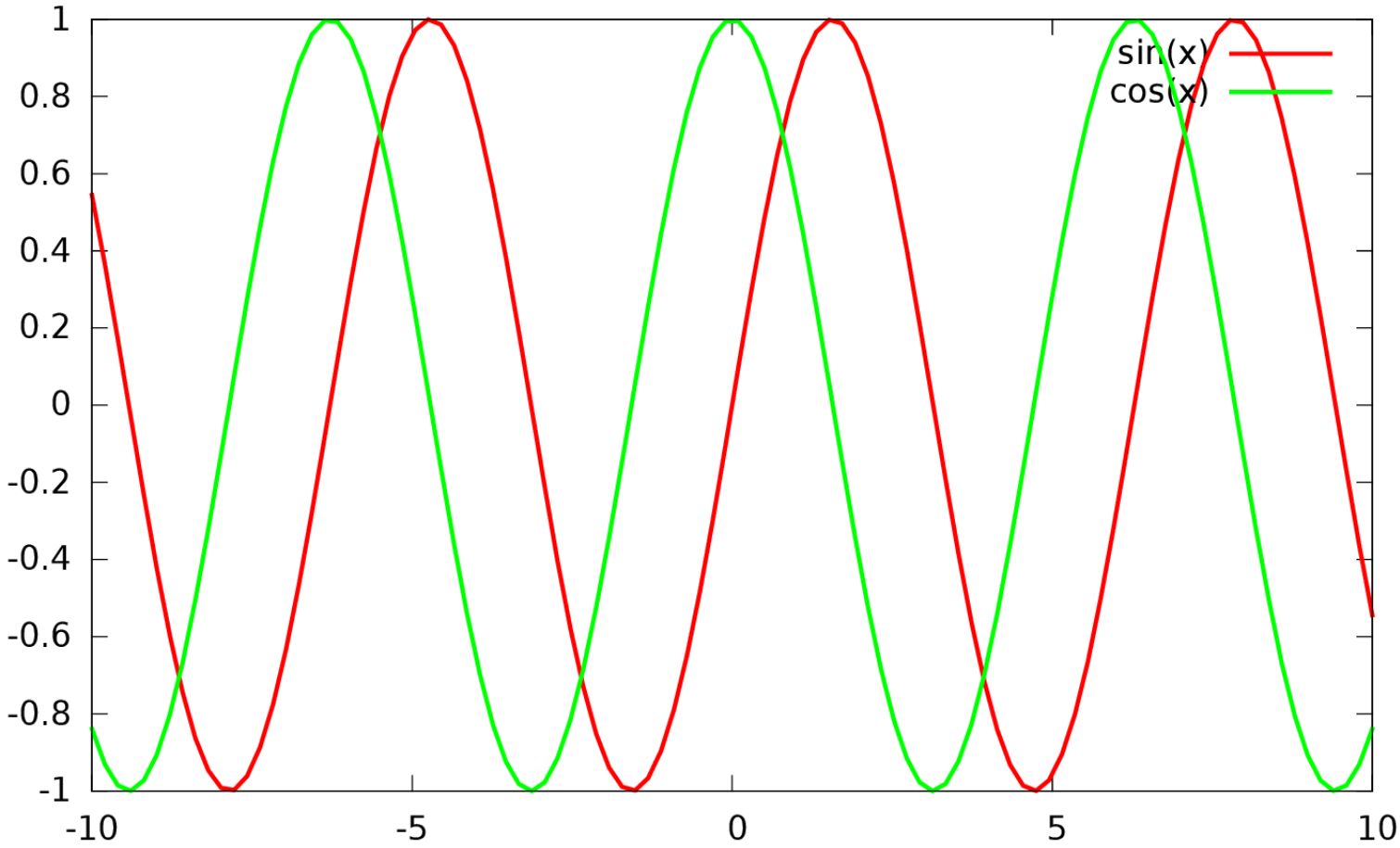
Why not extending non-regression tests to accessibility matter?

Samuel Thibault
Slides & stuff on
http://brl.thefreecat.org/

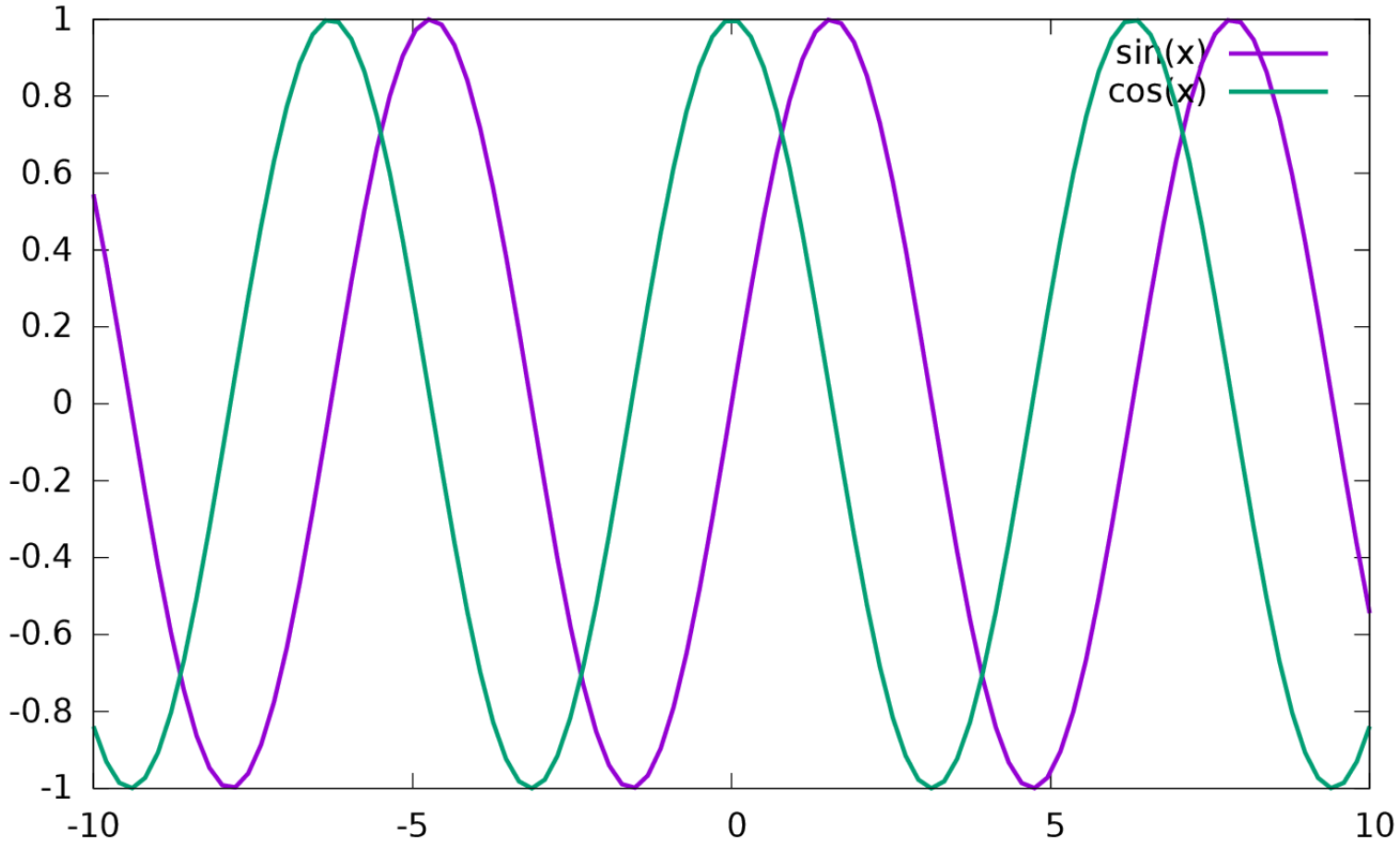
http://liberte0.org/

Gnuplot



Color blindness: 8% male, 0.5% female

Gnuplot 5!!



Color blindness: 8% male, 0.5% female

What is accessibility?

AKA a11y

Usable by people with specific needs

- Blind
- Low vision
- Deaf
- Colorblind
- One-handed
- Cognition (dyslexia, attention disorder, memory, ...)
- Motor disability (Parkinson, ...)
- Elderly

See Accessibility HOWTOs

- You

"Handicap" depends on the situation and is not necessarily permanent
10% handicapped – 20% limited

This is all about freedom #0

"The freedom to run the program, for any purpose"

What about being *able to use* the program?

- RMS said a11y was just a "desirable feature".
  - "Desirable" only, really?
- RMS said "this is free software, you can modify it" (freedom #1)
  - Can. Not. Happen.

UNO rights of persons with disabilities

"Discrimination on the basis of disability" means any distinction, exclusion or restriction on the basis of disability which has the purpose or **effect of impairing** or nullifying the recognition, enjoyment or **exercise**, on an equal basis with others, **of all human rights and fundamental freedoms** in the political, economic, social, cultural, civil or any other field. It includes all forms of discrimination, **including denial of reasonable accommodation**

"Reasonable accommodation" means necessary and appropriate modification and adjustments **not imposing a disproportionate or undue burden**, where needed in a particular case, to ensure to persons with disabilities the enjoyment or exercise on an equal basis with others of all human rights and fundamental freedoms;

A question of priority

- Should be prioritized
  - Just like internationalization

A question of who doing it

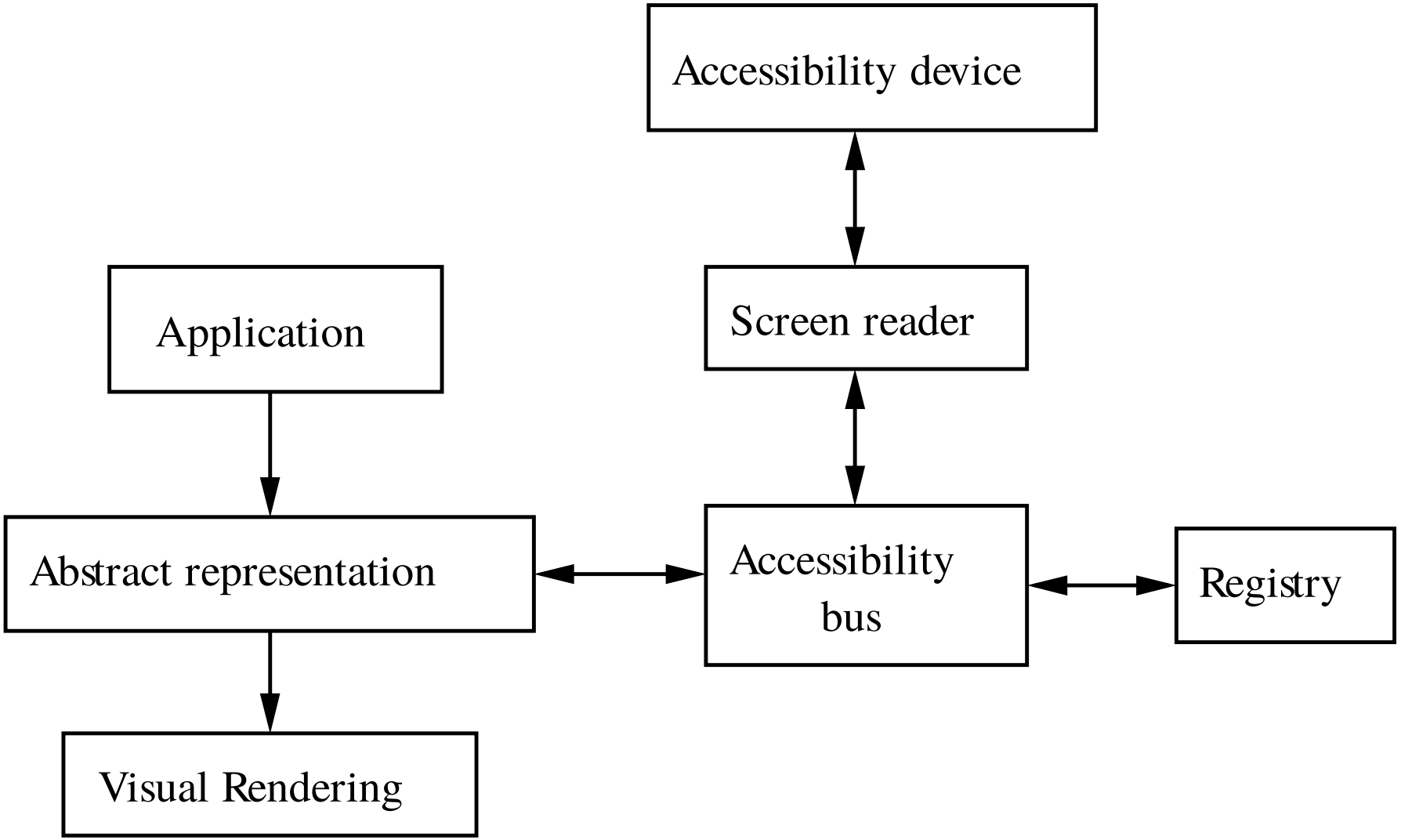
- Concerns only a small fraction of population
  - Already a hard time using computers...
  - Almost nobody with both disabilities and programming skills (and very difficult to work)
  - Even fewer people with awareness and programming skills

    → “This is free software, you can modify it” can not work.
- Support has to be integrated
  - Distributed among maintainers themselves
  - Not borne by the tiny a11y community

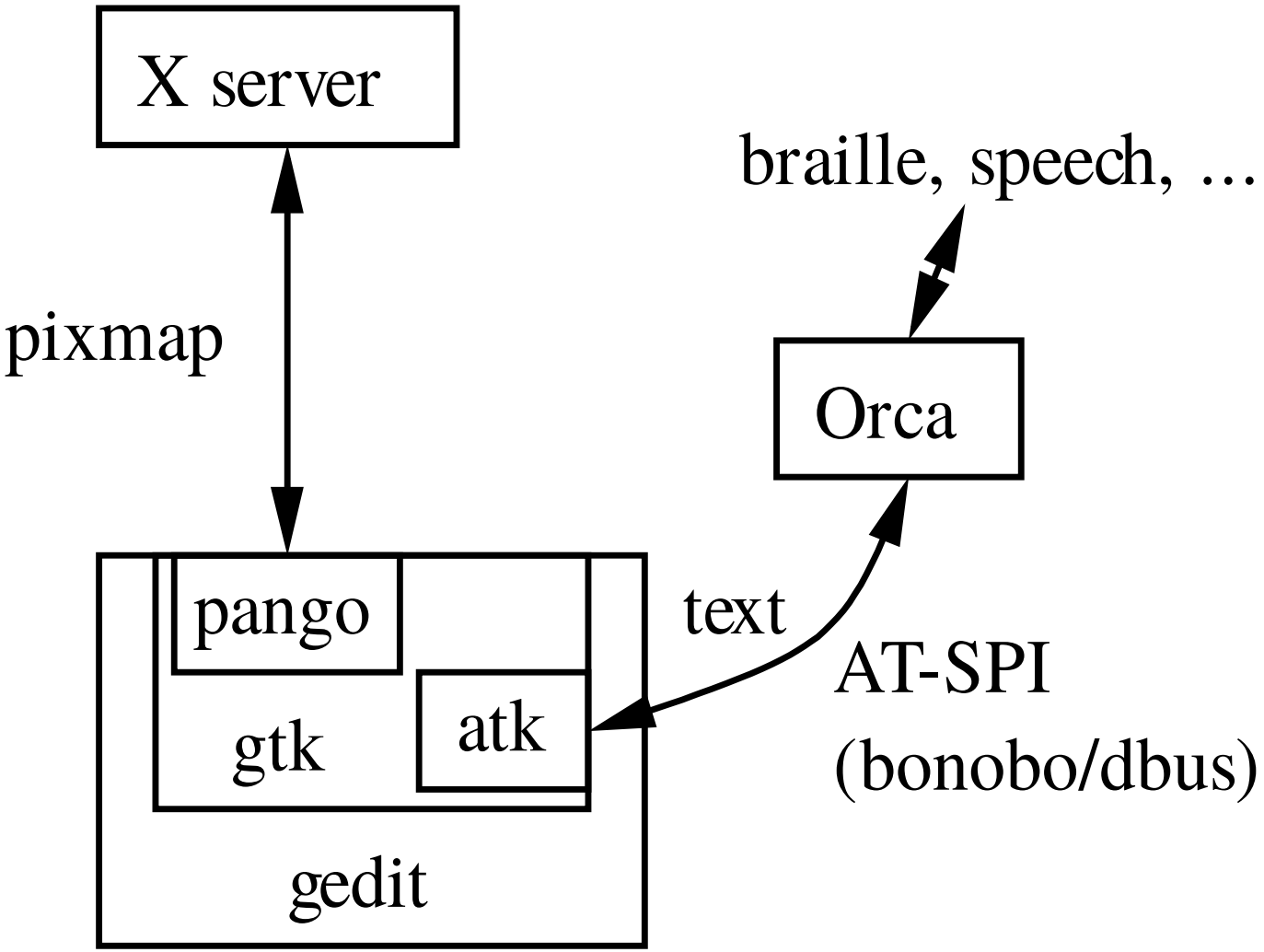
Design principles

- Same software, made accessible
  - Understand each other, get help, etc.
- Synchronized work
  - Just alternate input/output
  - Being able to work together
- Pervasive
  - Shouldn't have to ask for software installation / configuration

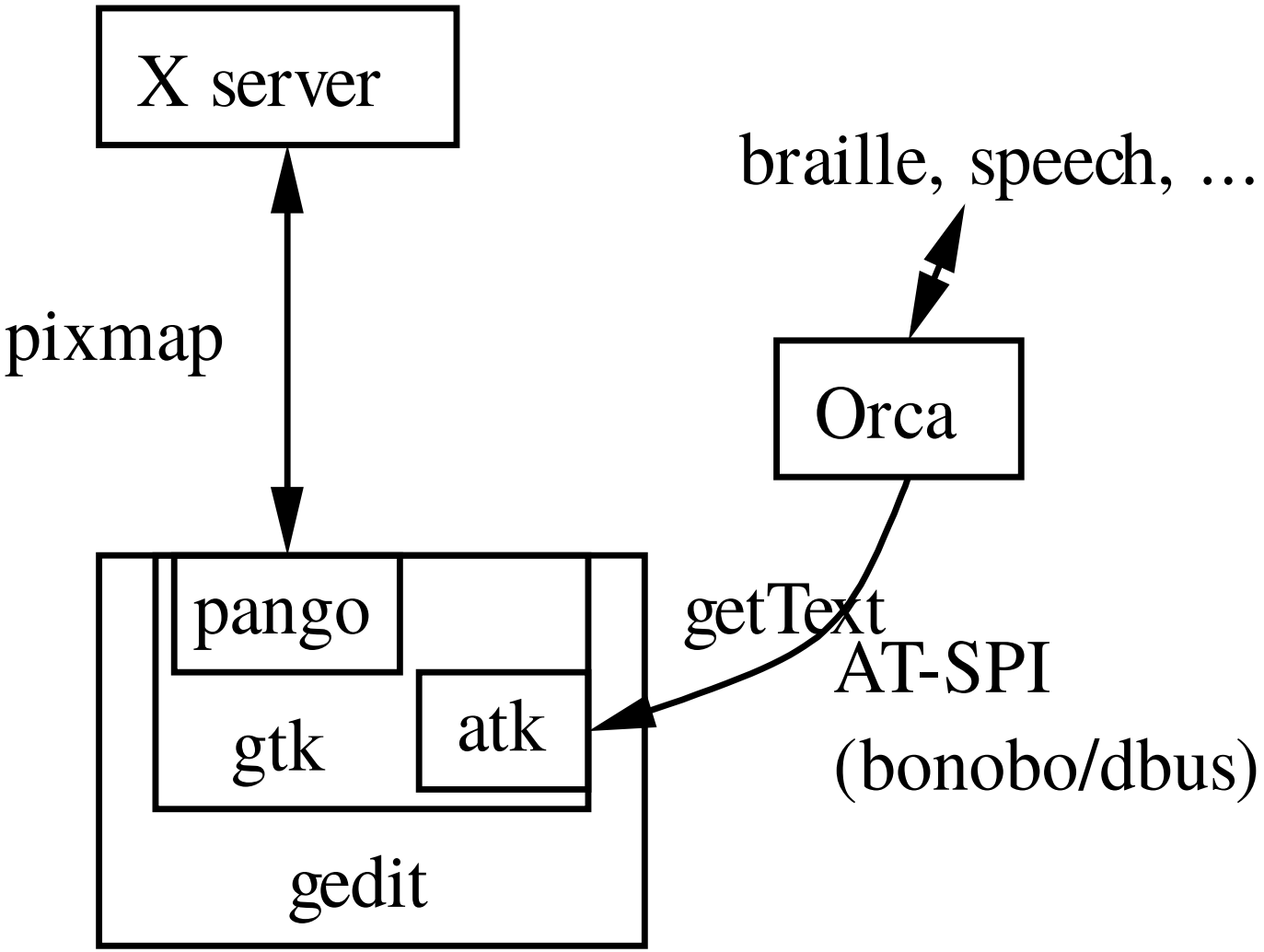
Generic methodology



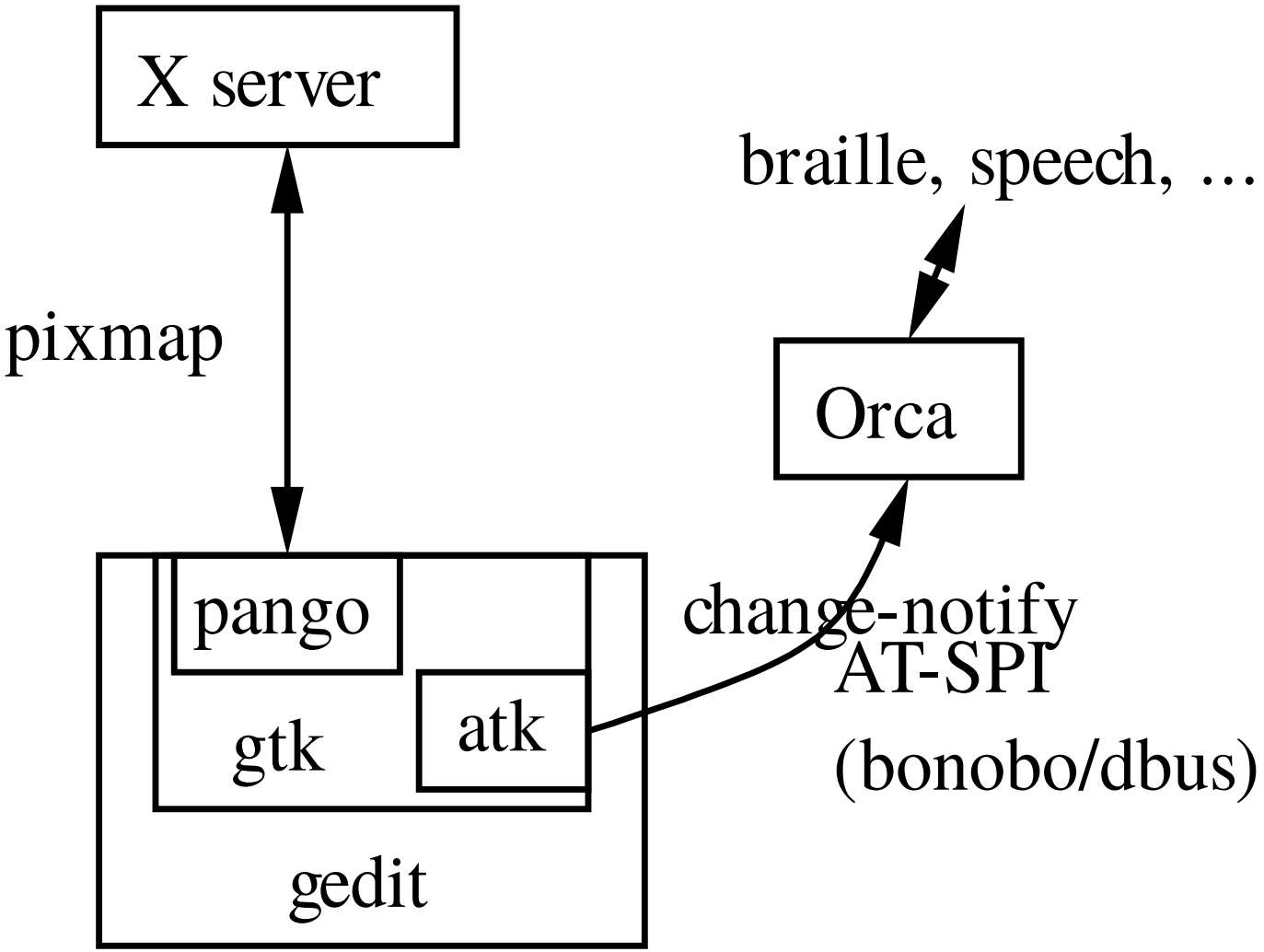
X accessibility, AT-SPI RPCs



X accessibility, AT-SPI RPCs



X accessibility, AT-SPI RPCs



Abstract representation

- Window
  - Vertical container
    - Menu bar
      - File Menu
        - Open Menu Item
        - …
      - ...
    - Horizontal container
      - Text area
      - Ok button

Technically speaking

A lot of applications already *technically* accessible

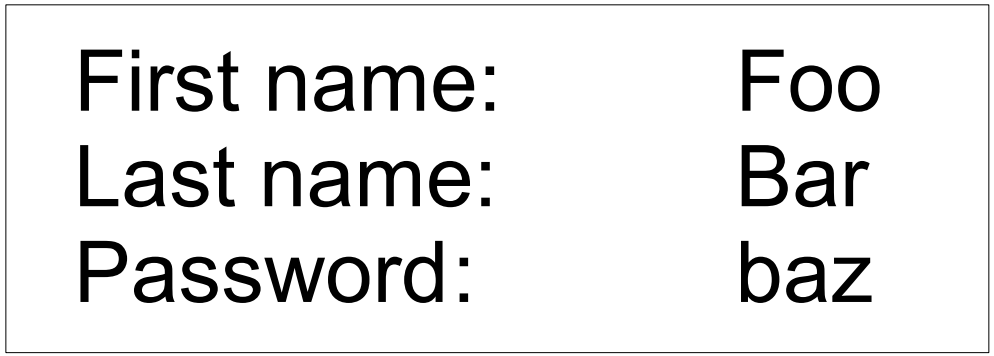
- Console
- GTK2/3
- KDE-Qt4 sketchy, Qt5 improving
- Java Swing
- Acrobat Reader

A lot are not

- Mono?
- KDE-Qt3
- Xt
- Self-drawn (e.g. xpdf)

In practice

- A lot of technically-accessible applications actually aren't really usable
  - A visually-organized mess of widgets...



In practice

- A lot of technically-accessible applications actually aren't really usable
  - A visually-organized mess of widgets...

First column
- Label First Name
- Label Last Name
- Label Password

Second column
- Text Foo
- Text Bar
- Text baz

In practice

- A lot of technically-accessible applications actually aren't really usable
  - A visually-organized mess of widgets...

- Label First Name for Text Foo
- Label Last Name for Text Bar
- Label Password for Text baz

In practice

- A lot of technically-accessible applications actually aren't really usable
  - A visually-organized mess of widgets...

First column
- Label First Name
- Label Last Name
- Label Password

Second column
- Text Foo
- Text Bar
- Text baz

In practice

- A lot of technically-accessible applications actually aren't really usable
  - A visually-organized mess of widgets...

First column
- Label First Name
- Label Last Name
- Label Password

Second column
- Text Foo
- Text Bar
- Text baz

➔ Screen reader “Script” for each application

Graphical applications

- Design your application **without** gui in mind first
  - Logical order, just like CSS ☺
- Use standard widgets
  - e.g. *labeled* text fields
  - Avoid homemade widgets, or else implement atk yourself for them
  - Always provide alternative textual content for visual content
- Keep it simple!
  - Not only to make screen reading easier, but to make life easier for all users too!

Test it yourself! (GUIs)

orca -e braille-monitor

- Then work as usual
- Only using keyboard
- Checking text appears there

And crash-test

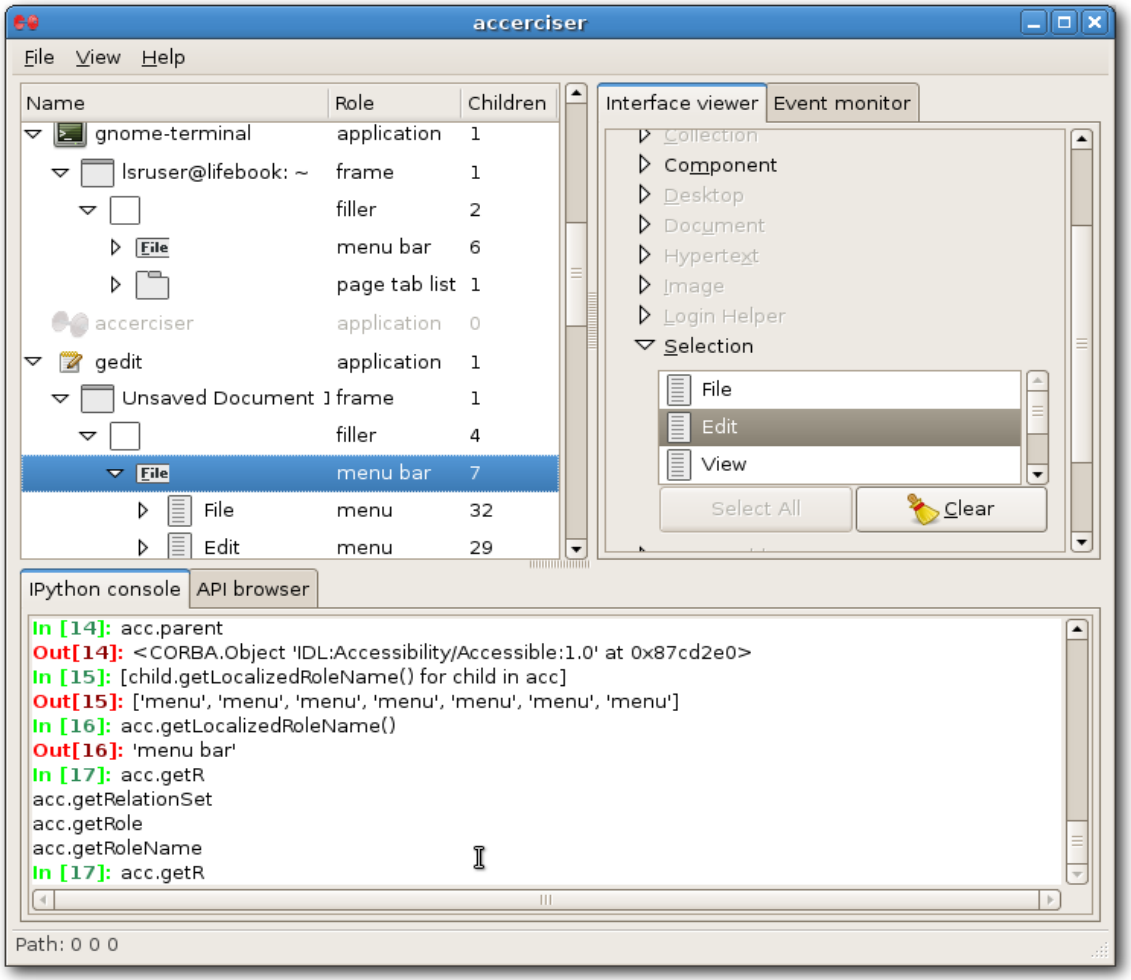
- Turn on speech, switch off the screen

https://developer.gnome.org/accessibility-devel-guide/stable/

Test it yourself! (GUIs)

Accerciser

- Sort of debugger
- Tree of widgets
- Properties



https://developer.gnome.org/accessibility-devel-guide/stable/

Regressions

- One step forward, two steps backward
- New features break old features
- Bug fixes break features
- So many potential regressions
  - User testing
  - Regression testing

Accessibility regressions

- One step forward, ten steps backward
- New features just unusable
- Bug fixes break usability
- User testing
  - So few concerned users
  - Difficult to test development versions
  - Usual users have no idea how to test it
- ➔ Regression testing?

Regression testing

New features: making sure they might be usable

Old features: making sure they remain usable

- All widget have some label
  - Except layout widgets & such
- All labels are attached to some widget
- All widgets are reachable with some keyboard shortcut

Testing widget labelling

Online with running application

- Expensive
- Relating results with source line?

Testing widget labelling

With glade UI (XML file)

- Browse XML widget trees
- Easy to embed in Makefiles

- False positives
- Use “suppression” files

Testing widget labelling

With C/C++/… source code

- Uh
- Source parsing?
  - Uh

Testing keyboard shortcuts

Checking existing shortcuts

- Online with running application
- Scenarii
  - E.g. all documented shortcuts

Testing keyboard shortcuts

Checking features are reachable with shortcuts

- Making widget shortcuts mandatory? Uh
  - Only widgets unreachable in a standard way?
- Note difference between
  - convenient, and
  - inaccessible
- Simulate banging at the keyboard arrows/tab/…
  - And see which widgets could be reached…

Conclusion

- Accessibility is a concern for a lot of people
  - 10% have major concerns
  - 20% have minor concerns
- Dealing with it usually boils down to common sense
- It very often actually also helps other users
- But we need to raise awareness of this