

What does "buildable" mean during a toolkit migration?

Sugar, GTK3 to GTK4 and X11 to Wayland

Krish Pandya, Sugar Labs
Build Systems MC



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Last December, in Tokyo

At [GNOME Asia 2025](#) I talked about porting the Sugar toolkit to GTK4. One slide ended with this:

"Contributors no longer have to understand Autotools to work on Sugar."

For the toolkit, that was true. Meson for the C library, pyproject for the Python, `pip install -e .` and you are in.

Then this summer a contributor started on the shell. It took the first five weeks just to get it to import.

So, what happened??

Dec 2025, Tokyo: the toolkit builds

summer 2026: five weeks to import the shell

Oct 2026, Prague: this talk



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



What even is Sugar?

A desktop and a bunch of learning apps for kids. It started on the OLPC laptop and has been in tree since April 2006.

TOOLKIT

The Python library everything else is written against. Windows, toolbars, palettes, icons. sugar3 is GTK3, sugar4 is GTK4.

SHELL

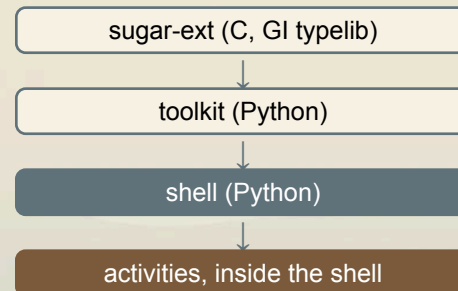
The desktop itself (jarabe). It launches the activities and shows them inside its own window.

ACTIVITIES

The apps. The set we are porting right now is called Fructose.

SUGAR-EXT

A small C library with GI bindings, for the bits Python wasn't able to do earlier (gestures, clipboard, session hooks).



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

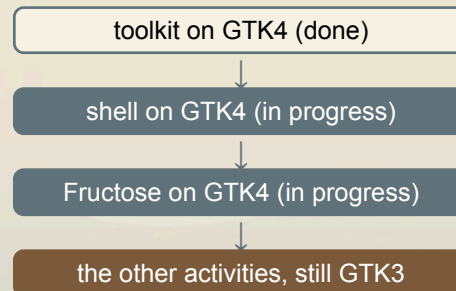


We have done this before

GTK2 to GTK3 ran from October 2011 to the 0.98 release in November 2012. About thirteen months.

This time it is GTK3 to GTK4 and X11 to Wayland at once. We are in year two, across twelve repositories. I ported the toolkit, and this year I mentor the two people porting the shell and Fructose.

The code side has a migration guide. The build side has nothing, and it has costed us more time than the porting.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

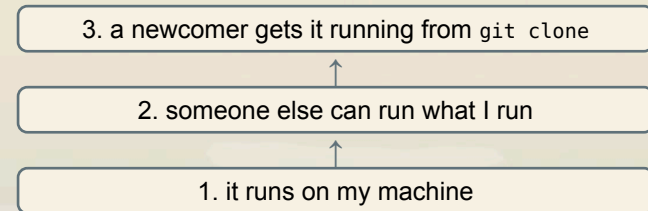


So what does "buildable" mean?

I used to think it meant "it compiles". That is the easy part.

In a stack that is half GTK3 and half GTK4 there are at least three rungs, and we broke on every one of them.

The toolkit was buildable in the way I meant in Tokyo.
The shell needs all three.



**LINUX
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026



What the shell asks for

Eleven `gi.require_version` calls, straight from the shell:

```
gi.require_version('Gtk', '4.0')
gi.require_version('Gsk', '4.0')
gi.require_version('Graphene', '1.0')
gi.require_version('Gtk4LayerShell', '1.0')
gi.require_version('GtkSource', '5')
gi.require_version('Casilda', '1.0')
gi.require_version('Gst', '1.0')
gi.require_version('Soup', '3.0')
gi.require_version('WebKit', '6.0')
gi.require_version('TelepathyGLib', '0.12')
gi.require_version('SugarExt', '2.0')
```

Ten come from the distribution. One is ours.

Each of these is a typelib on disk, and each one pulls in the typelibs it was built against. `GtkSource-5` pulls in `Gtk-4.0`. Our old `SugarExt-1.0` pulled in `Gtk-3.0`.

`require_version` pins the namespace you ask for. It says nothing about what that typelib pulls in after.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



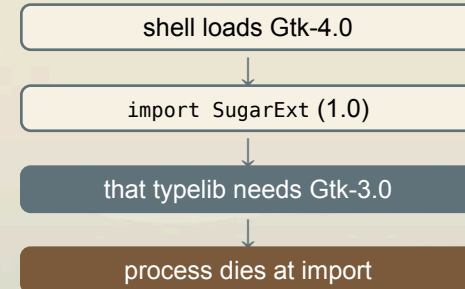
Typelibs that disagree

Our C library still shipped as SugarExt-1.0, and the installed typelib was built against GTK3. The shell loads GTK4 first, then imports SugarExt. The process dies at import.

In C this would be a link error. In GI nothing catches it at build time or at package time.

And the typelib knows! It carries its dependency list, with versions, and you can read it without loading anything.

Nothing pointed at the problem. Finding it took some time of a contributor's summer.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



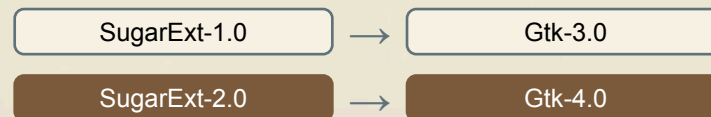
The fix was boring

Bump our namespace from SugarExt-1.0 to SugarExt-2.0, so the GTK3 build and the GTK4 build can't be mixed up. Same for SugarGestures. It went in with the autotools to Meson move (sugar-ext #6).

```
api_version = '2.0'  
...  
namespace: 'SugarExt',  
nsversion: api_version,
```

And the shell asks for it by version:

```
gi.require_version('SugarExt', '2.0')
```



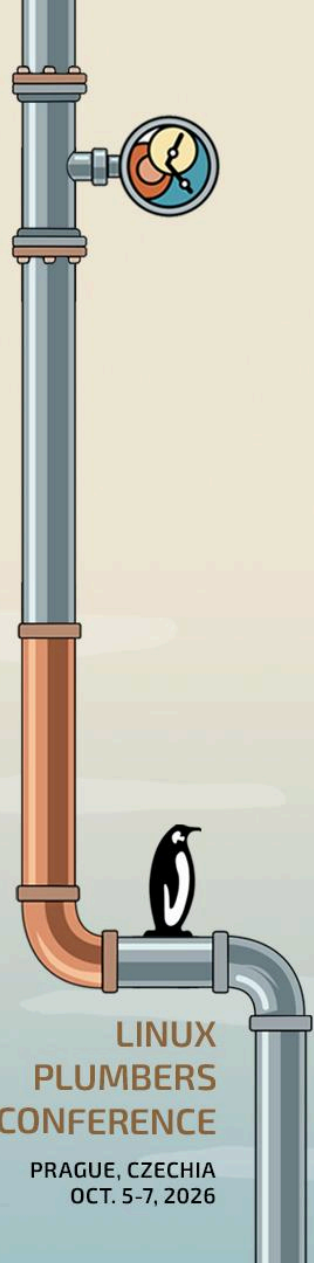
It works, and it is what I would tell the next project to do. Any GI-based project that carries its own introspected library through a toolkit change will hit this.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



A vertical pipe with a gauge at the top and a penguin perched on a horizontal section at the bottom. The pipe is grey with orange/brown sections. The gauge has a yellow face with a black needle. The penguin is black and white.

The typelib knows what it needs. Why do we only find out at import?

LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Which Casilda? Whichever nixpkgs lets you have

The shell puts a Wayland compositor inside a widget (Casilda). Activities run as Wayland clients inside the shell.

Our pinned nixpkgs has GTK 4.20.3 and wlroots 0.19.2. That pin is two layers away from our code, and it picks which Casilda we can build. Which Casilda we can build picks what a reviewer can run.

A contributor on Debian and a reviewer on Nix could not reproduce each other's bugs.

Casilda	wlroots	GTK	Builds on our pin
1.2.1	0.19	4.20	yes, the last one
1.2.2	0.20	4.20	no, wlroots
1.2.4	0.20	4.22.2	no, wlroots and GTK

The only place this is written down is a comment in our flake:

```
# Pinned to tag 1.2.1, the last release that targets  
# wlroots-0.19, which is what this nixpkgs provides.
```



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



git clone is not enough

ONLY IN THE RELEASE TARBALL

jarabe/config.py (built from config.py.in) and the compiled GSettings schemas. You make both by hand.

PACKAGED BY NOBODY

gwebsockets. No distribution ships it. Clone it, put it on PYTHONPATH.

THREE BUILD SYSTEMS AT ONCE

- autotools from 2006, in the GTK3 toolkit
- Meson, in sugar-ext
- pyproject, in sugar4

What it takes to start the shell from a checkout, inside the flake:

```
glib-compile-schemas "$ROOT/sugar/data"

export GSETTINGS_SCHEMA_DIR="$ROOT/sugar/data"
export SUGAR_GROUP_LABELS=...
export SUGAR_MIME_DEFAULTS=...
export PYTHONPATH="$ROOT/sugar/src:\
$ROOT/sugar-toolkit-gtk4/src:$ROOT/gwebsockets"

# Nix never expands $UID in a flake env string,
# so dbus got a literal "$UID" and dconf broke too
export DBUS_SESSION_BUS_ADDRESS=\
"unix:path=/run/user/${id -u}/bus"

exec python "$ROOT/sugar/src/jarabe/main.py"
```



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



How far does it boot?

With all three rungs, the GTK4 shell gets to the home view.

Then it stops while building the activities list. That list is still a `Gtk.TreeView`, deprecated since GTK 4.10. The fix is a `Gtk.ColumnView` rewrite. Voila!

The last step is a relief, it is the work we wanted to do. Every step before it was a build problem.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Back to the typelib question

So I went looking for what can check this.

You can already ask GLib what a typelib needs (`gi-inspect-typelib`, it used to be `g-ir-inspect`). Debian's `dh_girepository` uses that to fill in a package's dependencies. And GI does catch the clash, at runtime, which is exactly the crash we got.

What I could not find is anything that looks at all the typelibs one process is going to load and tells you before you run it. Maybe it exists and I missed it?

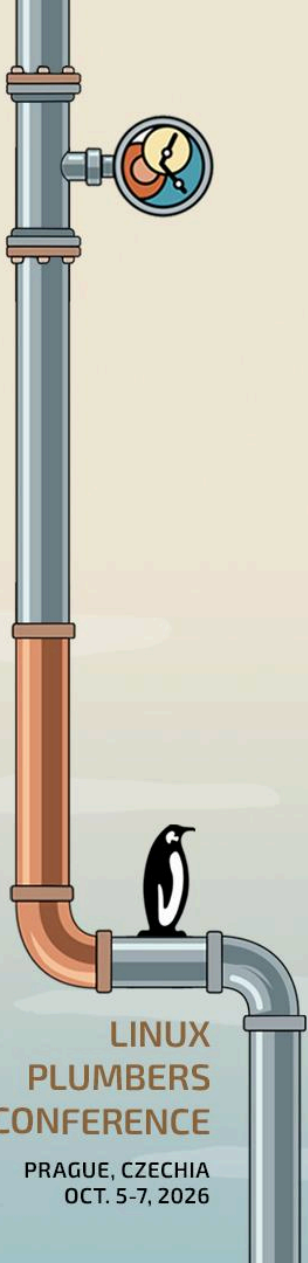
```
$ gi-inspect-typelib --print-typelibs \  
  --typelib-version=5 GtkSource  
typelib: PangoCairo-1.0  
typelib: Gtk-4.0  
typelib: freetype2-2.0  
typelib: GLib-2.0  
...  
typelib: Gsk-4.0  
typelib: Gdk-4.0
```



**LINUX
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026



A vertical pipe with a gauge and a penguin. The pipe is grey with orange sections. A gauge is attached to the top. A penguin is perched on a horizontal section of the pipe. The background is a light blue sky with white clouds.

So, was I right in Tokyo?

LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Kind of

Nobody has to understand Autotools anymore. That part held up.

But getting the shell to run still means knowing everything from the last few slides, and none of it is written down anywhere except our flake and my head.

What I want is boring. The next contributor's first week should be porting, not building!

3. a newcomer gets it running from git clone

2. someone else can run what I run

1. it runs on my machine



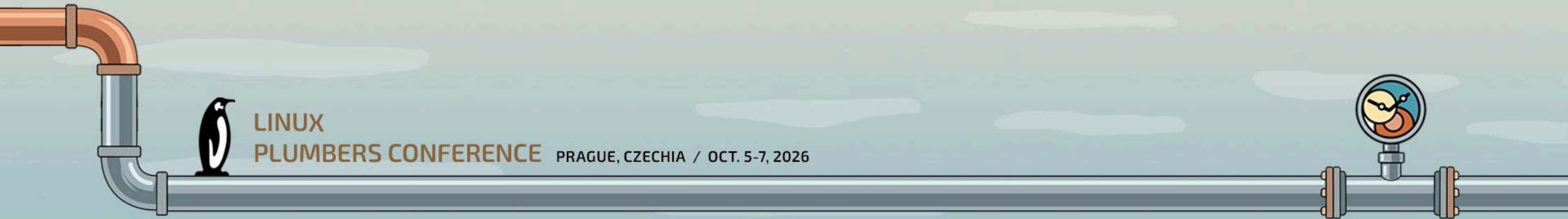
LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Questions

1. Should typelib agreement be a build-time or package-time check? Something that takes the set of typelibs a process will load and says whether they can live together. Does that exist, or is there a reason not to build it?
2. How should a build system or a distribution represent a stack that is half-migrated on purpose, where the right answer to "which version" differs per repository and changes every week?
3. What do you hand a new contributor so they can build a multi-repository stack in the middle of a migration?
4. Is there anything for coordinating the patches every distribution ends up carrying during an ecosystem-wide toolkit transition?



Links

- Shell migration, 188 files: <https://github.com/sugarlabs/sugar/pull/1106>
- sugar-ext, autotools to Meson and GDK4 event porting: <https://github.com/sugarlabs/sugar-ext/pull/6>
- Toolkit migration talk, GNOME Asia Summit 2025: <https://www.youtube.com/live/WZ63lQ-DsOA?t=14743>
- GTK4 migration guide: <https://docs.gtk.org/gtk4/migrating-3to4.html>
- dh_girepository: https://manpages.debian.org/unstable/gobject-introspection/dh_girepository.1
- g-ir-inspect (now gi-inspect-typelib in GLib): <https://manpages.ubuntu.com/manpages/resolute/man1/g-ir-inspect.1.html>

Thank you!

