Univ Access Inf Soc (2008) 7:117–128
DOI 10.1007/s10209-007-0108-6

LONG PAPER

Accessible design and testing in the application development process: considerations for an integrated approach

Gottfried Zimmermann · Gregg Vanderheiden

Published online: 9 November 2007

Abstract Accessible design principles should permeate virtually all phases of the application development cycle, using existing "best practices of software engineering" for accessibility purposes. This paper proposes a methodology for accessible design and testing that includes proven tools of software engineering, namely use cases and scenarios, to capture functional requirements. Guidelines developed through user testing and heuristics are made real using personas to exemplify accessibility requirements, reflecting a diversity of user capabilities and use contexts. For implementation and testing, test cases containing accessibility checkpoints are generated, based on the guidelines. Complementary to this methodology, expert reviews and user testing should be conducted for evaluation of the developed products and further refinement of the development process.

Keywords Accessible design · Personas · Testing · Universal design · Use cases

1 Introduction

It is commonly recognized that mainstream products should be developed so that they are accessible to people with disabilities and older users by following universal design principles.

Lawmakers are also pushing for making hardware and software products and websites accessible to people with disabilities. In the United States, section 508 of the Rehabilitation Act mandates all federal agencies to purchase only accessible products, and to provide information on the Web in a barrier-free manner. Similar regulations are under consideration, or have been enacted, in other countries, including Australia and many European countries.

In general, accessibility regulations rely on a common and standardized view of what makes a software product or a website accessible. The Trace Center of the University of Wisconsin and the World Wide Web Consortium (W3C) have been pioneers in developing guidelines for accessible design in the software and Web domains [3, 24]. Today, accessibility guidelines exist for the major operating systems and platforms, for software in general [9, 10], for the Web, for e-learning applications, for public terminals, for telecommunications, and other domains. For a list of accessibility standards and guidelines, refer to [26].

This paper proposes a methodology that draws from the work on accessibility guidelines, combining them with existing tools in software development, thus providing a model that integrates the implementation of accessibility in mainstream products within existing product development practice.

Guidelines need to be built into today's development environments to impact the products that are being developed in these environments. Application development is typically happening under time and budget constraints, and there is rarely time for designers and developers to go through long lists of accessibility criteria. It is important for accessible design to be integrated into the software engineering process so tight, that its manifestation mingles with

G. Zimmermann (✉)
Access Technologies Group, Wilhelm-Blos-Str. 8,
72793 Pfullingen, Germany
e-mail: gzimmermann@acm.org

G. Vanderheiden
Trace R&D Center, University of Wisconsin-Madison,
1550 Engineering Dr., 2107 Engineering Centers Bldg,
Madison, WI 53706-1609, USA

other concepts and methods to the degree that there is no difference in handling, whether an activity is motivated by accessibility concerns or for any other reason. In effect, this paper proposes to extend the development process in order to enhance the usability of the products so that they naturally accommodate and are usable by people with a wide range of abilities and in a wide range of environments.

It is important that the development of accessible applications be supported in an automated fashion as much as possible. Tools are needed that automatically assess the accessibility of a product in general, and with regard to specific user groups, and provide process-integrated and constructive guidance to the developer on how to apply accessibility principles.

Accessibility considerations need to be included into the development process from the beginning, and not as a patch that can be applied to a product after it is complete. In fact, if accessibility is added on to a Web application after development, costs are about 10 times more than if it is built in from beginning [23]. As a consequence, accessibility has the notion of being unaffordable, and is widely assumed to harm the competitiveness of a product.

To change this situation, it is necessary to make accessible design principles understandable and operable for mainstream application development, and to impact the design of the development tools and environments that are widely used by the software and Web industry. In such a context, accessibility requirements are treated as ordinary requirements, so that developers find them in their task descriptions together with functional and performance requirements. In this integrated approach, the developers do nothing special to make a product usable to those with disabilities—they just do their ordinary job of developing software to meet the specified requirements.

Software and Web application development is not a new discipline. Although there are many projects that have failed, there are well-known concepts that have proven successful in dealing with the complexity of application development. Therefore, the goal of this work is to use "best practices of software engineering" as a vehicle for incorporating accessible design principles into the mainstream application development. By building on existing process frameworks, designers and implementers can more easily address accessibility related issues, rather than having to learn and adopt fundamentally new tools and processes.

The remainder of this paper describes a proposed methodology or process model for accessible design and testing, combining existing software development best practices in a new way for an integrated approach:

1. Capturing accessibility requirements in a way that makes them tangible and comprehensible, through use cases and personas.
2. Making accessibility requirements concrete through scenarios and guidelines for accessible design. (In this paper, the term "guidelines" is used to mean accessibility standards and guidelines that contain interoperability techniques and heuristics for accessible design).
3. Manual and automatic testing based on test cases and accessibility checkpoints that are derived from guidelines.
4. Complementary user testing and expert reviews, thus evaluating intermediate and end results, and continuously improving the overall process model.

The proposed approach is based on experience of the authors with over 50 companies. It is summarized here in order to facilitate discussion and trial within corporations interested in exploring ways to incorporate accessibility directly into their processes. Since every company is different (and often even different from division to division), no company will be able to incorporate the process directly as described. Additionally, any implementations by one company may have limited relevance to successful implementation in another. The concepts are presented here therefore, not so much as a roadmap, but to facilitate thinking and exploration within companies interested in long-term evolution of their core design processes to include accessibility, including design for an aging population.

In this paper, the phrase "application development" is used as a term for the development of software in general, and the development of Web sites and Web applications. Also, "product" is used to refer to the outcome of the application development process, which may be any software application or Web-based application.

2 Related work

Use cases were introduced by Ivar Jacobson in 1992 [11]. A use case is a sequence of actions a system performs that yields an observable result of value to a particular actor [12]. Use cases focus on what a system should do rather than how it should do it. They act as a common language for communication between the customers or users and the system developers. Therefore, they can be used to capture requirements very early in a development project. They are the drivers of an incremental application development and the threads that hold the various development activities together throughout a product's lifecycle.

Even before being dubbed "personas", various techniques of user profiling and the creation of user archetypes have been used in software development for a long time, under various names (for an example, see [7]).

In 1999, Alan Cooper introduced the concept of "persona" [4]. A persona is a realistic description of a user of a system, capturing their goals and needs when using the system ("goal-directed design"). The specification of a persona gives a system's fictitious user a name and a face (photo), and typically contains a rough description of a day in the life of that persona. Though described as a user with a real life, a persona typically reflects a group of users rather than an existing individual.

Personas have been used in user-centered and participatory design. By the means of personas, developers are enabled to look upon their system through the eyes of potential users, with their specific goals and needs. This is extremely helpful if an application is to be developed that will be used by a variety of users with varying needs and preferences (cf. [18]).

Four years after the introduction of personas, Dayton reports [6]: "Storytelling through personas and scenarios is becoming a widely implemented practice in the design of large websites and other interactive technology products" (p. 332). Dayton examines how personas and scenarios have been used in user and task analysis, and identifies three main aspects: user research, empathetic identification with personas and their goals, and information and interaction design.

Randolph suggests the use of personas in smaller software development projects, to get an early start in thinking about user requirements [18]. Dantin reports on a study of how Randolph's approach would have benefited the user interface design of two small-scale educational applications [5]. In this study, the meaning of Nielsen's usability heuristics was clarified by attaching them to personas. The methodology proposed in this paper takes a similar approach—tying guidelines to personas for the sake of making it easier for designers to understand and apply them.

Creators of personas have followed different approaches and principles in application development projects. In an interesting study [22], Shyba and Tam compare the development of personas with the process of an actor identifying itself with a role in a play.

Among many companies that have taken the persona approach seriously in software development, Microsoft has adopted and extended the persona idea and made it part of their software development process. Pruitt and Grudin [17] describe how personas were used in two projects of different size at Microsoft, namely the development of the MSN Explorer, and the development of Windows. The use of personas was extended beyond the domain of designers and their clients, to include developers, testers, writers, managers, marketers, and others. Although not aiming for disabled personas (the 6 personas that were created for the development of Windows would hardly be sufficient to reflect the various kinds of disabilities), their descriptions included information on accessibility. Pruitt and Grudin argue that personas are more engaging than design based primarily on scenarios, and underline that with insights from psychological theory.

In consumer electronics, requirements engineering is particularly challenging, because of the anonymity of users and their large variety of preferences and needs. Aoyama [2] reports about using personas and scenarios for effective capture of user requirements for embedded software in mobile phones, using a combination of conjoint analysis and requirements engineering. The development of software based on a universal design approach has to cope with similar issues. However, in contrast to the approach promoted by [2], in the context of universal design user requirements that have only little user representation (called "marginal services" by Aoyama), should not be eliminated, since this would defeat the purpose of developing applications that are accessible for everybody.

Personas have captured user requirements in the development of domestic ubiquitous computing applications. Schmidt and Terrenghi [21] report about a study that brought about design guidelines for the creation of novel display artifacts for home environments, using a scenarios-based participatory design process.

Personas have also been used for modeling users of educative software. A process involving personas and scenarios for the development of interactive online teaching systems is introduced by Yu and Liu [25]. That process is similar to the methodology described in this paper, although its notion of scenarios is somewhat broader.

A more critical view on personas is taken by Rönkkö [19]. In three studies, Rönkkö claims that personas have played no or only a little role in user interface design, compared to other design influences. However, he admits that in one project the concept of personas was useful as a political instrument.

In some cases (that are of particular interest here), personas have been used in application development to reflect the requirements of a particular user group that would otherwise run the risk of being overlooked or not being sufficiently addressed.

Personas have been used to model children as a targeted user group, and adaptations to the original persona concept have been made to reflect the specific needs of children. Antle [1] describes a theoretical and empirical framework for creating and using user abstractions of children (child-personas), thus mostly eliminating a designer's assumptions about children.

Personas can also guide designers in understanding the needs of older people and people with disabilities. Mueller [14, 15] describes this approach (using personas with disabilities in the design process) as a tool to help

designers understand how users of all ages and abilities might react to their designs. His work has led to a set of personas developed at the Wireless RERC (Rehabilitation Engineering Research Center on Mobile Wireless Technologies) in Atlanta, GA. These personas are used to describe how people with disabilities use their cellular phones and smart phones.

The methodology proposed in this paper follows the use of personas for the sake of accessible design, following the universal design approach. Within countries of the European Community and beyond, universal design has been promoted as a promising approach for a sustainable improvement on people's access to products and services in information and communication technologies (ICT). In addition to reaching new customers who have been excluded from of the market previously, universal design has been shown to make products more attractive for many users [13].

3 Capturing accessibility requirements

When designing a product, it is necessary to know the users addressed by the design. Although accessibility is sometimes described as "usability for all users", some aspects of accessibility go beyond usability in its traditional interpretation, and involve usability by people with different sets of abilities and personal tools. A substantial aspect of accessibility is interoperability with assistive technology, and hence the requirement of conformance to established standards.

Accessibility regulations and policies will in many cases require that a product has to be accessible to users with a very broad range of functional limitations, including individuals who are older and often have multiple functional limitations. Typically, a product will have to be usable by almost any type of user, possibly with the exception of some age-based user groups. For example, an online banking system will not ordinarily be used by a 6-year old child.

Often, accessibility is thought of as a software quality that is inherent to a product or not. However, in a strict sense, accessibility is not a binary attribute that is true or false, but a relation between a product and a user in a particular context of use. Therefore, the question to be asked should not be: "Do we want to make this product accessible?", but rather: "What is the range of users of this product?", "What are their abilities and limitations?" and: "Under what circumstances will they use this product?" Based on these questions and their answers, the range of personal, environmental, and task related factors that should be addressed when identifying the requirements for a product should be clear.

At the beginning of a software development project, the first activity is to think about the envisioned product and its users [16]. In doing so, it is important that the full range of users be included, not just the typical or stereotypical users.

An online banking application is discussed here as an example. Online banking addresses a variety of people, men and women, old and young, English and Spanish speaking, computer programmers and people who are not comfortable with technology. The online banking application has to be simple to use—people will not bother reading a manual when doing their banking business. A significant user group is the group of elderly people, including those who are too frail to go to the bank building. Elderly users will often experience fear in dealing with technology. They may have difficulties when reading small text or not be able to see at all. Also, some users may not be able to use a mouse because of limited manual dexterity, including users of all ages with physical disabilities.

The primary use context of the online banking application will be the home environment. It will run on home computers that are connected to the Internet. Some users, however, may want to disconnect while working on their wire transfer order because they use a dial-up modem on a single phone line and want to make a phone call in the process of editing the electronic form. Only by taking these and other considerations on the target users and the contexts of use into account, the product can be successful in its goal of serving more customers at lower cost to the bank.

Based on the target users, their needs and constraints, important functional (i.e., what the product does) and non-functional (e.g., interface) requirements can be identified before the design has begun. In the online banking example, images should be used only if they are easy to recognize, even for users who have poor vision. Design constraints may also impact the platform the application can run on. For example, since the product is intended to be accessible by people who are blind, at least one screen reader should be available on the targeted platforms and work well with the application. Alternatively, the product could be designed to provide audio output by itself where this is practical and effective. The earlier these requirements are identified in the development process, the better.

The above are not intended to argue against an iterative development process in which requirements are added, modified or dropped during the lifecycle if indicated by technical, time, budget or other reasons. An example of such an iterative development process model is the Rational Unified Process [12]. However, even in an iterative approach, a major goal is to define an initial set of requirements (which includes use cases) that is as complete as possible, early in the process so that it can constitute the basis for further iteration planning and risk management.

At any point in the process, the set of requirements should reflect the current knowledge of the envisioned product.

Once accessibility requirements (including any legal requirements that aim at making a product accessible to people with disabilities) are captured, they should be treated equally to other requirements. Thus, accessibility for the entire target user base (an interface requirement) should be integrated in the overall quality of the product, in the same way that the product functional description (functional requirements) contributes to its quality. Consequently, it is a product defect if the product fails to be accessible to any one of its targeted users.

Accessibility requirements should be based on the targeted users, and the question arises of how these requirements can be used to drive the development process. Clearly, creating a long list of requirements that would cover all types, degrees and combinations of disability (requirements that the designers and developers have to go through manually) will not be practical for most projects. To make accessibility requirements effective in product development, they have to be captured in a way that is tangible, concrete and reasonable in number for the members of a project team. As described in the following sections, personas and guidelines, embedded in the context of use cases and scenarios, are suitable tools for doing this.

3.1 Use cases

In mainstream software engineering, use cases have been successfully employed in numerous projects to specify the external behavior of an application. Use cases make functional requirements operable and graspable for all stakeholders in a development project (see Fig. 1).

From the use case perspective, the user of a system is an "actor", whose user interface needs and abilities are only implicitly captured, if at all. While use cases are an excellent means to describe the functionality and external behavior of a system, they can not adequately convey information on the real users, their user interface needs and usage environments. The next section discusses how personas can fill this gap.

3.2 Use of personas

One of the most common obstacles for user-centered design is that designers of a product have little knowledge of how real users will use the product. This is especially true if the designers are young and technically versed (which is true for almost all projects making use of the latest technology), and are to design a product that will be used by older people, among others. Also, most designers have no idea about how people with disabilities use today's technologies.

In the proposed model, personas are used for describing the interface needs and other usage requirements of users with disabilities, when developing accessible applications. Personas should be described as sympathetic users, as people with individual feelings and struggles in life [17]. As they work with these personas over a period of time, developers will become more and more familiar with the personas and even empathize with them. As a result, the envisioned users with disabilities that are represented by the personas become tangible and ubiquitous in the development process.

The description of a persona will typically include the use environment, including any assistive technology that is employed by the persona to use the product. A detailed persona description is instrumental in determining the set of accessibility guidelines and standards that need to be applied to the products to make it accessible to the particular persona. Sometimes this means that one user group has to be represented as multiple personas, to reflect a range of different use contexts and assistive technologies that they are using. For example, the range of users who are blind would include those who can use Braille and those who can not.

Cooper distinguishes between primary persona and secondary personas. The primary persona represents the primarily targeted user, and is the main driver for designing the application's user interface [4]. All other personas are secondary personas. They have additional needs that the user interface design needs to accommodate. Of course, the resulting user interface must not conflict with the interface needs of the primary persona, or with the interface needs of other secondary personas.

In the proposed model for accessible design and testing, one set of personas is associated with every use case that involves a human actor (see Fig. 2). In each set of personas, the most prominent user type determines the primary persona. For a disability related product, the primary

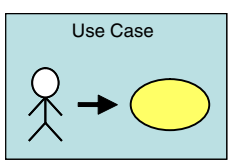

Fig. 1 Use case

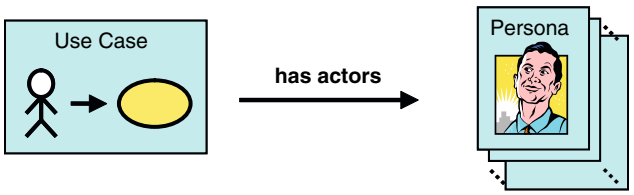

Fig. 2 Use case and personas

persona would be the primary disabled user type. For a mainstream product, it would be the most prominent mainstream user type. The other personas (both mainstream and disability related) are secondary personas. Both primary and secondary personas drive the user interface design for the use case. Secondary personas (both mainstream and disability related) are used to express additional requirements to the primary persona. This process creates a user interface for every use case.

When looking at a use case that involves a user, the user is virtually replaced with each one of the personas from the set of associated personas, thus amending the user interface design, and making sure that the user interface meets the needs of all secondary personas.

In unusual cases, a secondary persona may have a conflicting (rather than different) user interface need from the primary persona, which can not be met with a flexible user interface design. In this case, both personas are considered as primary, the use case is duplicated, and the personas are distributed among the two so that each use case has exactly one primary persona and non-conflicting secondary personas. However, this does not result in two products, but rather in alternate modes of operation for a product.

It should be noted that personas alone are not a substitute for specific accessibility requirements. However, as discussed later on in this paper, when used in conjunction with specific accessibility guidelines, personas can give meaning and context to them. Personas can make the requirements and guidelines easier to understand and more compelling to developers [5]. Over time, they can help designers internalize the requirements and guidelines as part of their experience and knowledge base, rather than having to memorize them or follow them superstitiously.[1] Superstitious conformance to accessibility guidelines usually results in designs that meet specifications but are not necessarily optimal or even operational by intended users.

In summary, personas are used as illustrations for accessibility requirements at design, as well as for linking to accessibility guidelines which generate checkpoints for conformance testing. As an additional benefit, personas are also useful when developing user support that accommodates users with different abilities (see next subsection).

3.3 How personas can help assist the users

After a product is developed, support must be available for the users using the deployed product. Similar to the development of a product, the set of personas should drive the support mechanisms for the "real users" after the product has been deployed.

Any documentation to the product, in any form, should be developed with the personas in mind to make it usable to the full range of intended users. Where documents are handed out in printed form (e.g., user's guide, installation guide, course and training material), it might be necessary to additionally make them available in an accessible electronic form. Any electronic documents (such as online help, release notes, Web pages for download) should be checked for accessibility to the targeted user groups. If the product is to be sold as a physical package, issues of packaging may be relevant as well. Also, for tutorial events, appropriate rooms have to be chosen to be accessible.

To provide instant support for the users, a hotline may be established. If so, its accessibility to the targeted users has to be checked. For example, a separate hotline for text telephone users may have to be installed, and its number included in the documentation.

4 Making accessibility requirements concrete

Use cases and personas make accessibility requirements comprehensible for the human reader and designer. However, they need to be translated into constructs that are more concrete and operable in the context of a software development process that involves tools and automatic tests. As described below, scenarios and guidelines are suitable tools for this.

4.1 Scenarios

Scenarios are useful to illustrate use cases in more details. While use cases capture a generalized view of a user and a task, scenarios describe a specific instance of a use case in terms of a concrete workflow with specific data, specific events and possibly a specific user interface, sometimes outlined on a storyboard (see Fig. 3). In a development project, one would typically devise a few scenarios per use case to illustrate the typical flow of events and some of its error conditions. Scenarios are based on real data, which

[1] Superstitious behavior occurs whenever someone does something in a certain way that has no correlation to facts—just because it worked that way before. For example, someone who always turns their phone off and back on at the end of a call because once they were unable to hang up without turning the phone off. Or someone who always deletes and reenters information into a new record rather than just editing the old record. Or someone that designs everything with a certain spacing because that was the rule of thumb they were given and they do not know what the basis was for deciding that spacing, so they just continue to do what they used to do (even if it is not necessary anymore—and even if it is not sufficient or correct anymore).

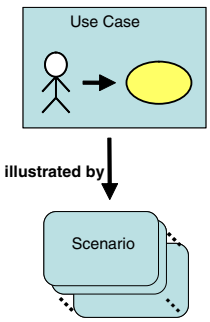

Fig. 3 A use case is illustrated by a set of scenarios

makes them suitable as a basis for test cases later on in the process.

4.2 Accessibility guidelines

In the proposed model, otherwise abstract accessibility requirements are made concrete and understandable, and express them in the form of personas and accessibility guidelines that can be applied to the development of the product. Importantly, every guideline is based on and tied to a subset of personas. This means that the guideline has to be met to serve any one of the linked personas. By linking the guidelines to personas, the guidelines become understandable in the context of application development and for the development team (see Fig. 4).

Guidelines are also related to scenarios. Every scenario needs to conform to the guidelines that are linked to the personas of the corresponding use case. The link between scenarios and guidelines forms the basis for testing conformance, as discussed later.

Personas help define the set of guidelines that are identified as accessibility requirements for the scenarios. It is important that the full set of personas be considered when designing the user interface, with each persona adding accessibility and usability requirements to the scenario.

For example, if one persona (this may be the primary persona) uses a mouse and keyboard in a visual interaction style, another persona (a secondary persona) that uses a screen reader and keyboard navigation would add the following requirements to the scenario: Make sure that all elements are exposed through the accessibility API of the pertinent platform, that images have textual equivalents, that all elements can be accessed through the keyboard, etc. Additionally, considering yet another persona with hearing impairment, one would have to make sure that audio output can be adjusted in volume, that there are subtitles available for video clips, etc. If a persona has a language or learning disability (common but often invisible), special attention should be paid to the product's ease of use, the reading level of the online help texts should be assessed, etc.

The success of this method hinges on the selection of the personas and their descriptions, and a comprehensive catalogue of guidelines linked with the personas. The set of personas should reflect the broad variety of targeted users, in particular users with disabilities, including multiple disabilities or functional limitations as is common with people who are older.

Unfortunately, most existing guidelines are not explicitly linked to certain types of disabilities (or even personas). In general, more research is needed to identify practical and reliable approaches for this issue. Also, testing is important to prove the effectiveness of particular accessibility guidelines for particular user groups.

The number of distinct categories of users with disabilities (different types, degrees, combinations, onset, etc.) that have different user interface needs has been calculated at over 100 [20]. Although a smaller set of personas might be used to represent them, it would still need to be significant in size and many times larger than is used for mainstream users. Using a set of personas that does not represent the diversity of limitation, onset, skills, etc., is very dangerous, yet common. It can lead to a serious misunderstanding of many disabilities and can result in much careful design effort inadvertently missing important user groups' needs—including groups of aging users.

When used together, use cases, personas, scenarios and guidelines form a powerful tool for user centered development (see Fig. 5). Use cases are employed to capture the

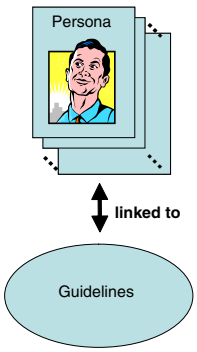

Fig. 4 Personas and guidelines are linked

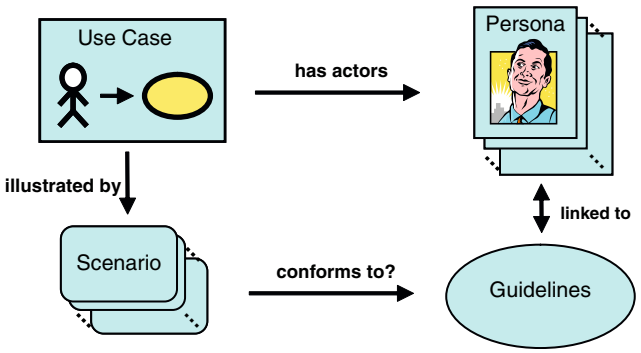

Fig. 5 Use case, personas, scenarios and guidelines

overall functional behavior of a product. Personas express accessibility requirements by specifying a (diverse) set of target users for a use case. Scenarios, linked to specific use cases, describe a real-life sequence of events of a "real" user. Finally, guidelines, linked to personas, guide the developer in making the product accessible and usable for the target users.

5 Testing for accessibility

Testing activities in application development should be spread over the complete lifecycle, as a continuous approach to quality. Early testing can significantly lower the cost of completing and maintaining the software. Tests are derived from requirements, but will also draw from other sources, for example when testing against specific error conditions. Productivity tools are essential for generating test data, and for running and analyzing tests.

5.1 Test cases

Heumann describes a method to generate test cases from use cases and pertinent scenarios [8]. For every use case, a set of scenarios is identified so that one scenario stands for each possible execution path through the main and alternate flow of events of a use case. Then one or more test cases are derived for each scenario, and data values are attached to the test cases (see Fig. 6).

5.2 Accessibility checkpoints

Test cases are useful for evaluating a product against the functional requirements of a product. To evaluate the product against the accessibility requirements, it is necessary to make sure that the user interfaces are accessible to the target users. The test cases need to contain accessibility checkpoints, reflecting the guidelines and their associated personas (see Fig. 7).

An accessibility checkpoint is an atomic checkpoint related to an accessibility guideline. It is specific to a particular platform and can be tested either in an automated or manual fashion. One guideline may have multiple related checkpoints.

Accessibility checkpoints are derived based on guidelines, personas and their contexts of use. For making the proposed model feasible, a seamless connection between personas, guidelines and accessibility test tools needs to be achieved. The model raises specific requirements on the guidelines which the checkpoints are drawn from.

First, the guidelines should identify for each of their checkpoints how it affects the usability for different types of users (personas). The benefit to a user can be described in three degrees: essential, important, and beneficial. If a checkpoint is essential for a particular persona, the persona will not be able to use the system at all if the checkpoint is not met. If the checkpoint is important, the persona can use the product, but only with a high effort and in an inefficient way. If the checkpoint is beneficial, the product will be more usable for the persona if the checkpoint is met, but could be used even without the checkpoint being met. The same checkpoint may be 'essential' for one persona, but only 'important' or 'beneficial' to another.

To accommodate those cases where a particular checkpoint may actually make it harder for a particular persona to use the product, and for the sake of symmetry, three degrees of disadvantage are also defined: excluding, impeding, and inconveniencing. Excluding means that if the checkpoint is met, the persona can not use the product at all; impeding means that it could use it but only with

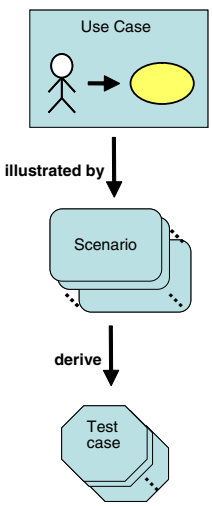

Fig. 6 Use case, scenarios and test cases

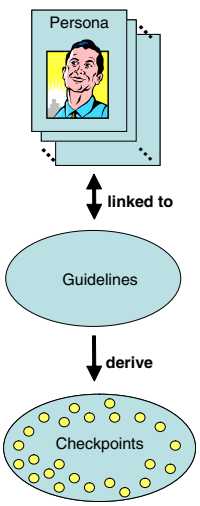

Fig. 7 Personas, guidelines and checkpoints

high effort; and inconveniencing means that the checkpoint makes the product less usable for the persona, but the persona can still use it without significant hardship.

A checkpoint may also be neutral to a persona, i.e., it has no impact on the accessibility and usability for the persona. Taken together, the relation between a checkpoint and a persona can be described along a scale of seven values as follows: essential, important, beneficial, neutral, inconveniencing, impeding, excluding. The goal is for a product to meet all "essential" and as many as possible "important" criteria for all personas, without creating any "excluding" or "impeding" elements for others.

The second requirement on standards and guidelines is that checkpoints must be available for multiple runtime platforms, if they vary for different platforms. Ideally, a standard or set of guidelines comes with multiple sets of checkpoints, each set complete in itself and pertaining to a specific platform. If in a development project the runtime platform was to be switched in the middle of the project, the set of checkpoints would be traded with the appropriate set of checkpoints for the new platform. In this trade, the set of requirements—based on the set of personas—would remain stable, and drive the selection of the checkpoints for the new platform.

5.3 Use automated and manual tests to verify accessibility

As with most testing, the testing of a product against accessibility requirements needs to be automated as much as possible. Running a test case in an automated fashion, makes it possible to reap the full benefit of regression tests that are run repeatedly over the lifecycle of the product. Repetitive tests are important because changes on a system's architecture or its user interface can break accessibility features that were previously built in. Ideally, even the test script (or part of it) that implements the test case, may be generated automatically.

Unfortunately, not all accessibility checkpoints can be tested by machines. For example, a machine can test whether a text equivalent is available for an image, but it can not test the quality of the text. However, if a tester validates a particular text for a particular image, a machine can validate whether the texts for all copies of the image are consistent throughout the application. Also, in subsequent test runs, a human tester needs to look at the text equivalent only if the image or the text has changed since the last test run. So, even for manual checkpoints, computers are useful to free a human tester from repetitive checks, and to keep a record of who tested what and when.

Most accessibility requirements are common across test cases and functional units if running on the same platform, so that guidelines and checkpoints can be reused across test cases and use cases within a project. This is because most accessibility requirements are concerned with user interface design, and as such repeat themselves for use cases with similar user interfaces.

There are a number of commercial and free tools available for accessibility checking that go a long way toward automatic testing. Most of these tools evaluate Web pages on the basis of guidelines and regulations such as the W3C Web Content Accessibility Guidelines 1.0 and section 508. Some of these tools allow configuring the set of checkpoints that are to be applied to a software or Web application. Unfortunately, the number and sophistication of the Web accessibility tools is not paralleled in the graphical user interface domain. However, this is likely to change, as the demand for productivity tools for accessible design in software development increases.

A project can draw from existing tools by having the developers write test code that calls these tools from a test case to validate the product against accessibility checkpoints. The test code may be even generated automatically. Of course, an appropriate test tool must be available and has to be configured to include checkpoints based on the requirements of the project, its use cases and personas. To further ease the application of accessible design principles in mainstream application development, accessibility test tools should be incorporated into the development environments in a seamless fashion.

6 Putting it together—a process model for accessible design and testing

The previous sections have introduced the components of the process model for accessible design and testing: use cases, personas, scenarios, guidelines, test cases and checkpoints. These model components are not new in software development.

As shown above, use cases, scenarios, test cases, personas, guidelines and checkpoints are proven tools, drawn from best practices in mainstream software engineering. However, this paper defines a new way to use these components for accessibility purposes, and to relate them to each other, thus facilitating automation as much as possible (see Fig. 8). For design projects that are employing a use case driven methodology, this approach allows to incorporate accessible design into the existing processes rather than having to add accessibility as a new process.

An example illustrating the benefit of this model, and the testing aspect in particular, is a situation where a test case is failing because an accessibility requirement is not met. In this case, the proposed model makes it possible to

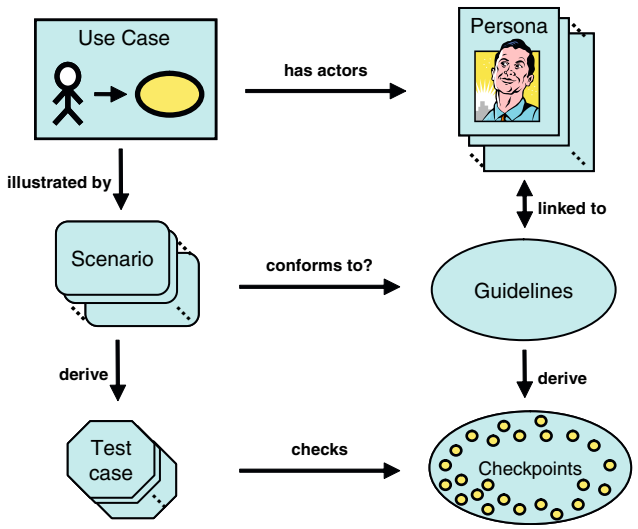

Fig. 8 Components of the integrated approach and their relationships: use case, scenarios, test cases; personas, guidelines and checkpoints

pinpoint to the particular checkpoint that is causing the failure, and trace it back to the particular guideline that is violated. This allows identifying the particular personas that will not be able to access the application because of that failure (see Fig. 9). This is not only useful for fixing the accessibility problem, but also provides context to the developer and an understanding of the consequences of the failure. Understanding the reason behind the guideline not only facilitates avoidance of the error in the future, but also increases the probability that the fix will be implemented in a proper and effective manner.

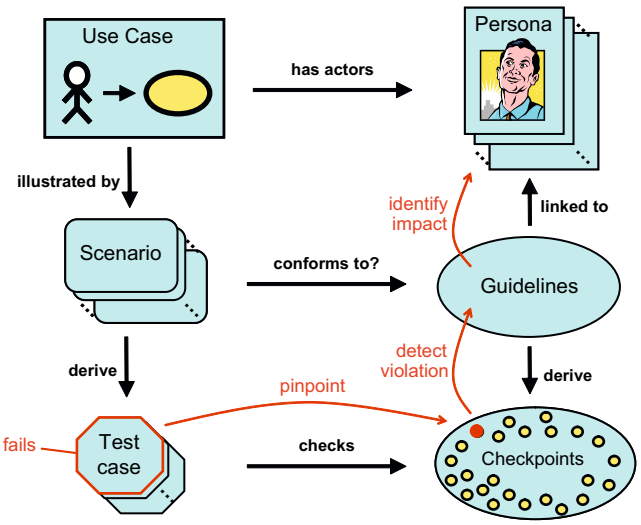

Fig. 9 When a test case fails, the model helps to pinpoint the appropriate checkpoint, detect the violated guideline and identify the impact on the user (personas)

7 Reinforcing the model—expert reviews and user testing

The described process model for accessible design and testing provides basic support for designers and implementers in the development of products that are accessible to the personas involved. However, since personas are only approximations of the users and not the users themselves, this model should be complemented with expert reviews and real user testing.

7.1 Expert reviews

Ideally, users should be involved at every stage of a project. This is mostly not possible due to budget and time constraints. Expert reviews can discover many design problems that user testing would identify, and provide guidance on how to fix them in advance—before users are brought on site.

Experts (who are familiar with the characteristics of people with disabilities and their implications on various product designs) can support the development of an application in various stages. At the beginning, they can help to identify the targeted users of a product, and the regulative requirements regarding accessibility and usability. The design or selection of personas based on the identified user base is a crucial task, and should not be performed without guidance by experts. To facilitate the process, a common sets of personas can be used that will be developed and verified in the context of research projects (cf. [14] as a first starting point). If necessary, these sets may be adapted to a specific application domain. A careful selection of the personas at the beginning of a project will pay off multiple times over a product's lifecycle.

Conducting accessibility reviews, experts can discover design problems, based on heuristics and their expertise. These reviews should be considered for every iteration of development, with emphasis on the earlier iterations, because design problems can be fixed easier if discovered earlier. Based on an expert review, it may be necessary to amend the personas, so that the set of personas gets more and more complete or understandable to designers with every iteration.

7.2 User testing

While expert reviews are significantly cheaper than user testing, there are design problems that can only be discovered by user testing. At some point in the project, the product should be tested by real users. It is recommendable to conduct user tests based on prototypes even at early

stages of the project, to mitigate accessibility risks as early as possible. Accessibility experts can help to plan for and to conduct appropriate user testing. Of course, they should be involved in analysis and evaluation of user testing.

Especially at earlier stages, the results of user testing can be used to fine-tune the set (and descriptions) of personas, in an attempt to have as many issues as possible covered by the personas and to have them described in ways that are most meaningful to the designers and for the purposes of the product under design. This will help to strengthen the efficiency of the proposed process model, and to improve its use across development projects. It is important that the knowledge about the design of personas that has been acquired in a project is reused beyond a product's lifecycle, in other development projects.

8 Conclusion

This paper has proposed and theoretically justified an integrated approach for accessible design and testing in the development of software applications. A key feature of the proposed process model is the combination of use cases, personas and accessibility guidelines, which drives the development of the product and triggers the inclusion of checkpoints for iterative testing. The proposed concepts fit well to an iterative development approach, but can be applied also to non-iterative processes.

It has also been emphasized that the model needs to be reinforced by expert reviews and user testing, thus mitigating for potential inaccuracies in the model and driving a process for continuous improvement of the model in a specific development context.

Some of the proposed concepts could only be touched upon in this paper, and other aspects were completely left out. Beyond the proposed concepts, the concept of architectural patterns and user interface patterns could be applied to generate code (semi-)automatically that would accommodate the user interface needs of the selected personas. This would increase the quality of code and boost development productivity.

Although the model presented here is based on author interaction with companies both large and small, the evaluation of the approach will, by necessity, only occur as companies experiment with these ideas. They will also need to be adapted to the various practices and constraints of the companies. Only in long-term practice can they (or variations of them) be proven.

In conclusion, the authors hope that this paper contribute to an "inclusive thinking" in application development that sets the user in the center of planning and builds accessibility features into software products from the beginning, rather than trying to add them on late in the lifecycle. In particular, development tool vendors and third-party suppliers for application development frameworks are encouraged to develop appropriate tools for an integrated approach in software development. The described integrated approach, together with appropriate tools being available, is envisioned to ultimately increase productivity and quality in making products accessible and usable for all of its users, and bring about new market opportunities for companies that excel in accessible design.

Acknowledgments This work was partially funded by the National Institute on Disability and Rehabilitation Research, US Department of Education under Grant H133E030012 as part of the Universal Interface and Information Technology Access Rehabilitation Engineering Research Center of the University of Wisconsin -Trace Center. The opinions herein are those of the authors and not necessarily those of the funding agency.

References

1. Antle A.: Child-user abstractions. In: CHI '06 Extended Abstracts on Human Factors in Computing Systems, Montréal, Québec, Canada, pp. 478–483. ACM, New York (2006)
2. Aoyama, M.: Persona-and-scenario based requirements engineering for software embedded in digital consumer products. In: Proceedings 13th of the IEEE International Conference on Requirements Engineering, pp. 85–94, 29 August–2 September 2005
3. Chisholm, W., Vanderheiden, G., Jacobs, I.: Web Content Accessibility Guidelines 1.0, W3C Recommendation, 5 May 1999. http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/
4. Cooper, A.: The Inmates are Running the Asylum. SAMS/Macmillan, Indiana (1999)
5. Dantin, U.: Application of personas in user interface design for educational software. In: Proceedings of the 7th Australasian Conference on Computing Education, vol. 42, Newcastle, NSW, Australia. ACM International Conference Proceeding Series, vol. 106, pp. 239–247 (2005)
6. Dayton, D.: Audiences involved, imagined, and invoked: trends in user-centered interactive information design. Proceedings of Professional Communication Conference, 21–24 Sept. 2003, pp. 327–335. IEEE International (2003)
7. Hackos, J., Redish J.: User and Task Analysis for Interface Design. Wiley, New York (1998)
8. Heumann, J.: Generating Test Cases from Use Cases. Rational edge, June 2001. http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf
9. HFES 200 Committee: Draft Standard for Trial Use: Human Factors Engineering of Software User Interfaces (BSR/HFES 200). Human Factors and Ergonomics Society, Santa Monica (2003, in revision 2006)
10. ISO TS 16071:2003: Ergonomics of Human–System Interaction—Guidance on Accessibility for Human–Computer Interfaces
11. Jacobson, I.: Object-Oriented Software Engineering: A Use-Case Driven Approach. Addison-Wesley, Reading (1992)
12. Kruchten, P.: The Rational Unified Process—An Introduction. Addison-Wesley, Reading (2004)
13. Microsoft (2004) The Wide Range of Abilities and Its Impact on Computer Technology. Research Study by Microsoft, conducted by Forrester Research. http://www.microsoft.com/enable

14. Mueller, J.: Getting personal with universal. Innovation, **23**(1) (2004). http://www.idsa.org/webmodules/articles/articlefiles/Mueller.pdf
15. Newell, A., Mueller, J., and Jones, M.: Promoting user sensitive inclusive design: strategies for communicating user needs to designers. In: Gibson, L., Gregor, P., Sloan, D. Accessible Design in the Digital World Conference 2005, Workshops in Computing (eWIC) Series (ISSN 1477-9358). British Computer Society, Wiltshire
16. Norman D.A.: The Invisible Computer. Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances are the Solution. MIT, Cambridge (1998)
17. Pruitt J., Grudin J.: Personas: practice and theory. In: Proceedings of the 2003 Conference on Designing for User Experiences, San Francisco, California, pp 1–15. ACM, New York (2003)
18. Randolph, G.: Use-cases and personas: a case study in light-weight user interaction design for small development projects. Inform Sci J **7**,105–116 (2004)
19. Rönkkö, K.: An empirical study demonstrating how different design constraints, project organization and contexts limited the utility of personas. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences. IEEE International, Washington, DC, USA, 3–6 January 2005
20. Sesto, M., Vanderheiden, G., Radwin, R.: Functional Characterization of Disability and Interface Use. RESNA 27th Annual Conference, Orlando, Florida (2004)
21. Schmidt, A., Terrenghi, L.: Methods and guidelines for the design and development of domestic ubiquitous computing applications. In: Fifth Annual IEEE International Conference on Pervasive Computing and Communications, pp. 97–107, March 2007
22. Shyba L., Tam J.: Developing character personas and scenarios: vital steps in theatrical performance and HCI goal-directed design. In: Proceedings of the 5th Conference on Creativity and Cognition, pp. 187–194. ACM, New York (2005)
23. Souza, R. Design Accessible Sites Now, Forrester Report, December 2001. http://www.forrester.com/ER/Research/Report/Summary/0,1338,11431,00.html
24. Vanderheiden, G.: Application Software Design Guidelines, Version 1.1, 1 June 1994. Trace Center. http://trace.wisc.edu/docs/software_guidelines/software.htm
25. Yu, Y., Liu, Z.: Research on a user-centered design method for interactive online teaching system. In: ICCT '06. International Conference on Communication Technology, pp. 1–4, November 2006
26. Zimmermann, G.: Access Technologies Group—Resources. http://www.accesstechnologiesgroup.com/Resources (2006)