Accessibility with Open Source

Klaus Knopper

Presentation of 11.12.2007 at the NUUG

The Problem

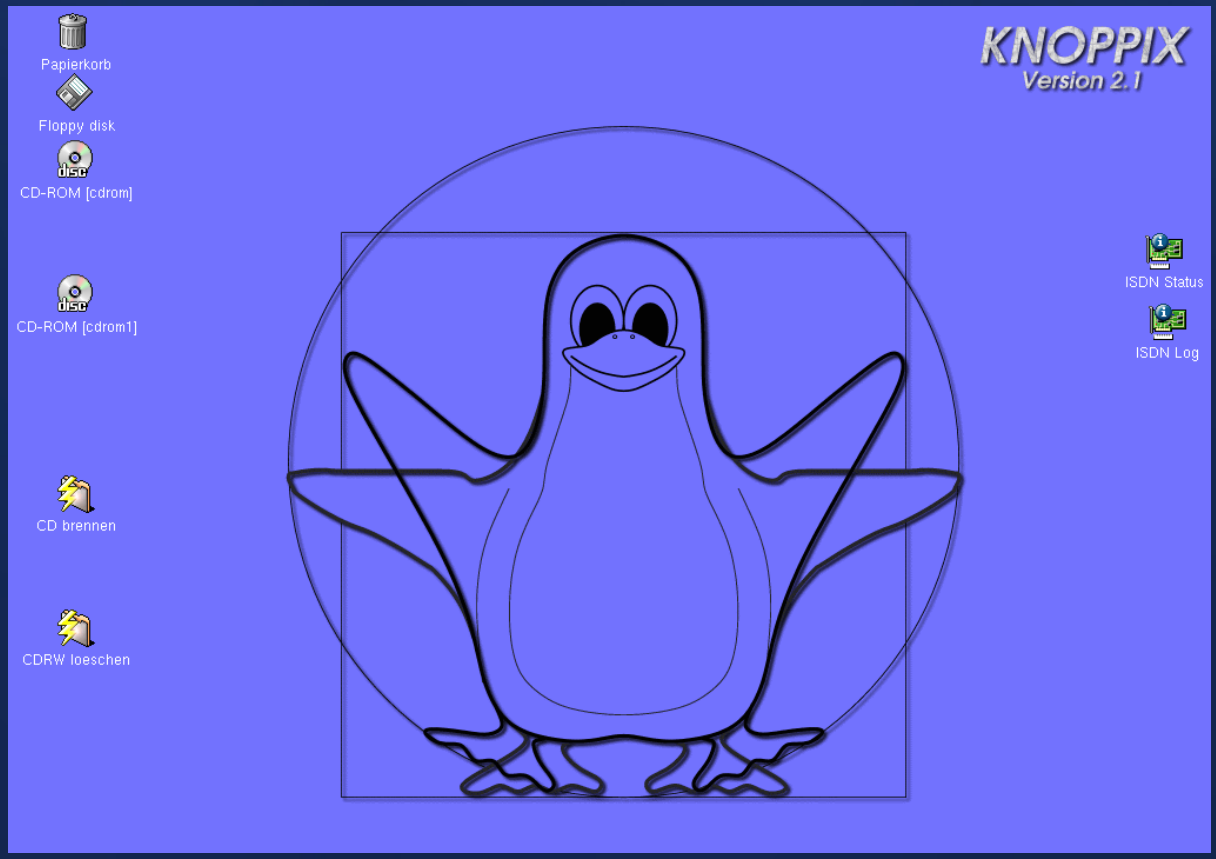
The “statistical average standard user’s“ mouse-oriented desktop

- ...gives you a visual overview of running programs and options,
- ...allows you to do many things in parallel,
- ...has many icons and menus that you just have to click to start a program,
- ...you don’t have to know or type any commands or options.*)

*) though many Shell users will say that’s not an advantage.

GNU/Linux has many beautiful desktops

...some of them are easy to use.

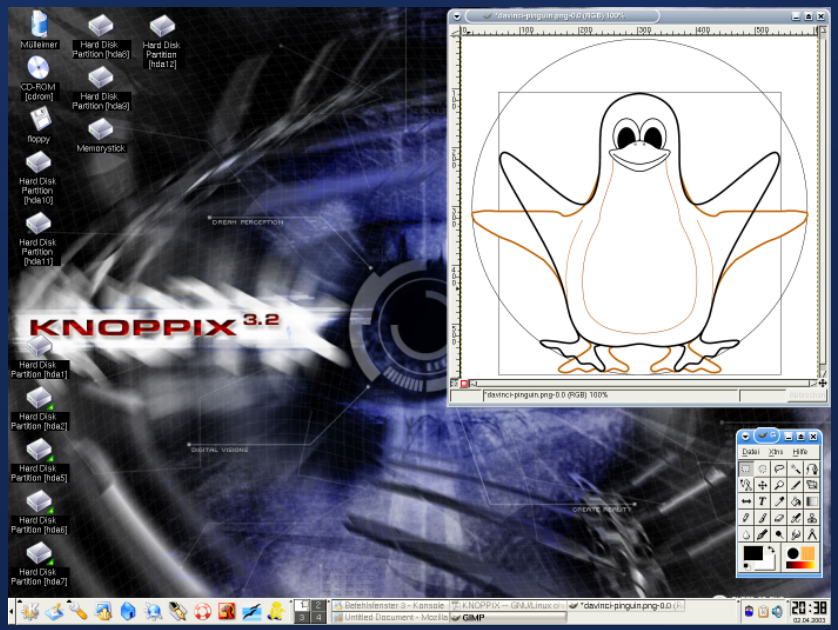


GNU/Linux has many cool desktops

...some of them have wobbly, flying windows and rotating cubes.

The problem

But if you take away all the fancy graphics...



...you get this:

And now, try to send email, or go to a web page.

Overview

1. Definitions,
2. Examples,
3. Solutions/Toolsets,
4. Trends.

Definition: Accessibility

Wikipedia: *Accessibility is a general term used to describe the degree to which a system is usable by as many people as possible.*

This very generic description is independent from computers, yet computers can help to make some things accessible that wouldn’t be without.

(Some) types of handicaps

- impaired/no hearing[1)]
- low vision
- *no* vision (from birth, or later)
- coordination (movement, mobility)
- mental (understanding, attention)
- low/insufficient skills[2)]

Each disability requires a [totally] different approach to remove barriers.

[1)] In computer usage, this type of disability is sometimes disregarded, since most programs don't carry essential information in audio formats. But for teaching, it can still be very important to represent sound in a different way.

[2)] Not really a „disability", but a problem that can be solved in similar ways.

The general problem

Abstraction layer missing for separating

- Content (information)
- Presentation (view/metadata)

which also means (as an example), missing application-independent layer for an „interaction interface“ that can by used by graphical as well as non-graphical GUIs and could be configured on an individual „capabilities“ base.

Approaches of accessibility

- Hardware layer: Letting additional hardware do the job*)
- Application layer: Building accessibility improvements directly into individual programs
- OS/System-Layer: Plugins, Daemons, Patches, Tricks & Workarounds
- Library/middleware layer: Linking with accessibility libraries and (actively) calling interface functions thereof

*) Insufficient in most cases, sometimes a lazy excuse.

Solutions/Hardware (1)

Braille device: Reading by touch

Optical magnification: Reducing noise, enhancing focus, brightness and contrast

Hardware speech synthesizer: Reading texts from a virtual screen or printer

I/O devices: (Joy-)stick-alike or optical measurement (head-mouse) instruments for operating/navigating, possibly with force-feedback

Sound/fingerprint-operated relays: Input by personal trained hardware/embedded software addons that emulate standard input functions

Problem: Not possible to interact with the software in the same way, by just emulating „normal“ I/O.

Solutions/Software (2)

Application layer: Adding accessibility functions directly into individual programs

- ➪ Games/Educational software for kids or people with low or no reading/writing skills (frozen-bubble, gcompris, OLPC system software),
- ➪ Active textreading support, cut&paste „Read alout“ functions (konqueror, kate editor-plugins),
- ➪ Desktop accessibility plugins/daemons (beryl/compiz-fusion, xzoom, gnome-magnifier)

☞ Pro: Program has the complete information about object description and action handling.

☞ Contra: Very inefficient and inconsistent if trying to rewrite each individual program for accessibility this way.

Solutions/API-Level (3)

OS/System layer: Patching/hacking the operating system or Desktop

- ➪ „Binary patches“ of system components to extend program capabilities (Jaws™ for Windows™),*)
- ➪ Kernel- or library-Patches (Speakup, X11R4-hacks)
- ➪ API extensions or plugins that add an interface to I/O plugins for accessibility, such as ATK, AT-SPI for the GTK+ widget set.

☞ Pro: Does not require changes in the enduser software itself.

☞ Contra for proprietary system software: Not all functions supported, likely to break with the next OS update, requires in-depth knowledge of „secret“ OS functions.

compiz-fusion Accessibility Plugins

... enhance the graphical desktop especially but not only for users with low vision by:

- follow-focus magnification,
- color switching / colorblind modes,
- transparency, brightness change and blurring to draw attention on active elements,
- annotation drawings, overview/preview,
- and of course also some less functional „fun effects“.

... a demonstration will follow later.

ADRIANE

A udio

D esktop

R eference

I mplementation

A nd

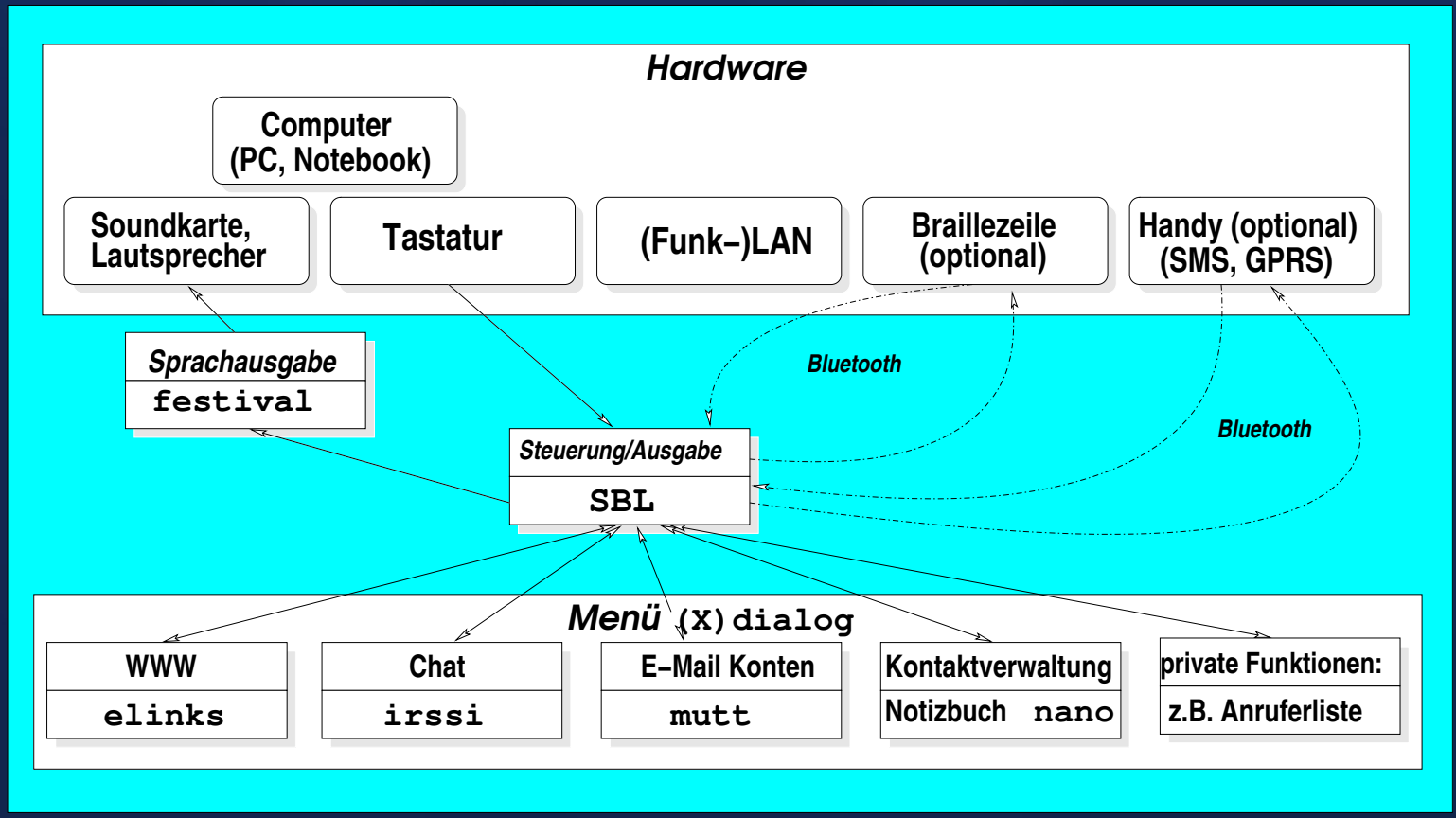
N etwork

E nvironment

Eliminates dependency of viewing the screen, and adds two more options of information reception, by hearing and (optional) by touch, focused at blind users and „non-techies“.

Middleware architecture

as an example of a textbased screenreading/navigation system.



Libraries and APIs

...for programs to enable interaction with screenreaders and other accessibility-focused software ((k)tts, orca[1]).

- ATK
- MSAA/IAccessible/IAccessible2
- Near future: Also QT4 w/ dbus?

☞ Pro: Standardized interfaces, easy to implement with a few changes, IF the software was designed in a GUI-independent way.

☞ Contra: For really „old“ or totally vision-focused programs, MANY changes or a complete redisign of the GUI parts are required, different interfaces for GTK+/GNOME- and KDE API based programs.

---

[1] RIP: **gnopernicus** New: **orca**

„Screenreaders“ (1)

... are actually more than just readers.

- Text/Console-based:
  - **`brltty`**
    - Braille-device focused,
    - supports (optional) text-to-speech.
  - **`sbl`** / **`sbl+brld`** (3.0)
    - navigation and output independently,
    - braille device and/or keyboard navigation mode,
    - profiles with auto-detect for different applications,
    - text-to-speech with speech-dispatcher,
    - braille API mode for other screenreaders.

„Screenreaders“ (2)

- Graphical/Widget-oriented
  - **`orca`**
    - currently works with GTK+ AT-SPI extension,
    - speaks/displays labels, text and meta-information (i.e. „type of button“) of elements,
    - mostly architecture-independent python,
    - optional magnification, braille, input accesskey support,
    - although heavy developmet is done to improve support of different programs with profiles, already works well with complex programs such as OpenOffice and firefox*).

Graphical screenreaders

*) But: What is shown on the graphical screen is still totally different from the blind persons perception of the user interface.

☞ Misunderstanding on employers side: “All users must be able to use the same programs on the same desktop.“

☞ While it is *possible* for a blind person to use vision-oriented graphical programs with speech and braille, it is not as *efficient*, and sometimes very painful, to use an interface that does not match the individually best way of working with software.

☞ Each user, in theory, requires a customized desktop and tools that match his/her way of working with the computer in the most convenient way.

☞ With OSS, we have the possibility to customize everything!

Webpages (1)

...and other „non-software“ information resources.
Accessibility Guidelines (as recommended by the W3C and other organizations):

- Strict separation of content and meta-information,
- **ALT** and **DESCRIPTION** tags for pictures,
- no „active scripting’nnavigation as the ONLY option,
- easy navigation, maybe a „skip to main content“ element,
- using structural tags for structure, not for enhancing the graphical layout.

Most of these things require some knowledge about the way that XML and HTML wrap up content/infrmation, and most GUI-based web editors still do it WRONG.

§§§ Law §§§

[1)] Germany: BGG §11 *requires* governmental websites to provide barrier-free content, and commercial websites are *strongly encouraged* to do the same.

Webpages (2)

...and other documents:
For some disabilities, it is not a matter of reading the raw information, but of reduced complexity.

- Alternate text with less „information overhead“ and shorter, context-free content,
- Avoiding „marketing language“ or „expert terms“,
- No more than one „viewpage“ per screen, i.e. no scrolling necessary (for example, on a standardized 800x600 pixels browser).

Other Document formats

The general rule is: Information (even if graphical ones such as diagrams) should be transported with descriptive tags, rather than a purely graphical layout focused on the „statistically average document reading person“.

Open Document Format: (ODF, ISO/IEC 26300), is an XML-based format that separates information from presentation by standardized XML-Tags as containers. It can be read even without OpenOffice.

LaTeX: Text editing as plaintext with macros, output in various formats (most popular: PDF, HTML).

HTML: in the way specified in the W3C documentation, is a very well accessible format, unless elements such as tables are misused for graphical layout.

Problem: Legacy applications

```
import java.applet.*;
import java.awt.*;

public class Text extends Applet {
  public void paint(Graphics g) {
    g.drawString("Hello World", 5, 25);
  }
}
```

Even with this simple Java-Applet, the screenreader has no chance to read the visible „Hello, World“ text from the browser window - because there is no text!

New trends: KDE4, DBUS

- QT4 (KDE) has a new, consistent abstraction layer for accessing all interactive elements from DBUS.
- „Everything that can be scripted, is (probably) also accessible“ -KK
- Accessibility as governmental requirement for the workplace is greatly driving development.

Conclusion

- Like common in OSS, we have a huge set of tools and libraries available that's heavily improved and integrated into the various desktop systems.
- Ready-to-use „complete products“ are very specific to the users needs, and not easy to find, though most vendors integrate accessibility tools especially for vision-impaired persons into their desktops and installation routines.
- Accessibility interfaces are alreay standard in the main Desktop APIs, though GNOME/GTK+ is somewhat ahead until KDE4 is released.
- For some users (not only blind), non-graphical environments are still the most *efficient* way of working, because they are not designed for vision-oriented work, and therefore need less „workarounds“.

Links

[1] http://www.linaccess.org/
Barrier-free GNU/Linux research portal (WIP)

[2] http://accessibility.kde.org/
The KDE accessibility homepage

[3] http://live.gnome.org/Orca
Python-based screenreader for graphical programs

[4] http://www.sun.com/software/star/gnome/accessibility/architectur
The SUN/GNOME-2.0 accessibility homepage

[5] http://knopper.net/knoppix-adriane/index-en.html
Homepage of the ADRIANE project

[6] *In the Beginning... Was the Command Line*
An absolutely worth-reading essay about how graphical desktops do not make your life easier, by Neal Stephenson, ISBN: 0380815931, 978-0380815937