While GUIs provide much information at a time, non-visual interfaces present information sequentially. That is why there is a need for additional exploration functions in non-visual interfaces.
3. **Graphical Information:** Graphical information, such as selected states or icons, must be presented in the non-visual interface. Furthermore, graphics, animations and multimedia files have to be accessible.
4. **Interaction:** The GUI has to be accessible via keyboard input. Mouse events – in particular dragging or clicking – have to be replaced.

All these design issues are important for accessible screenshots. In addition, screenshots should support the user in learning to handle an application. For that, an UI element mapping approach could be promising. For instance, Mynatt and Weber (Mynatt and Weber, 1994) presented some exemplary mapping from UI elements to Braille-based presentation, speech-based presentation and nonspeech-based presentation to transform information about UI objects in another modality.
Other approaches focus on Abstract User Interfaces (AUI) (Van Hees and Engelen, 2006) or on the development of an Off-Screen Model (OSM) (Kochanek, 1994). The latter developed a model that represents the structured information of an application. The resulting OSM has a tree structure to present interaction objects and areas, which can be useful to support speech or Braille output.
In contrast, Wersényi (Wersényi, 2010) presented auditory icons to improve the perception of icons in non-visual interfaces. Hailpern et al. (Hailpern et al., 2009) introduced an embedded online tutorial system for screen reader users. The authors describe how to embed an interactive tutorial system within a web application to support the learning of new web applications.
Overall, much research focuses on web accessibility. Other approaches explore requirements for accessible GUIs, the structure of GUIs and how to present it to blind people, or investigate barriers occurring for screen reader users. It is certain that blind people need other information for learning new applications than sighted users. Therefore, training materials have to be adapted for screen reader users. This implies the need for concepts making screenshots accessible for blind users.

3 Accessible Screenshots for Blind Users

Traditionally, screenshots were produced by the author of training materials as graphical image. The majority of screenshots includes additional visual elements that were added manually by the author. To identify typical use cases and elements within screenshots, we firstly have analyzed several existing screenshots – in particular screenshots from a manual published by the TU Dresden that explains how to generate accessible pdf-files from Microsoft Word (Loitsch and Prescher, 2011) – and those from the web.
After this analysis, we had to produce an alternative representation for the pixel-based screenshot and its elements. To meet the already mentioned design issues for GUI access (coherence, exploration, graphical information and interaction), the screenshot has to be transcribed into an accessible format that includes semantics. As HTML is highly accessible for screen reader users, our proposed concept will use it as base.

3.1 Use Cases for Screenshots

We identified three use cases for screenshots in the context of training material. For each, we defined a specific task that the screenshot meets as well as typical elements that these kinds of screenshots include:

1. **Overview and Orientation:** The screenshot shows an overview of the whole interface of the application including regions and areas (see Fig. 1). There is no need to perceive all available functions and icons. The purpose is to show the spatial arrangement, names and general functionality of the user interface.
   Typical Elements: Frames around the border of regions/windows to highlight them, annotations, arrows

2. **Navigation:** Such a kind of screenshot illustrates how to interact with the user interface to decrease visual barriers. The screenshot shows interaction paths as well as states of interaction objects – in particular selected menu items or states (see Fig. 2). These screenshots support the user in searching for specific interaction objects, menu items or icons.
   Typical Elements: Visual highlighting of interaction objects, including state and position; additional numbers and annotations to describe interaction steps; opened menus with highlighted menu items; multiple views of the same window (e.g. before and after instructed user input)

3. **Training/Verification:** The screenshot shows the result that the user should reproduce, for example a text with specific formatting. These screenshots may describe a task that the user should archive. In addition, the user may use the screenshot to verify the correctness of performed manipulations.

In addition, a lot of screenshots have no specific function, but only a decorative purpose (e.g. visualization of icons). Screenshots are very individual as additional elements were mostly added manually by the author. Furthermore, they may contain redundant information that already was explained textually, or exclusive information that was just shown within the screenshot.

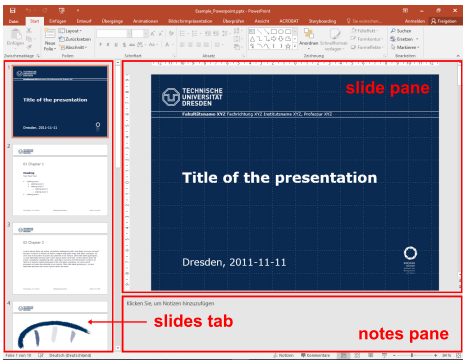

Figure 1: Example screenshot for use case "overview and orientation".

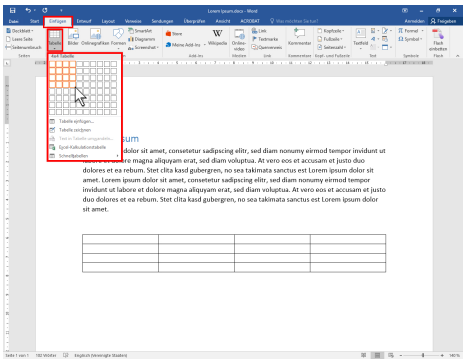

Figure 2: Example screenshot for use case "navigation".

3.2 Challenges and Requirements for Screenshots in HTML

HTML can be used to represent structures and containers. In addition, HTML can contain input elements – in particular buttons, radio buttons or checkboxes – which are very similar to interaction objects in user interfaces. The W3C recommends to use *Accessible Rich Internet Application (ARIA)*[2] that adds semantics to HTML. ARIA can be used to represent advanced UI elements, such as ribbon menus, for screen reader users. Therefore, HTML may be suitable to present accessible screenshots.

First, we mapped UI objects to HTML to create an interactive, accessible version of the screenshot that can be explored by screen reader users. These kinds of screenshots could not only provide the visual information shown on the inaccessible original screenshot, but also such information that blind users need in order to understand the application as well as the interaction model of the application. Furthermore, screenshots in HTML format can be easily enhanced with additional content such as annotations. Blind users can use the interactive screenshot to develop interaction strategies without having the application itself.

It is important that the HTML screenshot version supports the same Look and Feel like the original UI. This is necessary for the cooperation between blind and sighted users, which is an important requirement to support the inclusion of blind people. Blind users should transform knowledge and interaction strategies from the HTML screenshot to the real application. That is why it is highly important that the HTML version operates similar apart from functionality. In addition, the HTML version has to reflect the accessibility of the original application. It follows that inaccessible objects in the original UI are just inaccessible in HTML. However, the accessible screenshot should point out accessibility barriers. Furthermore, we consider to link screenshots to support fluid integration of several interaction steps needed to accomplish tasks.

[2]https://www.w3.org/WAI/intro/aria (zuletzt besucht am 03.07.2018)

3.3 Pilot Study

To proof our concept and find further requirements and challenges we have conducted a pilot study with two congenitally blind women. As test material we used two screenshots of Microsoft Word from (Loitsch and Prescher, 2011). The first scenario included the usage of Word style sheets, the other one dealt with inserting a table within a Word document. The study was realized as informal interview. In the first part we asked for the participants' experiences with Word, in the second part the users had to explore HTML prototypes of the two test screenshots, which were produced manually, with their screen reader.
Both participants already have experiences with Word and use it several times a week. Participant 1 uses the screen reader Jaws 17 with Word 2013 and participant 2 uses Jaws 15 with Office 2010. Usual functionalities, such as lists or headings, are usable without difficulties via shortcuts. However, keyboard shortcuts vary between the different Word versions and, therefore, have to be learned again by blind users. According to our participants, references and tables are much more challenging. Furthermore, sometimes the screen reader output lacks sufficient feedback (e.g. current formatting of the text).
We also asked for the users' expectations of a Word tutorial. Beside teaching the conceptual model (i.a. ribbons and menu navigation), blind users especially want to know effective strategies (shortcuts) as well as the screen reader output. According to our participants, accessible screenshots within a tutorial would be very helpful if their handling (including keyboard shortcuts) corresponds to that of the original application. To allow for an efficient navigation through the screenshot, additional interaction strategies are necessary. For instance, if a user only wants to know which GUI elements are highlighted, s/he maybe not wants to navigate through the whole screenshot by using the tab key, but s/he rather wants to directly jump between these elements by using a special shortcut. Furthermore, the screenshot should also visually look like the original application to allow for an interchange with sighted people and to enable the same training material for all user groups.
To get some information about the screen reader output within the original Word application, our participants should show us how they normally operate the above mentioned scenarios (style sheets and tables). It became clear that the handling of Word and the used keyboard commands not only depend on the used Word version, but also on the screen reader version and its settings/extensions. For instance, in Jaws 17 the ribbon subgroups can be navigated by the left or right cursor key, while in Jaws 15 each single menu item has to be navigated separately.
These results indicate that transforming screenshots to HTML could improve the accessibility and the usability of learning material. That is why we have implemented a prototype that generates these kind of screenshots automatically.

4 Prototype to Automate the Generation of Accessible Screenshots in HTML

We have developed a first prototype as well as some basic authoring tools that allow editing the screenshot by the author. In the following sections we present the workflow of the implemented prototype.

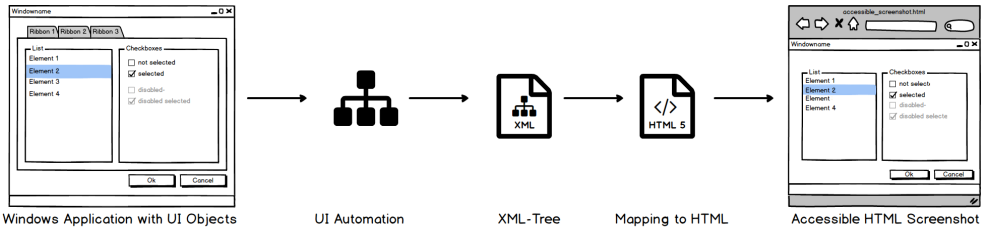

Figure 3: Implemented workflow for generating accessible screenshots in HTML: 1. Extracting UI information via Microsoft's UI Automation API, 2. Generating an XML tree structure, 3. Mapping XML elements to HTML, 4. Interactive, accessible screenshot in HTML.

GUI Element	*HTML Tag*	*GUI Attribute*	*HTML Attribute*
Radiobutton	Input	HelpText	Title
Combobox	Div	LabelledBy	ARIA labelledby
Edit	Textarea	IsKeyboardFocusable	tabindex, disabled

Table 1: Exemplary mapping from UI elements and attributes to HTML.

4.1 Workflow to Map GUI Elements to HTML

Our implemented workflow is shown in Figure 3. We utilize the UI Automation Framework from Microsoft[3] to extract the OSM from a given windows application. It is the same API that screen readers use. We clean the OSM by removing meaningless elements, such as empty elements or elements without a size. On that basis, we generate an XML tree where each UI object and its properties are represented by one XML element with attributes. We implemented an *ElementBuilder* that maps XML elements to HTML tags. Some UI elements – in particular buttons, checkboxes, radio buttons – are mapped directly to HTML elements (see Table 1). For advanced UI elements, such as tree views, ribbons, windows or split buttons, no equivalent HTML elements exist. We tried to recreate these elements in HTML by composing several basic HTML elements. We add the needed semantics, including marked elements, with the help of ARIA attributes and roles. Thus, we define mappings so that the generated HTML elements produce similar screen reader output like the original UI objects. We do not handle custom defined elements and images yet.

4.2 Automated Tool to Generate Accessible Screenshots in HTML

The generation of HTML-based screenshots should be done by authors of training material, which are very often non-expert users. That is why it is necessary to provide tools for editing

[3] https://msdn.microsoft.com/de-de/library/windows/desktop/ee684009(v=vs.85).aspx

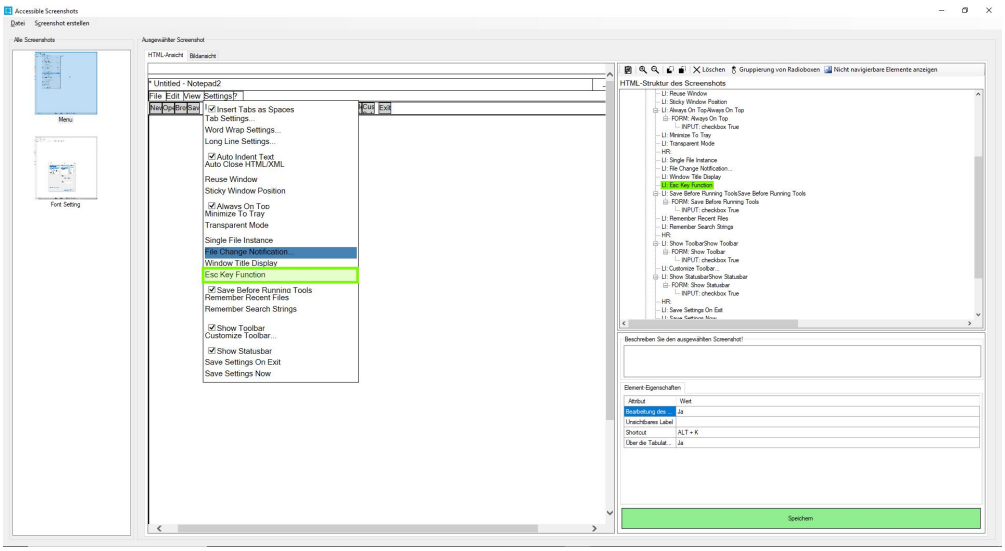

Figure 4: GUI of the current prototype with two exemplary screenshots of Microsoft's Notepad.

the HTML elements after automated generation. Therefore, we have implemented tools to enhance the original screenshot including highlighting elements and groups, defining interaction steps or provide a description. Figure 4 shows the interface of our prototype. To generate a screenshot the user has to simply press a button and the application minimizes. After that, the user has to switch to the target application and press a shortcut to generate a screenshot from the current focused application. The user can choose between two screenshot modes: Generate a screenshot of the whole application that is focused, including all child windows, or just include the current focused window. Afterwards, the user can input a title for the generated screenshot. The tool generates both, a graphical screenshot and the equivalent in HTML. On the left side of the application there is a listing of all generated screenshots that can be opened by clicking on the thumbnail. The application provides two views of the screenshot. First, a WYSIWYG-editor that shows the screenshot in HTML in the middle of the application and a tree view of all elements on the right side. Both views are linked with each other while selected elements were connected visually by brushing and linking (highlighted in green). At the bottom right corner the application provides important accessibility attributes of the current selected element. In addition, focused elements are highlighted visually and textually in the HTML version.

Furthermore, there is a need to fix several issues that arise in the automating process, such as renaming, reordering or deleting elements or grouping radio buttons. These changes can be done within the tree view (see Figure 5). Furthermore, inaccessible elements can be highlighted within the HTML-view. The author can switch to the tabbing mode, which allows to navigate by keyboard through the HTML screenshot without leaving the HTML-view. The author can define a link to another screenshot as well to ensure a seemless integration of the screenshots into the workflow and to show connections between UI objects.

Graphical screenshots are mostly remastered to point out specific elements or interaction steps. That is why we implemented a concept that allows to learn all needed interaction steps within

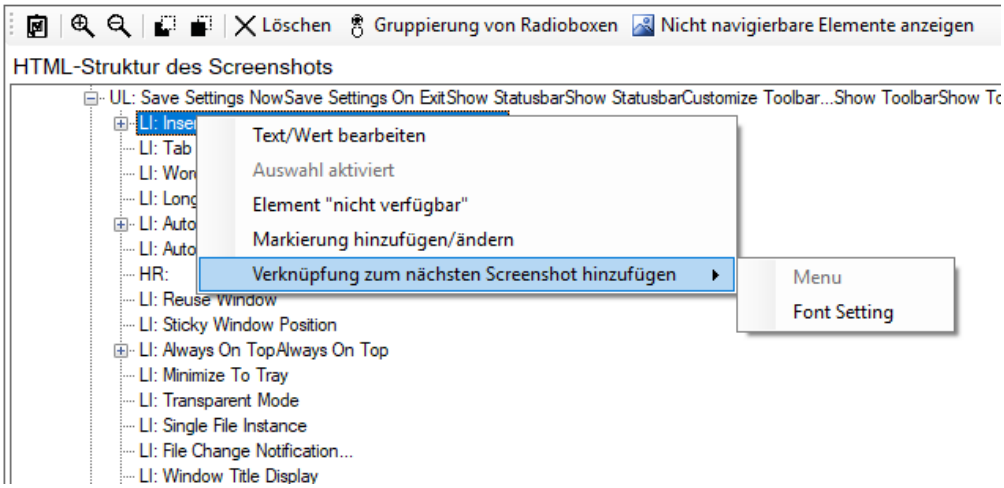

Figure 5: Additional functions provided by our tool to edit HTML elements (e.g. deleting, changing the order of elements or renaming).

the HTML screenshot. Therefore, the author defines an order for target interaction objects and gives a short description about the required interaction step (see Figure 6). This is useful when the screenshot combines multiple interaction steps. To just explore the relevant interaction objects the user can fade-out all other elements.

5 Discussion and Conclusion

In summary, we proposed a concept to provide accessible screenshots in training material and tutorials for blind people by mapping UI elements to HTML. Therefore, we have analyzed current screenshots and identified three key functions for which these kinds of graphics are used for. In addition, we pointed out typical elements used within screenshots. On the basis of a pilot study with two blind people, we summarized requirements that screenshots in HTML should fit. Finally, we proved our concept by implementing a prototype that automates the mapping between GUI elements and HTML. Furthermore, we implemented several authoring tools to include additional elements, such as step-by-step instructions.
In the future, we will implement authoring tools that enable enhancing the graphical and the HTML screenshot simultaneously (e.g. adding borders or highlighting elements). The usability can also be improved by automating the description of the given screenshots. We found out that keyboard navigation through UI elements sometimes differs in comparison to the same HTML elements. To support the same Look and Feel, we will adjust the navigation with JavaScript. Existing accessibility problems will be preserved in the HTML screenshot. Nevertheless, the tool may be usable to inspect accessibility barriers that can be communicated to blind users (e.g. if UI elements are inaccessible, the blind user can not find them). For that, it would be suitable to provide screen reader output inside the tool.

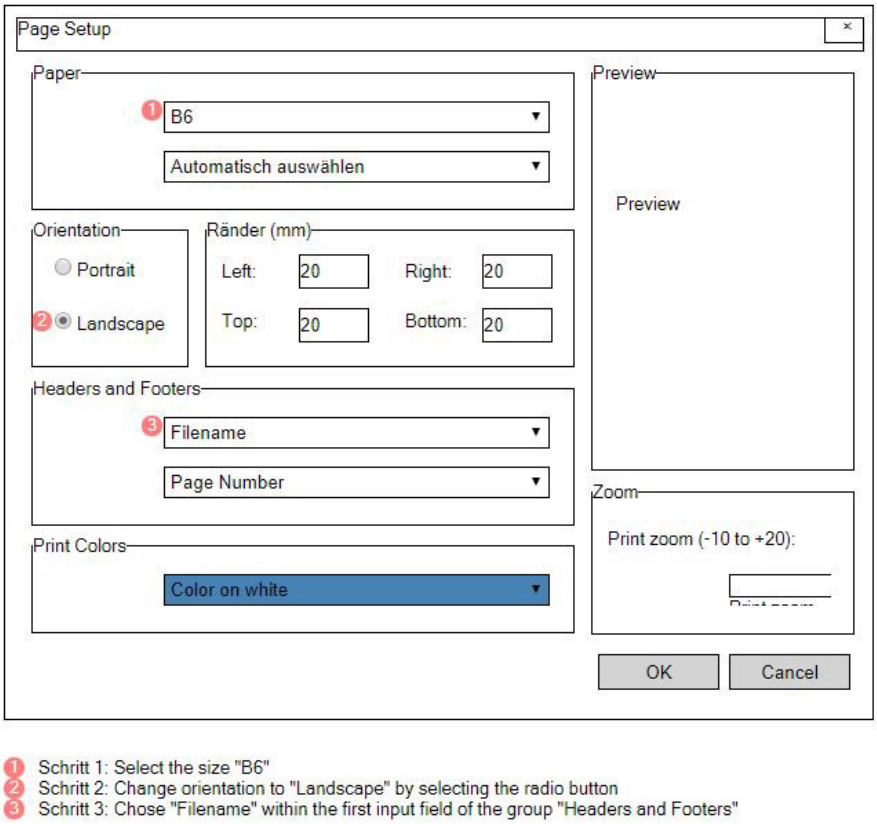

Figure 6: Enhanced HTML screenshot to provide a step-by-step instruction.

We have shown a concept for accessible screenshots that could improve the learnability of applications for blind users. HTML-based screenshots include adapted information that screen reader users need for learning the effective usage of a GUI. Furthermore, the screenshot supports the understanding of the conceptual model of an application, which is helpful to develop effective interaction strategies.

For future development, we will evaluate the resulting HTML-based screenshots as well as the handling of our tool with blind users. In addition, we will test the generation of screenshots from different applications to identify further issues. It is also necessary to include different screen readers within the evaluation.

Learning the structure and logical arrangement of a new application could be supported by providing an overview with the help of a tactile graphic. Therefore, we plan to additionally provide an abstract presentation of the application as well as a way to communicate icons within a tactile representation.

However, for blind people it is very hard to learn how to handle applications with traditional

training materials independently. Dealing with applications and learning new applications is a main requirement for many professions. Blind people not only need access to applications, but also the possibility to learn the handling effectively.
In the end, other target groups could profit from enriched HTML screenshots as well. For instance, visually impaired people could adapt the HTML output to their individual needs by using Cascading Style Sheets (CSS). Special annotations within screenshots can also support the learning process of an application for sighted beginners or people with learning difficulties. Thus, our tool may help to make learning materials more accessible and usable for diverse user groups.

Acknowledgments

We thank the participants of the pilot study for their contribution. We also thank Philipp Rabe, Samory Ka, Duc Anh Trinh and Dennis Metzger for their support. The paper is part of the Mosaik project, which is sponsored by the German Federal Ministry of Labour and Social Affairs (BMAS) under the grant number 01KM151112. Only the authors of this paper are responsible for its content.

References

Barnicle, K. (2000). Usability testing with screen reading technology in a windows environment. In Proceedings on the 2000 conference on universal usability (pp. 102–109). CUU '00. Arlington, Virginia, USA: ACM. doi:10.1145/355460.355543

Hailpern, J., Reid, L. G., & Boardman, R. (2009). Dtorial: An interactive tutorial framework for blind users in a web 2.0 world. In T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. O. Prates, & M. Winckler (Eds.), Human-computer interaction – interact 2009 (pp. 5–18). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kochanek, D. (1994). Designing an offscreen model for a GUI. In W. L. Zagler, G. Busby, & R. R. Wagner (Eds.), Computers for handicapped persons (pp. 89–95). Berlin, Heidelberg: Springer Berlin Heidelberg.

Loitsch, C. & Prescher, D. (2011). Barrierefreie Dokumente I - Anleitung zur Erstellung barrierefreier PDF Dokumente aus Word. Technische Universität Dresden, Institut für Angewandte Informatik, Professur Mensch-Computer Interaktion. Retrieved from https://elvis.inf.tu-dresden.de/dokumente/upload/c56da_barrierefreie_pdf_word.pdf

Mynatt, E. D. & Weber, G. (1994). Nonvisual presentation of graphical user interfaces: Contrasting two approaches. In Conference companion on human factors in computing systems (pp. 211–). CHI '94. Boston, Massachusetts, USA: ACM. doi:10.1145/259963.260338

Statistisches Bundesamt. (2017). Nutzung von Informations- und Kommunikationstechnologien in Unternehmen. Statistisches Bundesamt (Destatis). Retrieved from https://www.destatis.de/DE/Publikationen/Thematisch/UnternehmenHandwerk/Unternehmen/InformationstechnologieUnternehmen5529102177004.pdf

Van Hees, K. & Engelen, J. (2006). Non-visual access to GUIs: Leveraging abstract user interfaces. In K. Miesenberger, J. Klaus, W. L. Zagler, & A. I. Karshmer (Eds.), Computers helping people with special needs (pp. 1063–1070). Berlin, Heidelberg: Springer Berlin Heidelberg.

Weber, G., Petrie, H., Kochanek, D., & Morley, S. (1994). Training blind people in the use of graphical user interfaces. In W. L. Zagler, G. Busby, & R. R. Wagner (Eds.), Computers for handicapped persons (pp. 25–31). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wersényi, G. (2010). Auditory representations of a graphical user interface for a better human-computer interaction. In S. Ystad, M. Aramaki, R. Kronland-Martinet, & K. Jensen (Eds.), Auditory display (pp. 80–102). Berlin, Heidelberg: Springer Berlin Heidelberg.