Accessibility software

Wikipedia

In human–computer interaction, computer accessibility (also known as Accessible computing) refers to the accessibility of a computer system to all people, regardless of disability or severity of impairment, examples include Web accessibility guidelines.[5] Another approach is for the user to present a token to the computer terminal, such as a smart card, that has configuration information to adjust the computer speed, text size, etc. to their particular needs. This is useful where users want to access public computer based terminals in Libraries, ATM, Information kiosks etc.

It is largely a software concern;[*dubious* – *discuss*] when software, hardware, or a combination of hardware and software, is used to enable use of a computer by a person with a disability or impairment, this is known as Assistive Technology.

There are numerous types of impairment that affect computer use. These include:

- Cognitive impairments, illiteracy and learning disabilities, such as dyslexia, ADHD or autism.
- Visual impairment such as low-vision, complete or partial blindness, and color blindness.
    - For individuals with mild to medium vision impairment, it is helpful to use large fonts, high DPI displays, high-contrast themes and icons supplemented with auditory feedback and screen magnifying software. In the case of severe vision impairment such as blindness, screen reader software that provides feedback via text to speech or a refreshable braille display is a necessary accommodation for interaction with a computer.
    - About 8% of people, mostly males, suffer from some form of colour-blindness. The main colour combinations that might be confused by people with visual deficiency include red/green and blue/green. However, in a well-designed user interface, color should not be the only way of distinguishing between different pieces of information.
- Hearing-related disabilities including deafness, being hard of hearing, or hyperacusis.
    - While sound user interfaces have a secondary role in common desktop computing, usually limited to system sounds as feedback, software producers take into account people who can't hear, either for personal disability, noisy environments, silence requirements or lack of sound hardware. Such system sounds like beeps can be substituted or supplemented with visual notifications and captioned text (akin to closed captions).
- Motor or dexterity impairment such as paralysis, cerebral palsy, or carpal tunnel syndrome and repetitive strain injury.
    - Some people may not be able to use a conventional input device, such as the mouse or the keyboard. Therefore it is important for software functions to be accessible using both devices; ideally, software uses a generic input API that permits the use even of highly specialized devices unheard of at the time of software development. Keyboard shortcuts and mouse gestures are ways to achieve this. More specialized solutions like on-screen software keyboards and alternate input devices like switches, joysticks and trackballs are also available. Speech recognition technology is also a compelling and suitable alternative to conventional keyboard and mouse input as it simply requires a commonly available audio headset.

These impairments can present themselves with variable severity; they may be acquired from disease, trauma or may be congenital or degeneration in nature.

Accessibility APIs

- Microsoft Active Accessibility (MSAA) on Microsoft Windows
- Microsoft UI Automation on Microsoft Windows, replacing MSAA
- IAccessible2 on Microsoft Windows, a competitor of Microsoft UI Automation also replacing MSAA
- AT-SPI on UNIX and Linux
- Mac OS X Accessibility
- Java Accessibility and the Java Access Bridge for Java software.[4][5]

Some of these APIs are being standardised in the ISO/IEC 13066 series of standards.[6][7]

Accessibility Features in Mainstream Software

- Keyboard shortcuts and MouseKeys allow the user to substitute keyboarding for mouse actions. Macro recorders can greatly extend the range and sophistication of keyboard shortcuts.
- Sticky keys allows characters or commands to be typed without having to hold down a modifier key (Shift, Ctrl, Alt) while pressing a second key. Similarly, ClickLock[8] is a Microsoft Windows feature that remembers a mouse button is down so that items can be highlighted or dragged without holding the mouse button down throughout.
- Customization of mouse or mouse alternatives' responsiveness to movement, double-clicking, and so forth.
- ToggleKeys[9] is a feature of Microsoft Windows 95 onwards. A high sound is heard when the CAPS LOCK, SCROLL LOCK, or NUM LOCK key is switched on and a low sound is heard when any of those keys are switched off.
- Customization of pointer appearance, such as size, color and shape.
- Predictive text
- Spell checkers and grammar checkers

Software with support for learning disabilities

- Cause and effect software[10]
- Switch accessible software
- Hand-eye co-ordination skills software
- Diagnostic assessment software
- Mind mapping software
- Study skills software
- Symbol-based software[11]
- Text-to-speech
- Touch typing software

Aegis-project – Open Accessibility framework (OAF)

http://www.aegis-project.eu/index.php?option=com_content&view=article&id=176&Itemid=73

The Open Accessibility Framework (OAF - D1.2.1 AEGIS Open Accessibility Framework, version 1.2.5, CC licensed) consists of two parts:

- A document describing the **framework of things needed for 3rd generation accessibility**, as validated by the prototypes and user/developer feedback in AEGIS
- A collection of largely **open source prototypes and code Deliverables** implementing various aspects of the OAF, proven in AEGIS and contributed back to the open source projects of which they are part

It contains the initial AEGIS Open Accessibility Framework (OAF) description. It is based upon:

- The **accessibility API and framework support from the existing Open Desktop** (GNOME Accessibility framework) and the **Java platform** (the Java Accessibility API, keyboard operability guidelines, and theme support);
- The **AEGIS generic accessibility framework requirements** (AEGIS ID1.2.1).

The Java Accessibility API Interfaces and Classes

http://docs.oracle.com/cd/E17802_01/j2se/javase/technologies/accessibility/docs/jaccess-1.3/doc/core-api.html

http://docs.oracle.com/javase/7/docs/technotes/guides/access/jaapi.html

The Java Accessibility API defines a contract between individual user-interface components that make up a Java™ application and an assistive technology that is providing access to that Java application. If a Java application fully supports the Java Accessibility API, then it should be compatible with, and friendly toward, assistive technologies such as screen readers, screen magnifiers, etc. With a Java application that fully supports the Java Accessibility API, no screen reader off screen model would be necessary because the API provides all of the information normally contained in an off screen model.

US Section 508 Standards – software applications & operating systems

http://www.epa.gov/inter508/standards/index.htm

(a) When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.

(b) Applications shall not disrupt or disable activated features of other products that are identified as accessibility features, where those features are developed and documented according to industry standards. Applications also shall not disrupt or disable activated features of any operating system that are identified as accessibility features where the application programming interface for those accessibility features has been documented by the manufacturer of the operating system and is available to the product developer.

(c) A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.

(d) Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.

(e) When bitmap images are used to identify controls, status indicators, or other programmatic elements, the meaning assigned to those images shall be consistent throughout an application's performance.

(f) Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes.

(g) Applications shall not override user selected contrast and color selections and other individual display attributes.

(h) When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

(i) Color coding shall not be used as the only means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.

(j) When a product permits a user to adjust color and contrast settings, a variety of color selections capable of producing a range of contrast levels shall be provided.

(k) Software shall not use flashing or blinking text, objects, or other elements having a flash or blink frequency greater than 2 Hz and lower than 55 Hz.

(l) When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

"Engineering Software for Accessibility" – Microsoft Corporation

http://books.google.co.uk/books?id=LVnfMYuHaoYC&printsec=frontcover&dq=accessibility+software&hl=en&sa=X&ei=fABYUqbRE8rT0QXz6IGoDg&ved=0CEoQ6AEwAA#v=onepage&q=accessibility%20software&f=false

- When it comes to technology, accessibility pertains to a wide range of people with a range of abilities, not just the folks with disabilities.
- Accessible technology is technology that users can adapt to meet their visual, hearing, dexterity, cognitive, and speech needs and interaction preferences. Accessible technology includes accessibility options and utilities built into products, as well as specialty hardware and software add-ons called assistive technology (AT) that help individuals interact with a computer.

- There are essentially two types of users of accessible technology:
    1. Those who need it, because of disabilities or impairments, age-related conditions, or temporary conditions (such as limited mobility from a broken arm)
    2. Those who use it out of preference, for a more comfortable or convenient computing experience.
- A 2003-2004 study commissioned by Microsoft and conducted by Forrester Research found that over half – 57% – of computer users in the US between the ages of 18 and 64 benefit from accessible technology. Most of these users do not identify themselves as having a disability or impaired but expressed certain task-related difficulties or impairments when using a computer.

Table 1. Possible AT solutions users might use to address their difficulties or impairments

Class of Disability	User Experience Without AT	Possible AT Solutions
Vision		
Mild (low vision, color blindness)	Difficulty with legibility of software and hardware interfaces	▪ Setting changes to font size and colors ▪ Alternative font style and rasterization ▪ Larger screens
Severe (blindness)	Unable to use computer monitor, need the option of receiving information through hearing or touch	▪ Screen reader (for text-to-speech and sound cues) ▪ Audio description of video ▪ Refreshable Braille display ▪ Keyboard navigation
Dexterity		

Class of Disability	User Experience Without AT	Possible AT Solutions
Mild (temporary pain, reduced dexterity such as from a broken arm) to severe (paralysis, maybe carpal tunnel syndrome)	Using standard mouse or keyboard is painful or difficult	▪ Fine-tuning mouse and keyboard ▪ Software (on-screen) keyboard and mouse alternative ▪ Speech recognition utility ▪ Alternative input device, such as a joystick or head-tracking mouse
Hearing		
Mild (hard of hearing) to severe (deaf)	Difficulty distinguishing words and sounds or not at all, need to receive information visually	▪ Volume adjustments ▪ Sounds supplemented by visual cues ▪ Multimedia captioning

Class of Disability	User Experience Without AT	Possible AT Solutions
		▪ Sign language
Cognitive		
Mild (learning difficulties) to severe (Alzheimer's, dementia)	Difficulty with word recognition, memory, concentration, and reasoning; UI might be overwhelming	▪ Reading and learning aids ▪ Word prediction programs ▪ Audio speech paired with visual presentation ▪ Simplified UI ▪ Task reminders

By 2010, the number of accessible technology users is expected to rise to 70 million, up from 57 million users in 2003 (Forrester 2004). Among users who use built-in accessibility options and utilities, 68 percent have mild or severe difficulties or impairments, whereas the remaining 32 percent have no difficulties or impairments (Forrester 2004). Among users who use AT products, such as trackballs or screen magnifiers, 65 percent did not report health issues as reasons for using AT products, but rather cited that these products make computers easier to use, more

The Basics

As mentioned, programmatic access and keyboard access are two critical pieces to accessibility and are the basis for this book. Let's go over these two areas a little further, as well as some basic information and settings you should be aware of when developing for accessibility.

Programmatic Access

Programmatic access is critical for creating accessibility in applications. Programmatic access is achieved when an application or library of UI functionality exposes the content, interactions, context, and semantics of the UI via a discoverable and publicly documented application programming interface (API). Another program can use the API to provide an augmentative, automated, or alternate user interaction. Basic information conveyed through programmatic access includes: navigation, interactive controls, asynchronous changes to the page, keyboard focus, and other important information about the UI.

Programmatic access involves ensuring all UI controls are exposed programmatically to the AT. Without it, the APIs for AT cannot interpret information correctly, leaving the user unable to use the products sufficiently or forcing the AT to use undocumented programming interfaces or techniques never intended to be used as an "accessibility" interface. When UI controls are exposed to AT, the AT is able to determine what actions and options are available to the user. Without proper programmatic access, a user may receive useless, erroneous, or even no information about what they are doing in the program.

Keyboard Access

Keyboard access pertains to the keyboard navigation and keyboard focus of an application. For users who are blind or have mobility issues, being able to navigate the UI with a keyboard is extremely important; however, only those UI controls that require user interaction to function should be given keyboard focus. Components that don't require an action, such as static images, do not need keyboard focus.

It is important to remember that unlike navigating with a mouse, keyboard navigation is linear. So, when considering keyboard navigation, think about how your user will interact with your product and what the logical navigation for a user will be. In Western cultures, people read from left to right, top to bottom. It is, therefore, common practice to follow this pattern for keyboard navigation, though there are exceptions to this practice.

When designing keyboard navigation, examine your UI, and think about these questions:

- How are the controls laid out or grouped in the UI?
- Are there a few significant groups of controls?

  — If yes, do those groups contain another level of groups?
- Among peer controls, should navigation be done by tabbing around, or via special navigation (such as arrow keys), or both?

The goal is to help the user understand how the UI is laid out and identify the controls that are actionable. If you are finding that there are too many tab stops before the user completes the navigation loop, consider grouping related controls together. Some controls that are

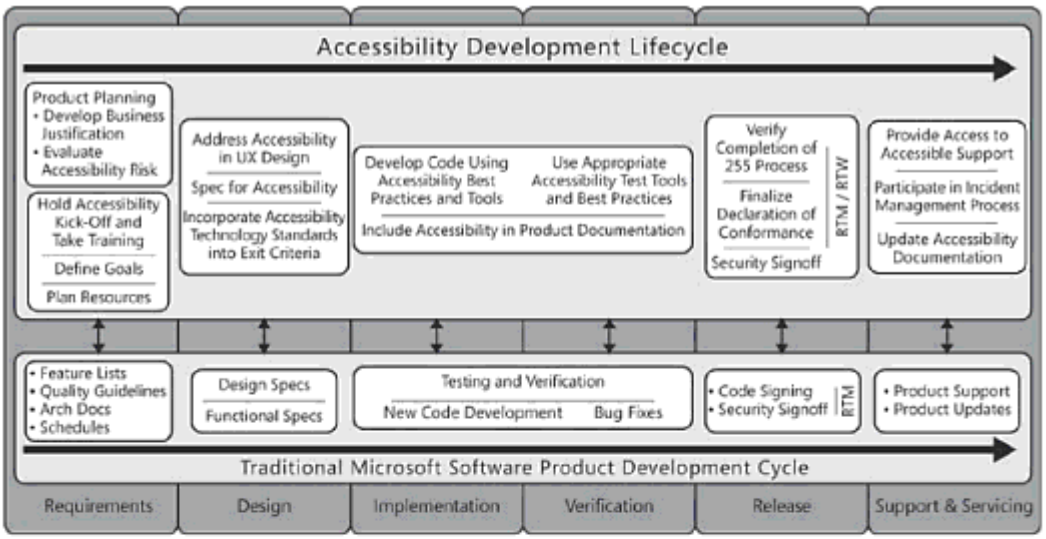

Figure 1. The development cycle

Requirements Stage

There may be a variety of reasons why you may want to incorporate accessibility into your product for a variety of reasons: you want to create software that's accessible for a loved one, you hope to sell your product to the U.S. government, you want to expand your market base, your company or the law requires it, or you simply desire to do the right thing for your customers. When you decide to create a new product or update an existing one, you should know whether you will incorporate accessibility into your product.

Once you have set your requirements, generate personas that exemplify users of varying types of abilities. Create scenarios to determine what design features will delight and assist your users, and illustrate how your

and make sure that all users can complete your use cases. Beware of blanks in your specifications! Your goal is to ensure that your product will be usable by people of varying abilities.

*Go further: For more information on personas, go to*http://go.microsoft.com/fwlink/?LinkId=150842.

Design Stage

In the design stage, the framework you will use is critical to the development of your product. If you have the luxury of choosing your framework, think about how much effort it will take to create your controls within the framework. What are the default or built-in accessibility properties that come with it? Which controls will you need to customize? When choosing your framework, you are essentially choosing how much of the accessibility controls you will get "for free" (that is, how much of the controls are already built-in) and how much will require additional costs because of control customizations. If accessibility was implemented in the past, look at the design docs for those earlier versions to see how accessibility features were implemented in them.

Once you have your framework, design a logical hierarchy to map out your controls (Chapter 2 covers this topic in more detail). If your design is too complex, or your framework won't even support the features that you are thinking of, it may not be worth the time, money, or effort to develop them. Accessibility can sometimes be a way to measure the usability and approachability of your product's overall design. For

instance, if you are finding that the design of your keyboard navigation or logical hierarchy is becoming way too complex, it's likely that your user will have a hard time navigating your UI and will have a bad experience with your product. Go back to the drawing board, and make sure you are following fundamental user experience (UX) and accessible design practices. It's likely that somebody has already addressed the same design issues you're facing.

When you have designed your programmatic access and keyboard access implementation, ensure that all accessibility API information is noted in the specs, including all the basic development settings touched on earlier (settings for high contrast, system font defaults, a dpi-aware UI, a 5:1 text-to-background contrast ratio, and color combinations that will be easy for users with color deficiencies to differentiate). Keep in mind that it may be harder (or easier) to adhere to certain accessibility settings, depending on the framework. Programmatic access is often limited by the UI framework for the application, so it is crucial in the design stage to reconfirm the standards and expectations of the accessibility API supported by the UI framework. Keyboard navigations and the flexibility of accessibility implementations are usually tied to the architecture of the UI framework.

It is absolutely critical to note that when designing your programmatic access, *you should avoid creating new custom controls as much as possible*, because the cost for development, documentation, and help on how to interact with the control is significant, and ATs may not know how to interact with the control.

Implementation Stage

In the implementation stage, you will need to make sure that the chosen architecture and specs will work. If the specs do not work, go back to the design stage, and figure out a more effective or less expensive alternative.

When you implement the specs, be sure to keep the user experience in mind as you develop your product. Accessibility personas are great for reminding you of who your users are!

Verification Stage

In the verification or test stage, ensure that all the specs were implemented correctly and that the accessibility API is reporting correctly for programmatic access. Your accessibility API, such as UIA, must expose correctly to AT. For testing, use both accessibility test tools and full-featured, third-party accessibility aids. Write test cases and build verification tests for your accessibility scenarios to ensure that all the specs were implemented correctly.

Consider leveraging automated testing, and establish a process and metrics for accessibility bugs. You want to have clear and consistent severity ratings for these problems. Such ratings may look something like the following:

- High severity means that no workarounds are available for your target users, or the bug blocks the user from completing the task.
- Moderate severity means that workarounds are available, or that the bug does not block the user's ability to complete the operation. Do

not overlook moderate severity issues, just because there is a workaround. These issues can sometimes introduce other, significant usability or product quality issues.

- Low severity means that the bug's impact to accessibility with workarounds is low.

The verification stage is a good time to start documenting all the accessibility options and features of your product. Just be sure to create documentation for your users in accessible formats! If you hope to sell your product to the U.S. government, you may also start funneling this information into a Section 508 Voluntary Product Accessibility Template (VPAT), which is a standardized form developed by the Information Technology Industry Council (ITIC) to show how a software product meets key regulations of Section 508 of the Rehabilitation Act. You absolutely want to address any high severity issues before the VPAT process, as any problems with your product will be subject to VPAT documentation.

Before your final release, be sure to obtain and incorporate feedback from your customers and partners throughout the development cycle. Include people with disabilities in your usability studies and beta testing. Work with your usability team to plan for specific accessibility studies. Include AT vendors in feedback programs, and collaborate with them to ensure that their products work with yours. Ideally, you should not need to make any major changes to your product at this stage. Any major (or expensive) changes should be reserved for your next revision.

*Go further: For more information on accessibility tools and declarations of conformance, go to*http://go.microsoft.com/fwlink/?LinkId=150842.

Research about accessibility by Microsoft

http://www.microsoft.com/enable/research/default.aspx

Microsoft commissioned Forrester Research to measure the market for accessible technology in the United States and to better understand how accessible technology is being used.

Findings are represented in the following two reports:

- **The Market for Accessible Technology—The Wide Range of Abilities and Its Impact on Computer Use** presents findings about individuals who are likely to benefit from the use of accessible technology.
- **Accessible Technology in Computing—Examining Awareness, Use, and Future Potential** presents findings about the use of computers among individuals with difficulties/impairments.

Oxford Dictionaries – def accessibility

http://oxforddictionaries.com/definition/english/accessible?q=accessibility#accessible__8

adjective

- **1**(of a place) able to be reached or entered:*the town is **accessible by** busthis room is not **accessible to** elderly people*
  - able to be easily obtained or used:*making learning opportunities more **accessible to** adults*
  - easily understood or appreciated:*an accessible account of his theories*
  - able to be reached, entered, or used by people who have a disability:*features such as non-slip floors and accessible entrances*
- **2**(of a person, especially one in a position of authority) friendly and easy to talk to; approachable:*he is more accessible than most tycoons*

IBM – def software accessibility

http://pic.dhe.ibm.com/infocenter/jviewmap/v8r8/index.jsp?topic=%2Fcom.ibm.ilog.jviews.defense.doc%2FContent%2FVisualization%2FDocumentation%2FJViews%2FJViews_Defense%2F_pubskel%2Fps_usradvfwork1429.html

An accessible software product is software that can be used effectively by users with certain kinds of disabilities. Accessibility involves converting software that can only be used by people without disabilities into software that can be used by people with and without disabilities alike.

JViews Enterprise includes support for users with several kinds of disabilities and each kind of disability is handled by one or more software techniques. Some techniques are implemented by JViews Enterprise, whereas others, as in the case of thin-client applications, are implemented by the browser; operating system, and support software.

Note

Applications that use JViews Enterprise are not automatically accessible. They must use at least some of the APIs and techniques described here to become accessible. The design tools, such as the designers, and most of the JViews samples are not accessible. They are not part of the applications that you deploy to your users. Rather, the techniques for accessibility are exemplified by specific samples for each product.

The following types of disabilities can hinder a user's ability to work with visualization products created using JViews Enterprise.

- Physical disabilities that prevent the user from holding or controlling a mouse. For these users, JViews Enterprise provides keyboard navigation across a diagram or chart. See the section "Keyboard navigation" for your product.
- Physical disabilities that prevent the user from typing and using a keyboard with the same ease and speed as normal users. For these users, the operating system provides keyboard accessibility features. These features are accessible across Java/Swing, browsers, and JViews. See the section "Keyboard operation modes" for your product.
- Photosensitive epilepsy. For users with this type of disability, the blinking frequency of blinking objects must be limited. Only blink periods longer than 0.5 seconds are acceptable. This can be implemented using the JViews`IlvBlinkingColor` and `IlvBlinkingPaint` classes.
- Low vision. For users with low vision, it might be necessary to enable “zoomed” or “high contrast” display. See the sections "Zoomed display" and "High contrast mode" for your product.
- Color blind users. For color blind users, you must present information in a form other than colors. You can let the application auto-select the colors from a palette that avoids combinations of colors that would be indistinguishable to a color blind user. See the section "Use of colors in accessible applications" for your product.
- Blind users. Use screen readers for blind users. A screen reader is a software application that reads portions of the screen aloud. Screen readers are available for Windows, either built into the operating system, or available from third-party vendors, and for Linux/Gtk. The built-in screen reader is a simpler version with less features. JViews Enterprise does not currently support these devices.

Also includes info on Java APi for keyboard navigation etc.

References

Java accessibility APi

- http://docs.oracle.com/cd/E17802_01/j2se/javase/technologies/accessibility/docs/jaccess-1.3/doc/core-api.html
- http://docs.oracle.com/javase/7/docs/technotes/guides/access/jaapi.html

Books

- Engineering Software for Accessibility by Microsoft Corporation. Microsoft Press, 2009
- Sarah Horton, Access by Design : A Guide to Universal Usability for Web Designers , New Riders, 2005
- Clarkson et al (Ed), Designing Accessible Technology, Springer, 2006

OAF

- http://www.aegis-project.eu/index.php?option=com_content&view=article&id=176&Itemid=73

Web accessibility

- initiative
  - http://www.w3.org/WAI/
- Evaluation tool
  - http://www.webaim.org/

Research about accessibility by Microsoft

- http://www.microsoft.com/enable/research/default.aspx

US section 508 standards

- http://www.epa.gov/inter508/standards/index.htm

Oxford Dictionaries – definition of accessibility

- http://oxforddictionaries.com/definition/english/accessible?q=accessibility#accessible__8

IBM – definition of software accessibility

- http://pic.dhe.ibm.com/infocenter/jviewmap/v8r8/index.jsp?topic=%2Fcom.ibm.ilog.jviews.defense.doc%2FContent%2FVisualization%2FDocumentation%2FJViews%2FJViews_Defense%2F_pubskel%2Fps_usradvfwork1429.html

IBM ILOG Visualization for Java (JViews Enterprise Features)

- http://www-01.ibm.com/software/integration/visualization/java/