



Access by Contract: The New Way to Make Technology Accessible

Peter Korn
Accessibility Architect
Sun Microsystems, Inc.



Agenda

- A brief history: Accessibility from 1960 to today
- Sun's philosophy & approach to accessibility
- How things work today in Windows:
 - > A case study of JAWS with MS-Office
- How things work today in UNIX (and perhaps tomorrow in Windows)
 - > Well defined, rich protocols expose application & content information explicitly (“engineered accessibility”)

1st Generation Accessibility: late 1960s, 1970s, early 1980s

- Blind access with speech, Optacon
- Low vision access via special hardware
- Early Braille printers
- Talking calculators

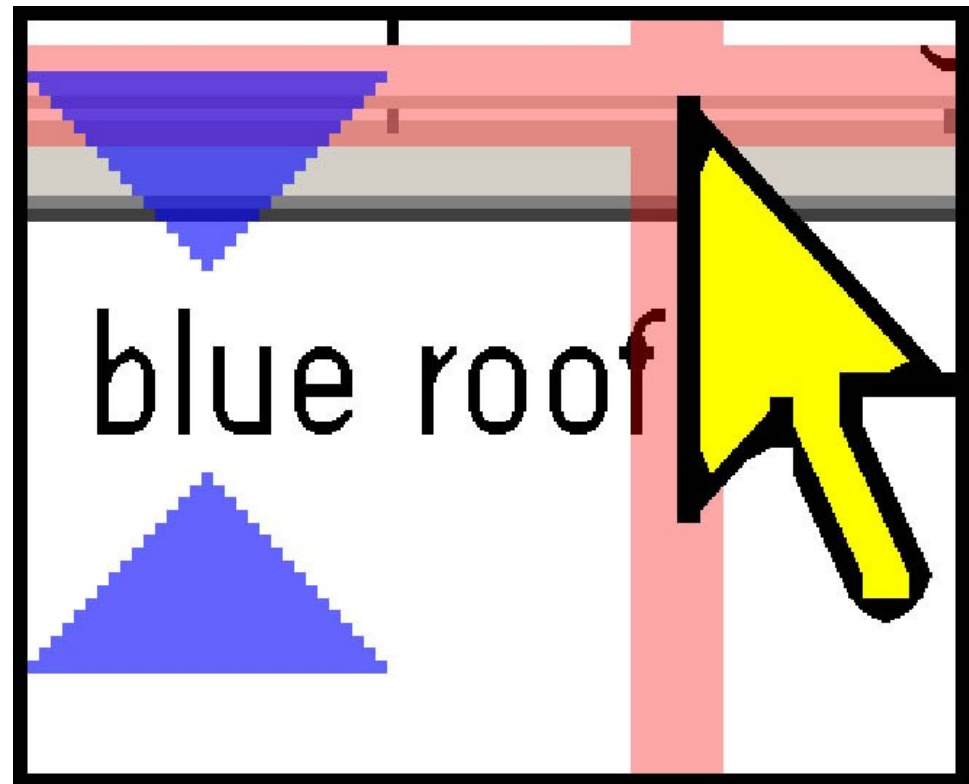


1st Generation Limitations

- Options expensive, difficult to use (such as Optacon)
- Computing limits restricted what users could do – only job access; no Web, no on-line books
- Very few disabilities served
 - > No augmentative communication devices
 - > Poor options for physical impairments
 - > No voice recognition
 - > Nothing for cognitive impairments

2nd Generation Accessibility: late 1980s, 1990s, early 2000s

- Software TTS, access to the GUI, scripting
- Software mag
- Voice recognition, OCR, Aug Comm
- WYNN, TextHelp, other LD products



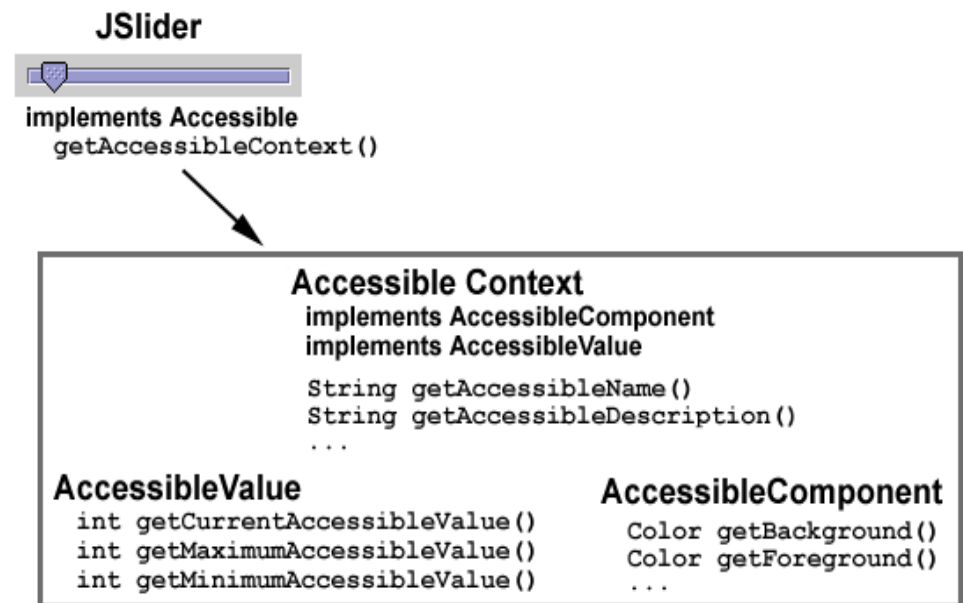
2nd Generation Limitations

- Patching of OS display drivers, keyboard driver means solutions are brittle: must update every OS release
- Special case work for applications – results in brittle solution: must update ever app update
- Many apps bypass patching mechanisms, so are inaccessible or need lots of special work:
 - > HyperCard, DirectX, X Servers, Terminal Server

Note: Section 508 developed in this environment

3rd Generation Accessibility: 1997 and onward

- Access by Contract:
“Engineered Accessibility”
- Every UI element implements it
- Everything needed by all AT provided
- Rich, extensible, flexible, powerful



3rd Generation Accessibility: How it came to be; why needed

- HyperCard, Macintosh X Servers → Access Aware
- Web Accessibility → WCAG, UAAG, ATAG
- Microsoft Office → MSAA
- Java via Direct X & own text rendering → JA-API
- UNIX & X → GNOME Accessibility API
- Macintosh OS X → Apple Accessibility API

Sun's Approach to 3rd Generation: Sun Accessibility effort, goals

- Ensure that all Sun products and technologies are accessible to people with disabilities
- Make it very easy to build accessible products on top of Sun technologies

“Accessibility is about enabling people with disabilities to participate in substantial life activities that include work and the use of services, products, and information.”

First big theme:

Built-in vs. Bolt-on

- Working with mature platforms means accessibility comes late to the table, is never really part of the underlying design
- Working with young platforms means making a gamble - will the platform be important enough in the future?

Second big theme: Evolution of screen access

- First generation - access to TTY systems [primitive]
 - > Magnification as a special video card
 - > Access to text in the video buffer (C/PM and DOS access)
- Second generation - access to the GUI [brittle]
 - > Video buffer re-direction & re-rendering for magnification
 - > Off-Screen Model for screen reading
 - > Patches in the OS for on-screen keyboard, other access
 - > App-specific customization in speech recognition & control
- Third generation - access via an API [flexible, rich]
 - > Magnification still needs video buffer re-direction
 - > Screen reading direct to the API
 - > On-Screen keyboard can be much more dynamic
 - > Products for cognitive impairment can be system-wide

Third big theme: Formal division of responsibility

- First & second gen., AT had to do everything
 - > Get the text, determine context (from buffer or OSM)
 - > Magnify the text
 - > Special-case applications (MS-Office, Internet Explorer)
 - > Create specialized drivers for specialized hardware
- The climate has changed
 - > Greater awareness of people with disabilities
 - > Laws worldwide requiring accessibility
- Proposal: divide the work into three pieces:
 - > Platform: define, implement accessibility architecture
 - > Application: support the platform accessibility arch.
 - > AT: focus on the user experience

Fourth big theme: Open source accessibility

- All source code available for examine
- AT developers can fix their own bugs, release their own patches
- AT may be open source too
- Vendors and users can control their own destiny

Problems People Face

- Assistive Technology price per machine
 - > Screen reader (JAWS): \$900-\$1,300
 - > Screen magnifier (ZoomText): \$600
 - > Other AT products: variety of prices
- Deployment
 - > Dedicate a system to a use; expensive and wasteful in computer labs
 - > Systems with AT very brittle – don't let non-disabled touch them!
- Assistive Technology upgrades expensive, frequent

Open Source Accessibility Benefits

- Built in: a great price!
- Supported architecture for accessibility; things no longer brittle
- Huge collection of apps which support the architecture, interoperate on the desktop
- AT from the same vendor as desktop apps (StarOffice, Mozilla, Evolution, etc.) - single source for assistance

Open Source Community Engagement makes this Work!

- GNOME
 - > Project began with a public meeting that was webcast (and real-time close captioned)
 - > “Universal Accessibility” core GNOME value
- OpenOffice.org
 - > Design & implementation public, two-way
- Mozilla
 - > Sun engineers continued work started by Netscape; working in concert with IBM



Helen Keller Achievement Award



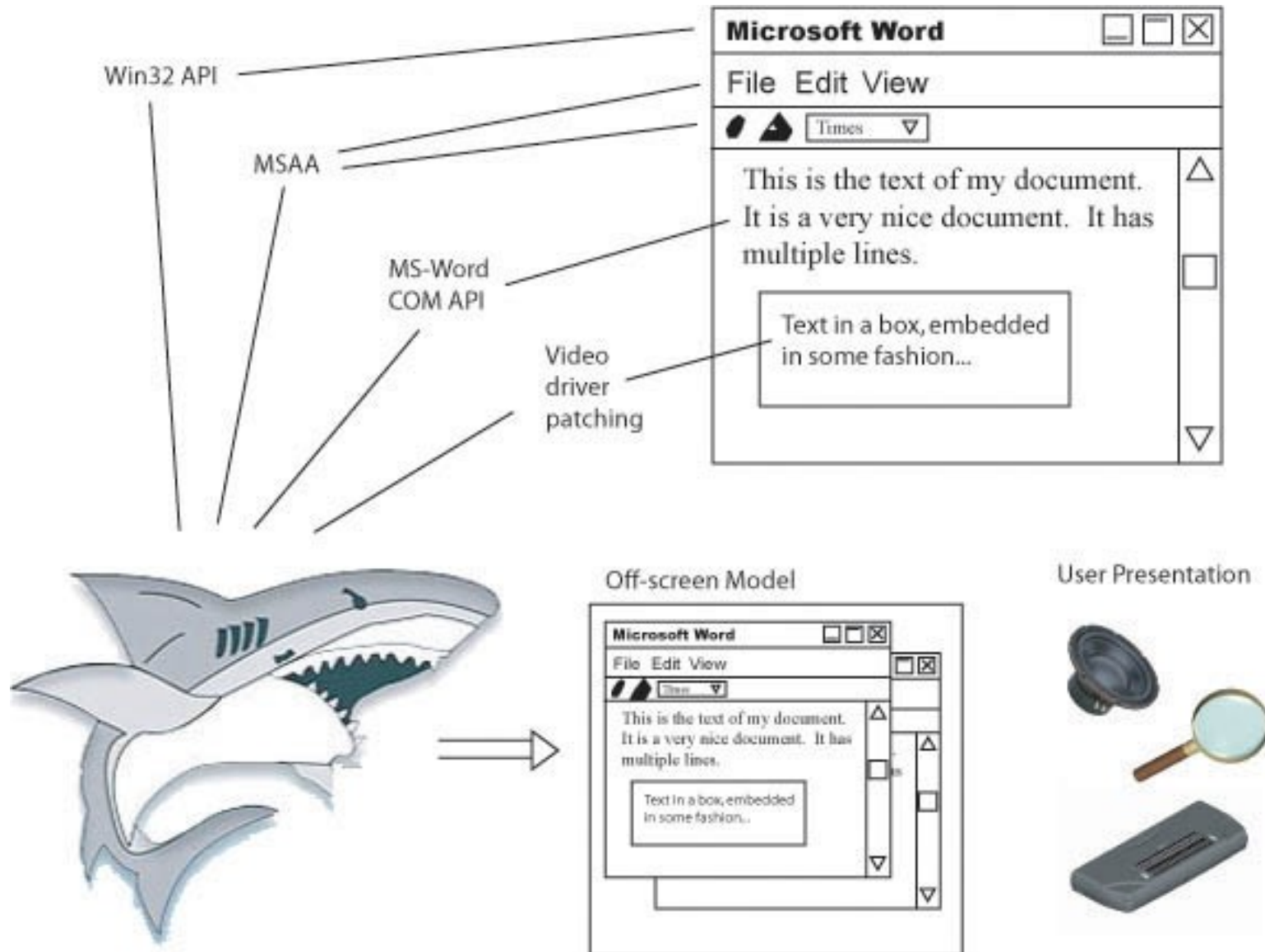
Sun/GNOME Accessibility Architecture Adoption

- Core part of GNOME platform
- KDE/Qt to adopt it in version 4.0
- Adobe Reader 7 for Linux supports it
- Free Standards Group Accessibility Working Group to base accessibility standard around this architecture (members include Sun, IBM, and Adobe)
 - > see <http://www.a11y.org>

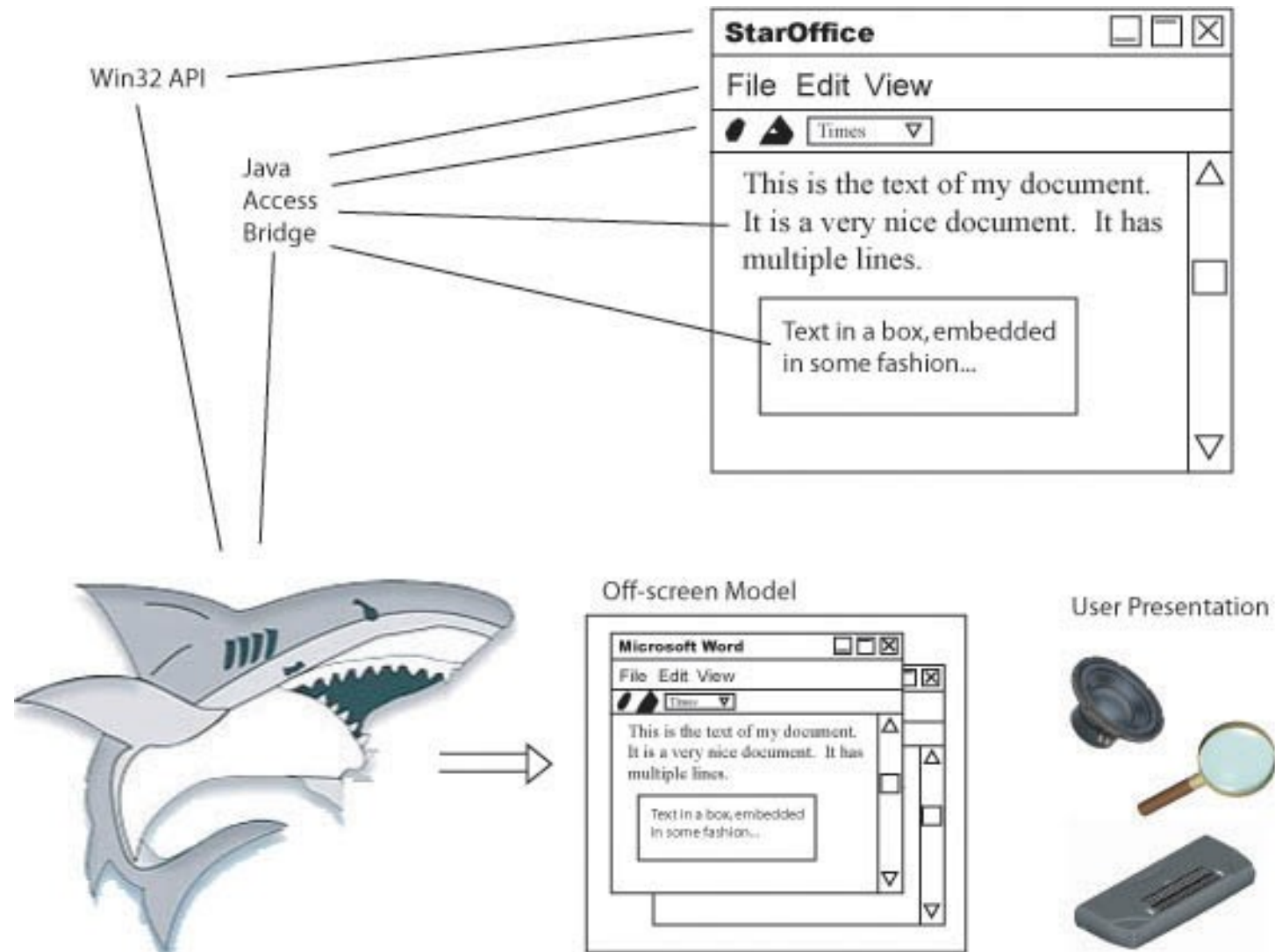
FOSS/UNIX Accessibility Timeline

1994	AccessX moves into X on SunOS, DEC UNIX
1997-8	Built javax.accessibility.* in Swing 
1998	Section 508 amendment signed into law
2000	Began GNOME, StarOffice, Mozilla accessibility 
2001	Section 508 amendment regulations go into effect
2003	GNOME 2.4 developer release with AT included Free Standards Group Accessibility workgroup starts
2005	Solaris 10 ships with accessibility support, AT
2006	Ubuntu Linux 6.10 ships with accessibility, AT

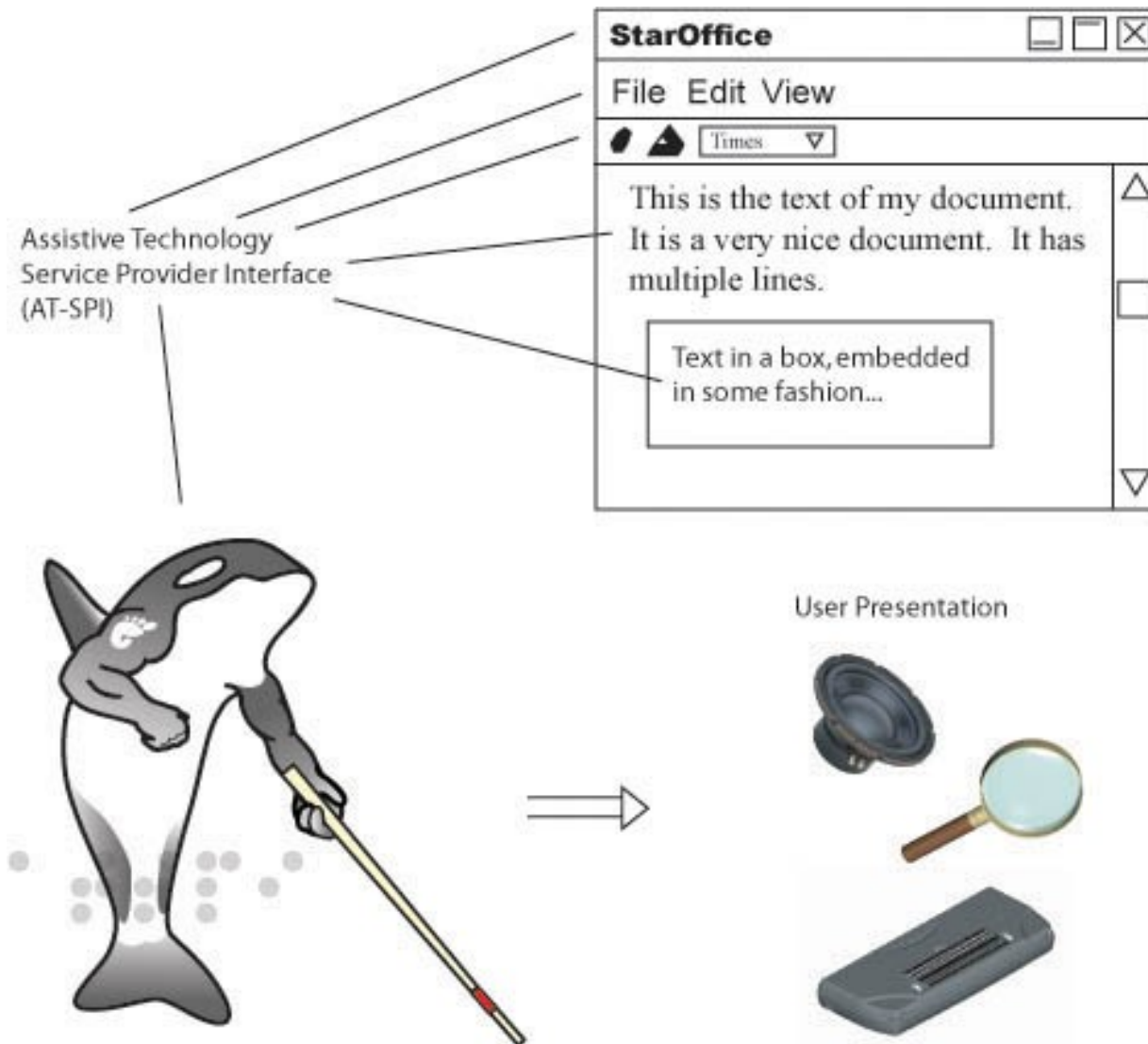
AT in Windows



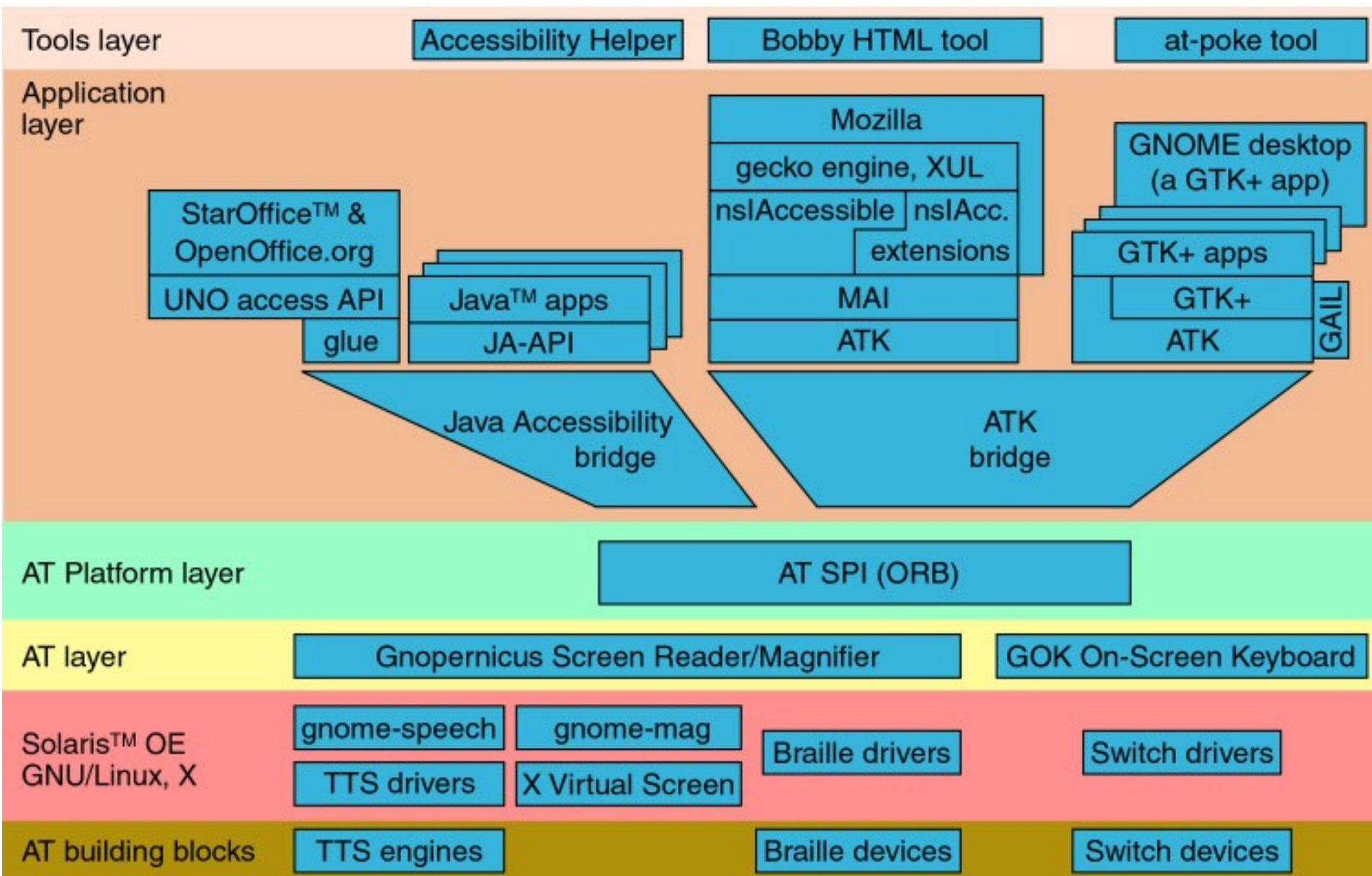
AT with StarOffice in Windows



AT with StarOffice in UNIX



Desktop Accessibility Architecture



How the architecture works

- In-process definition and implementation of Accessibility API
- Use CORBA for IPC to AT
- Have a small piece for discovery, window mgr./desktop stuff

A closer look at the API

- Accessible object
 - > Name, Description, Role, States
 - > Parent, Children, location in Parent
 - > Relations to other Accessible objects
- Set of Accessible sub-interfaces:
 - > Component: visual/rendered information
 - > Action: Manipulate objects (“click”, “PgUp”)
 - > Selection: add/remove objects from a selection
 - > Text, EditableText, HyperText: rich text interfaces
 - > Table: 2D array of children, (x,y) access
 - > Image: get icon embedded within an object
 - > Streamable: future

API example in Java

JSlider



implements Accessible

`getAccessibleContext()`



Accessible Context

implements AccessibleComponent

implements AccessibleValue

`String getAccessibleName()`

`String getAccessibleDescription()`

...

AccessibleValue

`int getCurrentAccessibleValue()`

`int getMaximumAccessibleValue()`

`int getMinimumAccessibleValue()`

AccessibleComponent

`Color getBackground()`

`Color getForeground()`

...

DEMO

at-poke

engineered accessibility, or
“Access by Contract”

Implications

- No off-screen model for screen reading
- Dynamic voice recognition grammars
- Automated testing of user interfaces
- Clean path to Web 2.0 / Rich Web Applications via WAI-ARIA
- Ability to develop desktop-wide LD apps like WYNN and TextHelp



Access by Contract: The New Way to Make Technology Accessible

Peter Korn
peter.korn@sun.com

