About the Authors

Jason Grieves is a Program Manager in the Windows Accessibility Group. Jason works with students of all ages to identify their abilities rather than disabilities. In turn, he finds solutions to use those abilities to live, work, and play.

Masahiko Kaneko is a Senior Program Manager for UI Automation. A program manager in accessibility at Microsoft for more than 10 years, he has been involved with several releases of the Windows Operating System as well as many other Microsoft products.

Technical Contributors

Larry Waldman has been a Program Manager working on Microsoft Office and accessibility for more than four years. While working on Office, he has led research in graphics accessibility, and recently became the driver for accessibility across the entire line of Office products.

Annuska Perkins is a Senior Accessibility Strategist at Microsoft. She is passionate about improving the usability and effectiveness of accessible technology solutions. She does product planning and incubation, in collaboration with business groups across Microsoft.

Greg Rolander is a programming writer in the Windows Experience division. Greg writes the documentation for the Windows SDK for the Windows Automation API, as well as several other Windows components.

Introduction

What comes to mind when you think of accessibility? If you're like most people, you might conjure up images of a wheelchair or perhaps someone who is blind. What about someone with a broken arm, a child with a learning disability, or a 65-year-old who needs high-prescription eyeglasses to read? When it comes to technology, accessibility pertains to a wide range of people with a wide 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), and (2) those who use it out of preference, for a more comfortable or convenient computing experience. The majority of computer users (54 percent) are aware of some form of accessible technology, and 44 percent of computer users use some form of it, but many of them are not using AT that would benefit them (Forrester 2004).

A 2003–2004 study commissioned by Microsoft and conducted by Forrester Research found that over half—57 percent—of computer users in the United States between the ages of 18 and 64 could benefit from accessible technology. Most of these users did not identify themselves as having a disability or impaired but expressed certain task-related difficulties or impairments when using a computer. Forrester (2003) also found the following number of users with these specific difficulties:

- One in four experiences a visual difficulty.
- One in four experiences pain in the wrists or hands.
- One in five experiences hearing difficulty.

Besides permanent disabilities, the severity and type of difficulty or impairment an individual experiences can vary throughout a person's life. Table I-1 lists the four key classes of disabilities and the types of accessibility options, utilities, or AT devices your users might use to address their difficulties or impairments.

TABLE I-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		
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 • 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 comfortable, and more convenient, or that they wish to avoid developing a future health issue (Forrester 2004).

If a majority of your users could benefit from your product being accessible, doesn't it just make sense to build an accessible product? If you have decided to do so, you are sending a message to your customers that their needs matter. Populations in many countries are getting older. Civil rights for people with disabilities are gradually being extended to encompass digital inclusion. Governments are requiring procurement officials to purchase products that are the most accessible (mandated in the U.S. by Section 508 of the Rehabilitation Act). For technology producers, creating accessible products is just the right thing to do, and it makes good business sense.

Who Should Read This Book

This book is intended to be an introduction to create accessible software products. If you want to understand how to incorporate programmatic access and keyboard access into your interfaces and how accessibility fits into the software development cycle, this book is for you. If you are a project manager or someone who is overseeing the development of an accessible product, you should also find this book helpful in understanding how accessibility is integrated at each stage of the development cycle.

What This Book Covers

As you might guess, accessibility should be integrated from the beginning of the product development cycle, when the application or product is in the planning or design phase, rather than later, when retrofitting your product for accessibility can be extremely costly—and sometimes impossible, because part of accessibility development requires attention at the architecture level. This book will guide you through the process of planning for the two critical pieces for accessibility, programmatic access and keyboard access, from the beginning of the software development lifecycle and integrating it throughout. It is, therefore, suggested that you first read the chapters in this book sequentially and then afterwards use this book as a reference as you develop your product. This book will also show you how to map out the logical hierarchy for your product and plan for implementation using UI Automation (UIA), Microsoft's accessibility API, to create products that work with assistive technologies.

Here is what to expect in each chapter:

- **Chapter 1, "The UI Automation Environment,"** provides definitions and an overview of UIA and its role in accessibility.
- **Chapter 2, "Designing the Logical Hierarchy,"** walks you through the steps for designing a logical hierarchy of your product, which will serve as a model for your accessibility implementation.
- **Chapter 3, "Designing Your Implementation,"** guides you through the process of designing the implementation of the controls in your UI.
- **Chapter 4, "Testing and Delivery,"** discusses testing for the programmatic access and keyboard access in your product and documentation for delivery, as well as a brief summary of steps for incorporating accessibility into your product.

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 related, such as a hybrid control, may need to be addressed at this early exploration stage. Once you begin to develop your product, it is difficult to rework the keyboard navigation, so plan carefully and plan early!

Go further: For guidelines on designing keyboard focus and keyboard navigation, go to http://go.microsoft.com/fwlink/?LinkId=150842.

Respect Your User

When developing accessible products, a key thing to keep in mind is to respect your end user's preferences and requirements. Whether they are selecting larger icons, choosing high contrast, or using a screen reader, users configure their system settings for a more comfortable user experience. It is absolutely essential, then, that you allow system-wide settings to work with your product. Overriding those settings through hard-coding might impede or even prevent a user from accessing parts of your products.

Visual UI Design Settings

When designing the visual UI, ensure that your product has a high contrast setting, uses the default system fonts and smoothing options, correctly scales to the dots per inch (dpi) screen settings, has default text with at least a 5:1 contrast ratio with the background, and has color combinations that will be easy for users with color deficiencies to differentiate.

High Contrast Setting

One of the built-in accessibility features in Microsoft's Windows operating systems is the High Contrast mode, which heightens the color contrast of text and images on the computer screen. For some people, increasing the contrast in colors reduces eyestrain and makes it easier to read. When you verify your UI in high contrast, you want to check that controls, such as links, have been coded consistently and with system colors (not with hard-coded colors) to ensure that they will be able to see all the controls on the screen that a user not using high contrast would see.

System Font Settings

To ensure readability and minimize any "unexpected" distortions to the text, make sure that your product always adheres to the default system fonts and uses the anti-aliasing and smoothing options. If your product uses custom fonts, users may face significant readability issues and distractions when they customize the presentation of their UI (through the use of a screen reader or by using different font styles to view your UI, for instance).

High DPI Resolutions

For users with vision impairments, having a scalable UI is important. UIs that do not scale correctly in high dpi resolutions may cause important UI components to overlap or hide other components and can become inaccessible. Since the release of Windows Vista, the Windows platform replaced large font settings with dpi configurations.

Go further: For more information on how to write high dpi applications, go to http://go.microsoft.com/fwlink/?LinkId=150842.

Color Contrast Ratio

The updated Section 508 of the Americans with Disability Act, as well as other legislations, requires that the default color contrasts between text and its background must be 5:1. For large texts (18-point font sizes, or 14 points and bolded) the required default contrast is 3:1.

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

Color Combinations

About 7 percent of males (and less than 1 percent of females) have some form of color deficiency. Users with colorblindness have problems distinguishing between certain colors, so it is important that color alone is never used to convey status or meaning in an application. As for decorative images (such as icons or backgrounds), color combinations should be chosen in a manner that maximizes the perception of the image by colorblind users.

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

How Accessibility Fits into the Development Cycle

Now that we've covered some of the basics, let's talk about how accessibility fits into each stage of the development cycle—requirements, design, implementation, verification, and release. You can adapt this model to the development cycle for your product. Figure I-1 provides a comprehensive view of a traditional software development cycle and activities you can do to incorporate accessibility into your product.

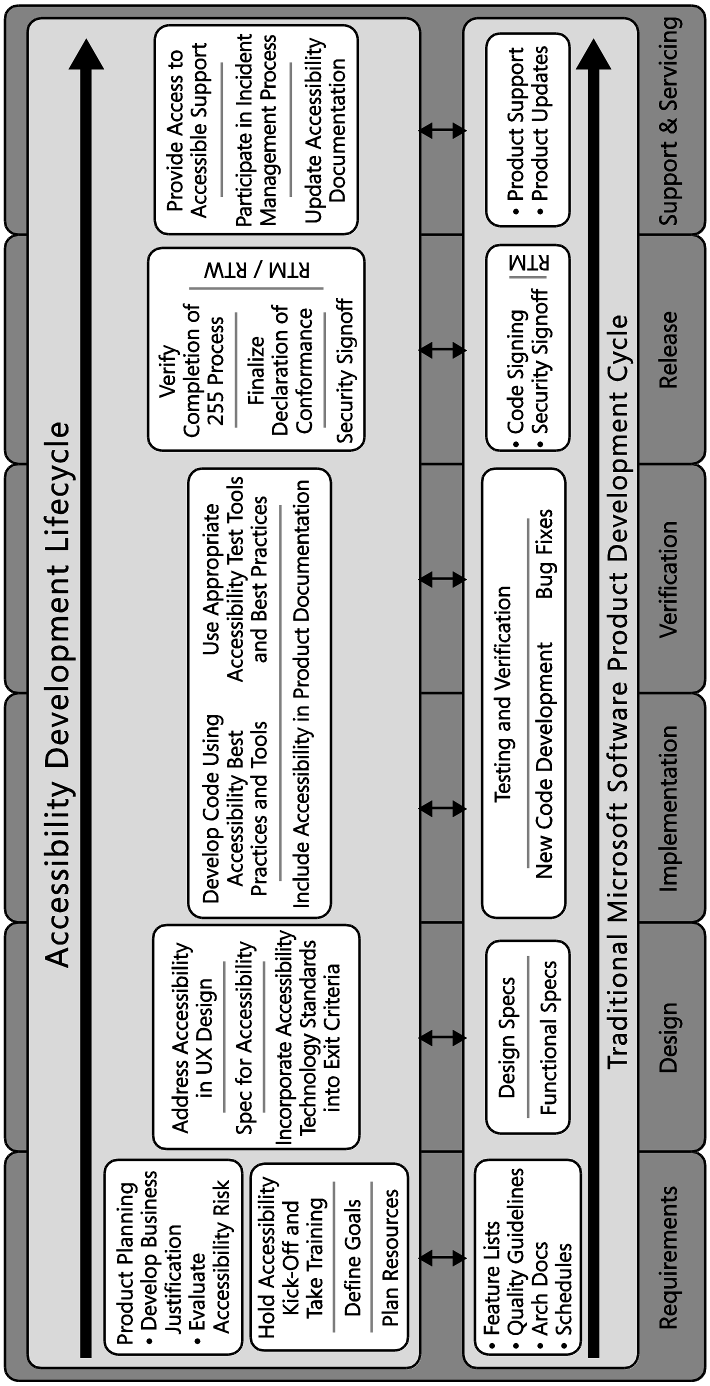
Accessibility Development Lifecycle
Product Planning
• Develop Business Justification
• Evaluate Accessibility Risk
Hold Accessibility Kick-Off and Take Training
Define Goals
Plan Resources
Address Accessibility in UX Design
Spec for Accessibility
Incorporate Accessibility Technology Standards into Exit Criteria
Develop Code Using Accessibility Best Practices and Tools
Use Appropriate Accessibility Test Tools and Best Practices
Include Accessibility in Product Documentation
Verify Completion of 255 Process
Finalize Declaration of Conformance
Security Signoff
RTM / RTW
Provide Access to Accessible Support
Participate in Incident Management Process
Update Accessibility Documentation
Traditional Microsoft Software Product Development Cycle
• Feature Lists
• Quality Guidelines
• Arch Docs
• Schedules
Design Specs
Functional Specs
Testing and Verification
New Code Development
Bug Fixes
• Code Signing
• Security Signoff
RTM
• Product Support
• Product Updates
Requirements
Design
Implementation
Verification
Release
Support & Servicing

FIGURE I-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 users will accomplish tasks with your product. Prioritize your features, 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.

Release Stage

In the release stage, continue to engage with AT vendors and users. Include accessible documentation both internally and externally with your product, and collaborate with your marketing group on go-to-launch activities and external messaging for your product.

Ready, Set, Go!

At this point, you should now have a general understanding of what accessibility is, the types of AT your users may be relying on to use your product, the basic development settings you should include in your product, and how accessibility fits into the development cycle. You are now ready to learn more about the various components in the UIA architecture, how to design a logical hierarchy, design your implementation, and how to test your implementation and deliver your product. Through each stage of the process, you will continue to learn how to set the foundation for accessibility through programmatic access and keyboard access. For more information on the visual UI design settings mentioned earlier (such as high contrast, default font, and high dpi settings), which are also necessary for an accessible product, check out the sample of resources we provide to get you started.

Remember, designing and developing for accessibility is one of the best ways to give you clarity about the user experience in general. By creating accessible products, you are working to improve the user experience for all people. The next chapter proceeds with an introduction to UIA, Microsoft's accessibility API, which will help you integrate accessibility into your product.

Find Additional Content Online As new or updated material becomes available that complements your book, it will be posted online on the Microsoft Press Online Developer Tools Web site. The type of material you might find includes updates to book content, articles, links to companion content, errata, sample chapters, and more. This Web is available at *www.microsoft.com/learning/books/online/developer*, and is updated periodically.

Support for This Book

Every effort has been made to ensure the accuracy of this book. As corrections or changes are collected, they will be added to a Microsoft Knowledge Base article.

Microsoft Press provides support for books at the following Web site:

http://www.microsoft.com/learning/support/books/

Questions and Comments

If you have comments, questions, or ideas regarding the book, or questions that are not answered by visiting the sites above, please send them to Microsoft Press via e-mail to

mspinput@microsoft.com.

Or via postal mail to

Microsoft Press

Attn: *Engineering Software for Accessibility* Editor

One Microsoft Way

Redmond, WA 98052-6399.

Please note that Microsoft software product support is not offered through these addresses.

References

Forrester Research, Inc. 2004. "Accessible Technology in Computing: Examining Awareness, Use, and Future Potential." Cambridge, MA: 22–41.

————. 2003. "The Wide Range of Abilities and Its Impact on Technology." Cambridge, MA: 7–18.