A Concept for User-Centered Development of Accessible User Interfaces for Industrial Automation Systems and Web Applications

Farzan Yazdi[1], Helmut Vieritz[2], Nasser Jazdi[1], Daniel Schilberg[2], Peter Göhner[1], and Sabina Jeschke[2]

[1] Institute of Industrial Automation and Software Engineering, University of Stuttgart, Pfaffenwaldring 47, D-70569 Stuttgart, Germany
{farzan.yazdi,nasser.jazdi, peter.goehner}@ias.uni-stuttgart.de
[2] Institute of Information Management in Mechanical Engineering, RWTH Aachen University, Dennewartstraße 27, D-52068 Aachen, Germany
{helmut.vieritz,daniel.schilberg, sabina.jeschke}@ima-zlw-ifu.rwth-aachen.de

Abstract. The importance of industrial automation systems and Web applications, often in combination, has been a growing area during the past decades. They are becoming an inseparable part of our lives. Hence they must be accessible to all users. Often certain user groups are being neglected in the development of such systems. Therefore, a systematic concept is required to support the development of such systems. In this paper, we will present a concept, which is proposed in the context of an ongoing research project, addressing the accessibility problem of the user interface. Both, industrial automation systems and Web applications share similar accessibility requirements. Hence, our concept addresses both systems, while discussing the suited methods to effectively assess accessibility requirements.

Keywords: Accessibility, User-centered Design, User interface modeling, Model-driven development.

1 Introduction

To overcome the problem of accessibility one can start with the development process or try to provide a run-time solution. Our concept focuses on the development process. Many critical decisions upon system structure are being already made in the early stages of development [1]. Therefore, an early assessment is very reasonable.

Web-based applications are increasingly gaining importance in our lives for a wider range of users, including those with disabilities. Hence, they must be accessible to all their users. To overcome such problems, a wide range of accessibility guidelines, e.g. WCAG 2.0, for Web applications have been created. Yet, they do not address the development process; a systematic approach for developing accessible Web applications is missing.

Similarly, industrial automation systems have become an inseparable part of our lives, e.g. ticket vending machines. With the phrase *industrial automation systems* we

C. Stephanidis (Ed.): Universal Access in HCI, Part IV, HCII 2011, LNCS 6768, pp. 301–310, 2011.

refer to product automation or technical devices [2]. The big competition and saturated market has forced the producers to try to gain a bigger market fraction by embedding more new functionalities into their industrial automation systems. Hence, usage of such systems is getting more complex. However, often certain user groups are being neglected in development of such systems. Lack of a systematic approach for directing the development and the diversity of industrial automation systems make assessing accessibility to them cumbersome [3, 4]. The existing solutions are very dependent on the underlying technologies. Hence, a generic development solution is still missing.

In the upcoming sections we will provide some basic methods, which are used in the proposed development concept for accessible *user interfaces* (UIs). Section 2 contains the state of the art. In section 3 we will describe our concept and in section 4 a conclusion will bring this paper to the end.

2 State of the Art

User-centered Development (UCD) is a possible approach to embed accessibility into the development process. It involves the end users in the development process. Here the development is done in several iterations, while in each iteration developers are being provided with the feedbacks of the users on using the under development system. The iterative nature of UCD offers parallel treatment of system design, definition of functionalities and UI design. Hence, an appropriate architecture pattern is required to support this parallelization. The basic idea is to separate UI layout and control from system core functionalities. This may be realized by different alternatives, including the separation of front- and backend, a client-server concept, the Model-View-Controller (MVC) architecture pattern, or three-tier architecture.

Furthermore, Model-driven Development (MDD) is among the important methods used in combination with UCD. Typically, UML-based methods are being employed by the developers for the purpose of modeling. However, modeling of accessibility requirements of UI acquires a strong focus on user activities, which is not trivial using standard UML. Furthermore, it must provide a possibility of modeling UI elements abstract enough to be independent of the underlying technology. The existing solutions for modeling UI can be grouped into two categories: 1) practical approaches, e.g. *User Interface Markup Language* (UIML) [5], *User Interface Description Language* (UIDL) [6], or *Extensible Application Markup Language* (XAML) [7] and 2) analytical approaches, e.g. *User Interface Extensible Markup Language* (UsiXML) [8] or the *Unified Modeling Language for Interactive Applications* (UMLi) [9].

Practical approaches are relied on platform-independent development of UIs, using certain development tools. Recent implementations such as Microsoft Silverlight [10] or Adobe Flex [11] provide the separation of UI layout from UI behavior as well as from system core functionalities. Whereas, analytical approaches are more abstract and include a meta-model for all UI aspects, which describes UI structure, components, behavior, navigation, domain, and resources. However, the analytical approaches are still under research and there is lack of tools for actual development. The existing analytical approaches for the modeling of UIs, e.g. useML, UMLi [9], *Task Modeling Language* (TaskML) and *Dialog Modeling Language* (DiaMODL) integrate the requirements of the modeling process including analysis.

Few modeling solutions address complete and systematic integration of accessibility in model-driven design and more importantly in the development process [3, 12].

3 A UCD-Based Concept to Develop Accessible UIs

Our concept combines UCD and MDD, since UCD is focused on user's requirements and modeling allows describing particular aspects and views on the UI as navigation, state or behavior. The combination of UCD and MDD allows a clear analysis and declaration of accessibility onto UI design. Later on, the Abstract Presentation Model (APM) provides a semantic description of UI navigation, views, structure, elements, their roles, states and behavior. Finally, existing development tools can be used to implement the UI. Moreover, our concept considers that MVC architecture pattern for modeling is used; the separation of presentation (view) and functionality (controller) allows focusing on the UI with little interference to the system logic.

The concept addresses UI accessibility problems of industrial automation systems and Web applications, where often some user groups have difficulties interacting with the system, e.g. the ticket vending machines, which are not applicable for those with visual impairment or cell phones. Furthermore, it covers both industrial automation systems and Web applications by defining a generic approach which is applicable to both fields and limiting the target system specifications to guidelines and recommendations.

Industrial automation systems and Web applications may have different accessibility requirements, as far as the technology is related. However, both have accessibility requirements, which rely on similar principles. Based on this principle, the concept tries to provide the developers with guidelines on what factors needs to be considered. For this purpose, a repository of static helps has been included in the concept. It contains e.g. accessibility guidelines. In the following sections different aspects of the proposed concept will be described.

3.1 Accessibility

Accessibility in UI design means free perception and control of all relevant information for all users, also for those with disabilities [13]. Here, the information refers to those being needed by the user to follow his intended workflow constituted by his activities, actions and their relations. Despite the importance of accessibility, it is still a big challenge integrating it to the industrial automation systems, due to numerous reasons. Some reasons are:

- Variety of the requirements of different user groups
- Limiting UI technologies e.g. cell phones with small GUIs
- Difference between developer's mindset with that of the user

The existing accessibility solutions are focused on accessibility guidelines, e.g. on the Web Content Accessibility Guidelines (WCAG) [14]. They mostly aim at runtime behavior of the system, i.e. how the system should work but not how it should be developed. Additionally, they are specifically created for Web applications. As long as human interaction with UI is concerned, industrial automation systems have similar

accessibility requirements as Web applications. Therefore, the guidelines can be adopted for industrial automation systems and UCD is a promising approach for the UI development of industrial automation systems, since both systems require similar accessibility requirements, regarding their UI. Moreover, industrial automation systems and Web applications are being often used in combination. Projects like W3C Mobile Web Best Practices 1.0 [15] have used such guidelines for industrial automation systems, e.g. to overcome the accessibility problems of the elderly using cell phones. In that example more than 70% of the guidelines could be employed for the industrial automation system [15]. Hence, our concept uses WCAG as the basis for addressing the critical points regarding the accessibility.

3.2 Multimodality

Multimodal interaction offers users to use various interaction channels in interfacing with a system. It uses different senses of human being, e.g. hearing or sight, for input and output of data. A simple example for a multimodal interaction can occur in a navigation system in a car, which can be interfaced via touchpad or voice input. The goal of a multimodal interaction is to use *optimally* the users' capabilities, by considering extra senses of the human users in the human-computer interaction (HCI) [16]. In comparison to traditional interfaces (single-modal) multimodal interaction allows users to communicate more naturally and interface with complex information with more freedom of expression [17]. In [16] multimodality is considered as a combination of interface styles, which can be used to improve the individual components in isolation so that the combination works better. However, this has to be differentiated from instantiating several copies of a UI in systems like in Mac systems. The proposed concept uses the approach used in [16] for parameterization of the interaction channel.

3.3 Methodology

The proposed concept consists of two parts; the *Generic Parallel Process* (GPP) and the *Accessibility-supportive Repository* (ASR) (see Fig. 1). GPP is a supportive process running in parallel to the typical development process. It helps developers to identify critical spots that are relevant to accessibility, during the analysis and design phase, as these two phases shape the basis of the under development system. GPP, then, provides the developers with tools and guidelines to take actions and assess the identified accessibility requirements. The identification and assessment of accessibility requirements are being supported by ASR, which is a predefined repository containing reusable components such as accessibility guidelines, design patterns and norms, which are generally necessary for an accessible UI.

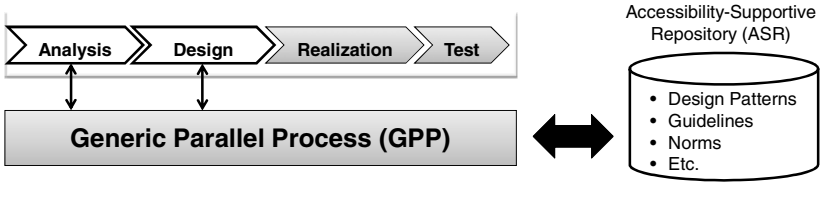

Fig. 1. An overview on the proposed concept

3.4 Generic Parallel Process in Analysis Phase

In the analysis phase the requirements of the system are being collected. While in the traditional approach, a great focus on functional requirements is given, GPP tries to help developers identify accessibility requirements (none functional requirements). Besides methods like *appreciative inquiry* or *structure laying technique*, GPP investigates sight, hearing, motoric capabilities and knowledge (the 6th sense) of a user in the interaction with the systems [16]. The aforementioned aspects can be affected in two ways: (1.) by user impediments, e.g. visual impairment, and/or (2.) external constraints, e.g. voice input in a noisy environment. Considering these aspects with the two affecting sources, the accessibility requirements can be analyzed. For example, Table 1 shows interaction aspects and constraints for a coffee dispenser and its process of maintenance.

Table 1. Speculations of user impediments and his/her external constraints in the example of coffee dispenser

	Sight	Hearing	Motoric	Knowledge
User impediments	Very good	Very good	Very good	Technician
external constraints	High light intensity	Very Noisy Environment	Back and front side of the machine must be accessed. Access at the same time needed.	n. a.

Out of the information provided in Table 1, following accessibility requirements can be concluded.

1. Visual interaction with the system is possible.
2. Technical staff functionalities are required.
3. High contrast of the UI elements is required.
4. Voice input and output is not possible for the user.
5. Multimodal interaction is needed.

Another important step is to model with their correlations and conflicts. This will help identifying requirements that need to be eliminated or might be added to the requirements set. Currently, the concept assumes that this step of analyzing requirements is to be done by project managers or similar authorities.

For the purpose of modeling requirements, the *requirements graph* method presented in [1] is used. This method is very suitable for the purpose of modeling none-functional requirements [1]. It arranges requirements as graph nodes, connected to each other by labeled arcs. The labels are “hurt”, “help”, “make”, or “brake”, with “Hurt” referring to a conflict, “Make” to dependency, “Help” to recommendation, and “Hurt” to discouraging relation between the requirement nodes. The intensity of each relation can be defined using an intensity value from the set $I=\{--, -, 0, +, ++\}$, where “--” is the strongest negative and “++” is the most positive intensity value [1].

3.5 Generic Parallel Process in Design Phase

In the design phase, the concept supports the developers to design the UI in parallel to the design of the system logic. It helps identifying critical points in the interaction that affect the accessibility. In this phase a scenario-based approach is being employed for designing the structure of the system. Once the scenarios are defined, all cases of the scenario are being modeled. The proposed concept employs a UML-based approach, since it is universally used, hence it can be easily used in parallel to designing the system logic. Furthermore, it is easily extendable using UML profiles. Focusing on the *architecture pattern*, *workflow*, and *relevant information* on the end users, three aspects of the interaction with UI is being modeled: task modeling, dialogue modeling, and presentation modeling. Despite existing methods, which are very much syntax based, the concept uses guidelines aimed at the aforementioned criteria for modeling the three aspects of interaction. The syntactical complications of modeling UI are being removed semantically. Fig. 2 depicts the important guidelines for each criterion, used in the concept.

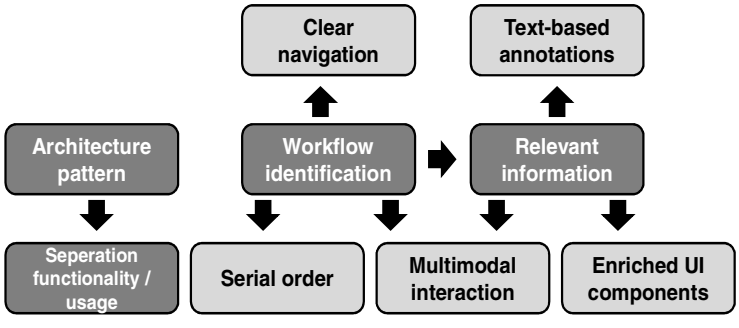

Fig. 2. The important guidelines for each criterion

As it can be seen in Fig. 2, the knowledge about user's workflow is necessary to determine the relevant information on the end user, since it is only being considered in the usage context. The relevant information is being stored using a text-based annotation such as XML. The separation of data model/core-functionality and UI-presentation/control enables multimodal concepts for usage and different layouts for particular input/output devices. Using a well-suited architecture pattern, e.g. MVC, this separation can be easily achieved. Furthermore, deployment of multimodal interactions is tightly dependent on knowing the workflow and the relevant information. Design of interface components in a serial order – corresponding to user's workflow – to support Assistive Technology (AT) such as screen readers and structuring the interface navigation distinctly and avoiding deep hierarchical structures, in order to support sitemaps and breadcrumbs for instance are further guidelines used to model UI. In Table 2 the sequence of UML diagrams, which should be implemented in each of the three aspects of interaction with UI are listed. The diagrams in each row should be considered together, while modeling.

Table 2. Overview of the UML diagrams in the GPP (AD – UML activity diagram, UD – use case diagram, CD – class diagram, SC – state chart)

Task Modeling Diagrams	Dialog Modeling Diagrams	Presentation Modeling Diagrams
Use case – UD		
Activities – AD	Modes, macro-navigation – AD	Views, main & utility navigation, site-map – CD
Actions – AD	Interactors, Roles of UI components – CD	UI elements, attributes & behavior – CD
Work objects (UI data) – AD		
Workflow – AD	Micro-navigation (Sequences, alternatives), UI behavior – SC	Order of UI elements – CD

3.6 Accessibility-Supportive Repository

ASR feeds the GPP with design patterns, guidelines, norms and standards, which have been collected. Based on existing patterns, the concept includes a pattern structure as it can be seen in Fig. 3. The design pattern contains information on the problem, usage context and solution, illustration, reference to other patterns or code/design idioms. Using the patterns developers can design each step more easily. The design patterns are based on a selection/adoption of WCAG 2.0 accessibility guidelines [14]. The patterns can be categorized into four categories: perceivable, operable, understandable and robust; each addressing a certain aspect of accessibility [14]. Fig. 3 illustrates the various categories of the accessibility patterns and their connection to diagrams.

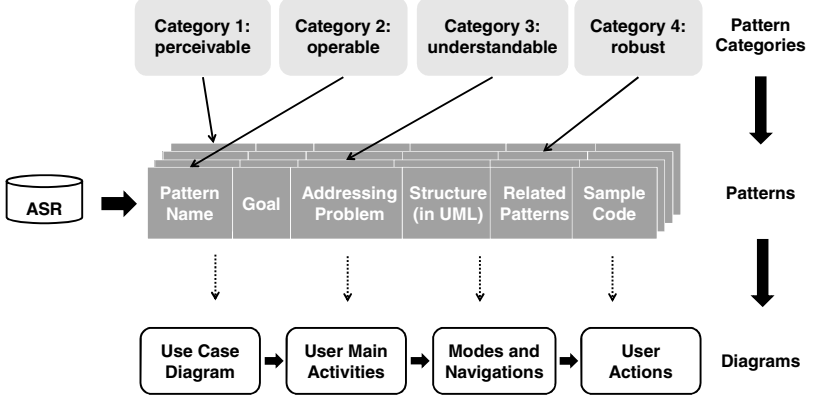

Fig. 3. Various categories of the accessibility patterns, their instances and derivation of design diagrams

The UI of a coffee dispenser as a case study shows examples of the GPP analysis and design phase. Fig. 4 shows the first step in analysis – the identification of user modes. Based on the use cases, main activities are detected. In the example of Fig. 4 three main activities “Change product”, “View product”, and “Search product” are detected. Correspondingly, the user workflow is grouped by modes of usage.

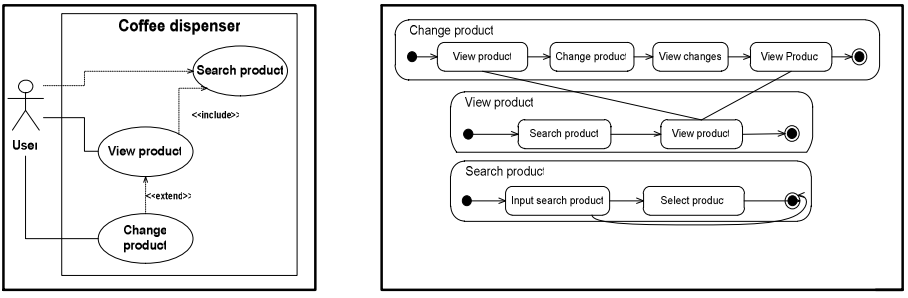

Fig. 4. Use cases and user main activities

Fig. 5 shows the main navigation network of modes drawn in dashed boxes. The relation of each node describes the possibilities of moving from one mode to another one. Here the technological description is not of interest. The selection of a drink, in this example, might be a simple read through the buttons on a coffee dispenser or a sound module reading out the available drinks.

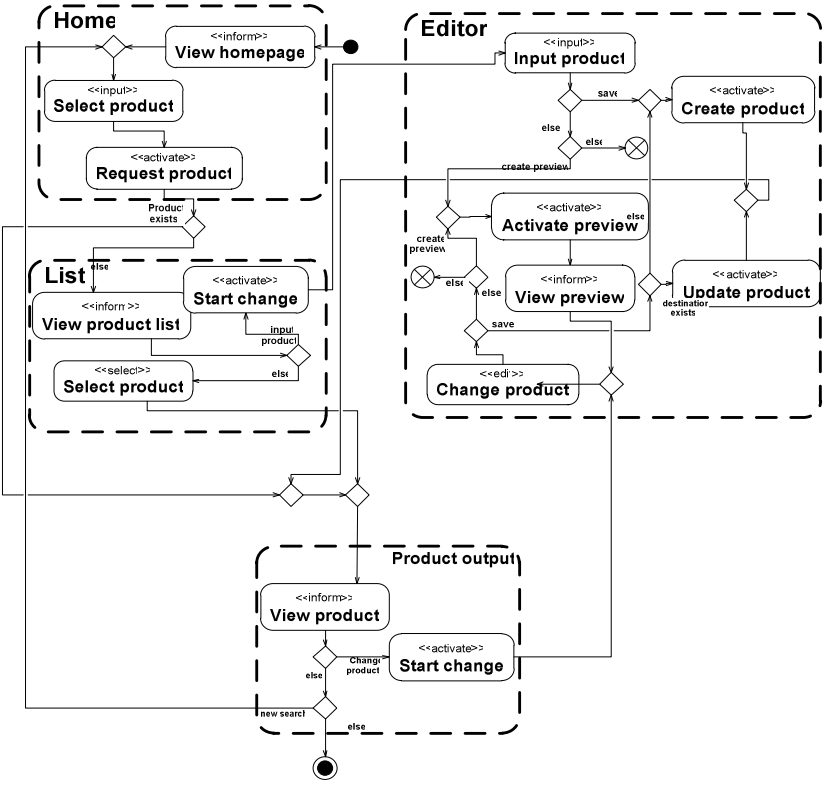

Fig. 5. UI main navigation modes and user's actions within each mode in the example of coffee dispenser

Furthermore, each navigation mode contains several actions. For instance, once the user has decided upon selecting his drink, he might move to the corresponding navigation page. Then he can take several steps, i.e. actually selecting his desired drink, reviewing his selection or canceling it. Here, each mode itself is being refined by internal actions. The actions are based on atomic action categorization described in [18]. The actions are defined as five stereotypes *select*, *activate*, *inform*, *input*, and *edit*. In the concept the two steps of defining mode navigation and user actions are being done using separate diagrams. In this paper, for more compactness we have shown both in one diagram, in Fig. 5.

4 Conclusion and Outlook

The proposed concept is based on the survey made on the state of the art and consists of a set of suitable methods. It provides a parallel process to the development process, while considering the end users.

We have discussed the potential of UCD and MDE in maintaining accessibility in developing UIs. Having focused on the early development stages, i.e. analysis and design phase, we have investigated suitable methods and technologies, which can assist developers by designing an accessible UI for industrial automation systems and Web applications.

The future work of our project is focused on an automated identification and analysis of the requirements, a systematic bridge between analysis and design phase, and an automated transformation of the detected diagrams.

Acknowledgements. We highly appreciate Deutsche Forschungsgemeinschaft (DFG) for funding this project.

References

1. Cysneiros, L.M., Werneck, V., Kushniruk, A.: Reusable Knowledge for Satisficing Usability Requirements. In: 13th IEEE International Conference on Requirements Engineering, pp. 463–464 (2005)
2. Lauber, R., Göhner, P.: Prozessautomatisierung I, 3rd edn. Springer, Heidelberg (1999)
3. Nieminen, M.: Information Support for User-oriented Development Organisation, Dissertation for the degree of Doctor of Science in Technology, Helsinki University of Technology Department of Computer Science and Engineering (2004)
4. Göhner, P., et al.: Integrated Accessibility Models of User Interfaces for IT and Automation Systems. In: Proceedings of the 21st International Conference on Computer Applications in Industry and Engineering (2008)
5. UIML community. User Interface Markup Language (UIML) (2009), `http://www.uiml.org/`
6. UIDL community. User Interface Description Language (UIDL) (2008), `http://www.uidl.net/`
7. Microsoft Corp. Extensible Application Markup Language (XAML), `http://msdn.microsoft.com/en-us/library/ms747122.aspx`

8. UsiXML community. User Interface Extensible Markup Language (UsiXML) (2010), http://www.itea.defimedia.be/
9. Da Silva, P.P.: Object Modelling of Interactive Systems: The UMLi Approach. University of Manchester (2002)
10. MacDonald, M.: Silverlight 2 Visual Essentials. Apress (2008)
11. Tapper, J., Boles, M., Labriola, M., Talbot, J.: Adobe FLEX 3. Addison-Wesley, Reading (2008)
12. Jeschke, S., Pfeiffer, O., Vieritz, H.: Developing Accessible Applications with User- Centered Architecture. In: ICIS 2008, Seventh IEEE/ACIS International Conference on Computer and Information Science, pp. 684–689. IEEE Computer Society, Los Alamitos (2008)
13. ISO13407:1999, Human-centred design processes for interactive systems (1999)
14. World Wide Web Consortium (W3C). Web Content Accessibility Guidelines 2.0 (2008), http://www.w3.org/TR/WCAG20/
15. World Wide Web Consortium (W3C). Mobile Web Best Practices 1.0, http://www.w3.org/TR/mobile-bp/
16. Stiedl, T.: Multimodal interaction with industrial automation systems. Dissertation at IAS, University of Stuttgart (2009)
17. Taib, R., Ruiz, N.: Integrating semantics into multimodal interaction patterns. In: Popescu-Belis, A., Renals, S., Bourlard, H. (eds.) MLMI 2007. LNCS, vol. 4892, pp. 96–107. Springer, Heidelberg (2008)
18. Reuther, A.: useML – Systematische Entwicklung von Maschinenbediensystemen mit XML, Universität Kaiserslautern (2003)