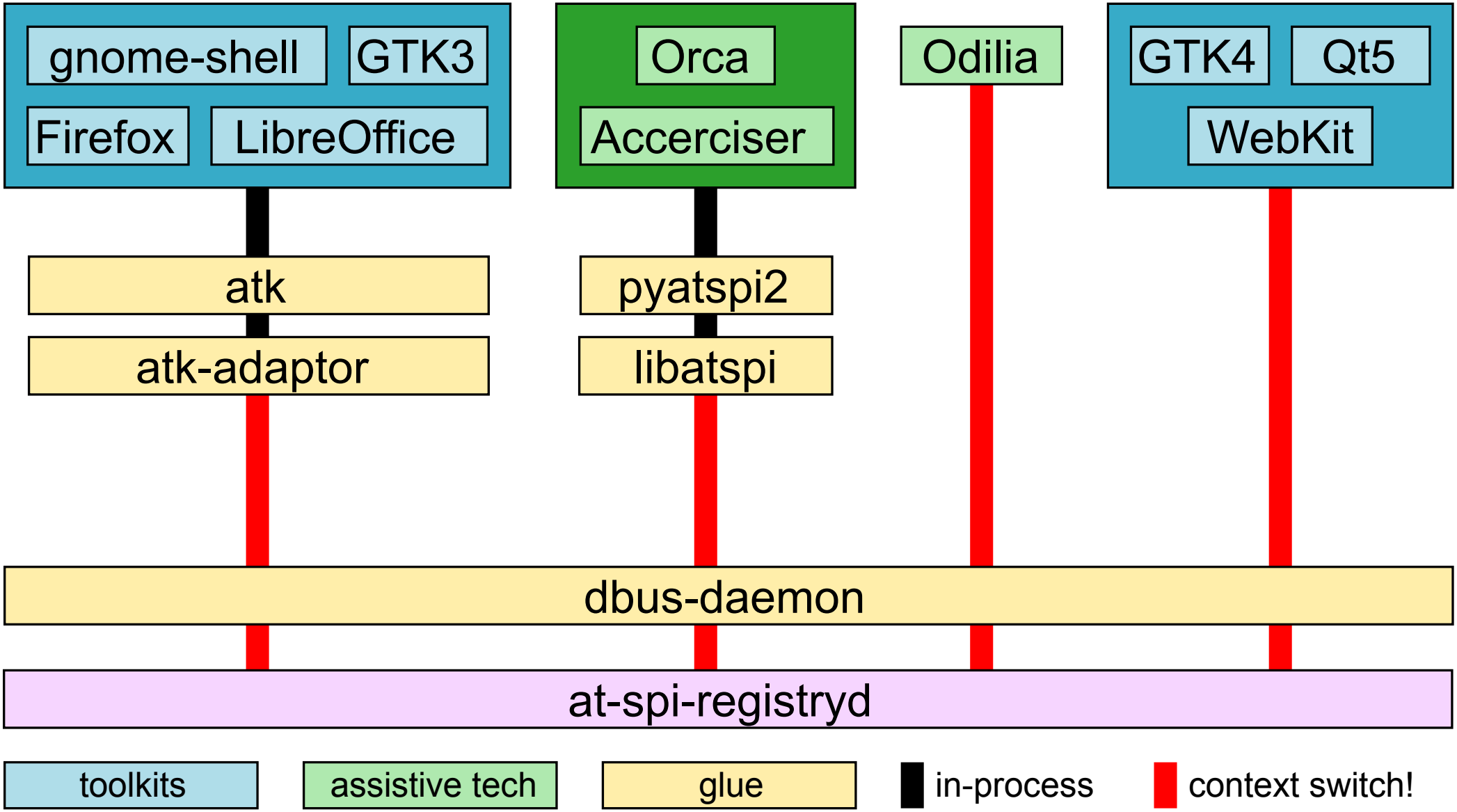
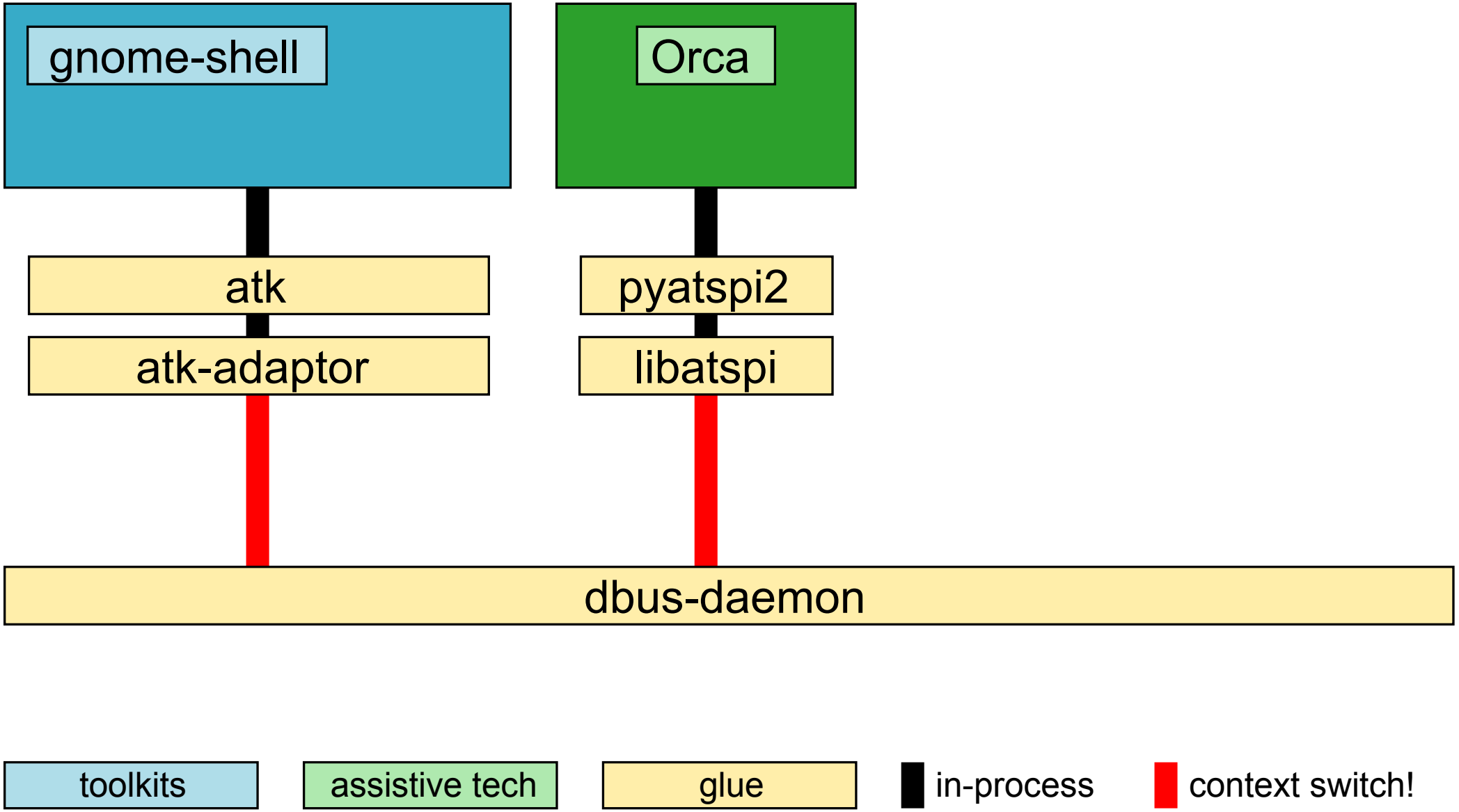
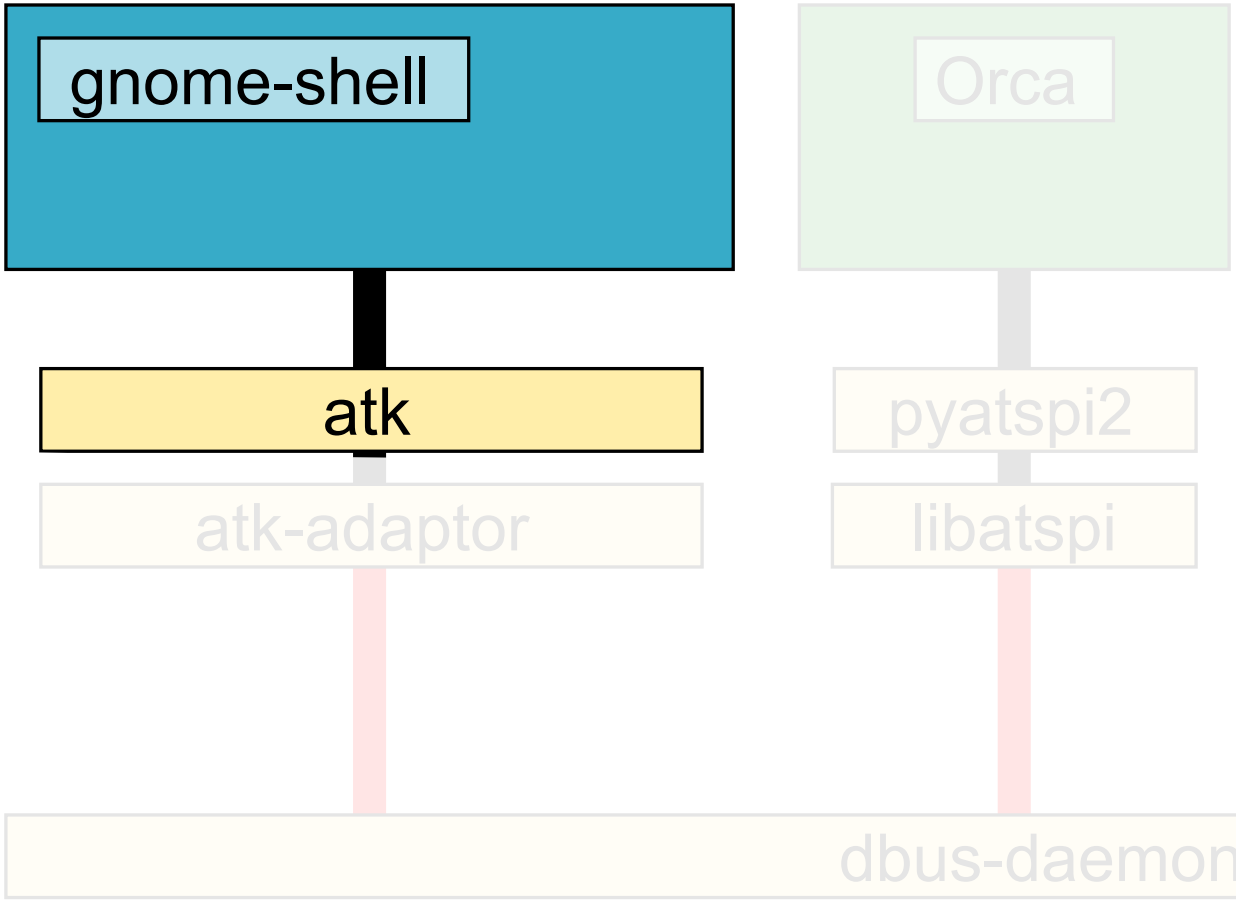
Pagar la deuda técnica en nuestra infraestructura de accesibilidad

Federico Mena Quintero
(pronombre: él)
federico@gnome.org
@federicomena@mstdn.mx

GNOME-LATAM 2023


gnome-shell
GTK3
Firefox
LibreOffice
Orca
Accerciser
Odilia
GTK4
Qt5
WebKit
atk
atk-adaptor
pyatspi2
libatspi
dbus-daemon
at-spi-registryd
toolkits
assistive tech
glue
in-process
context switch!


gnome-shell
Orca
atk
pyatspi2
atk-adaptor
libatspi
dbus-daemon
toolkits
assistive tech
glue
in-process
context switch!



atk:

Sólo interfases.

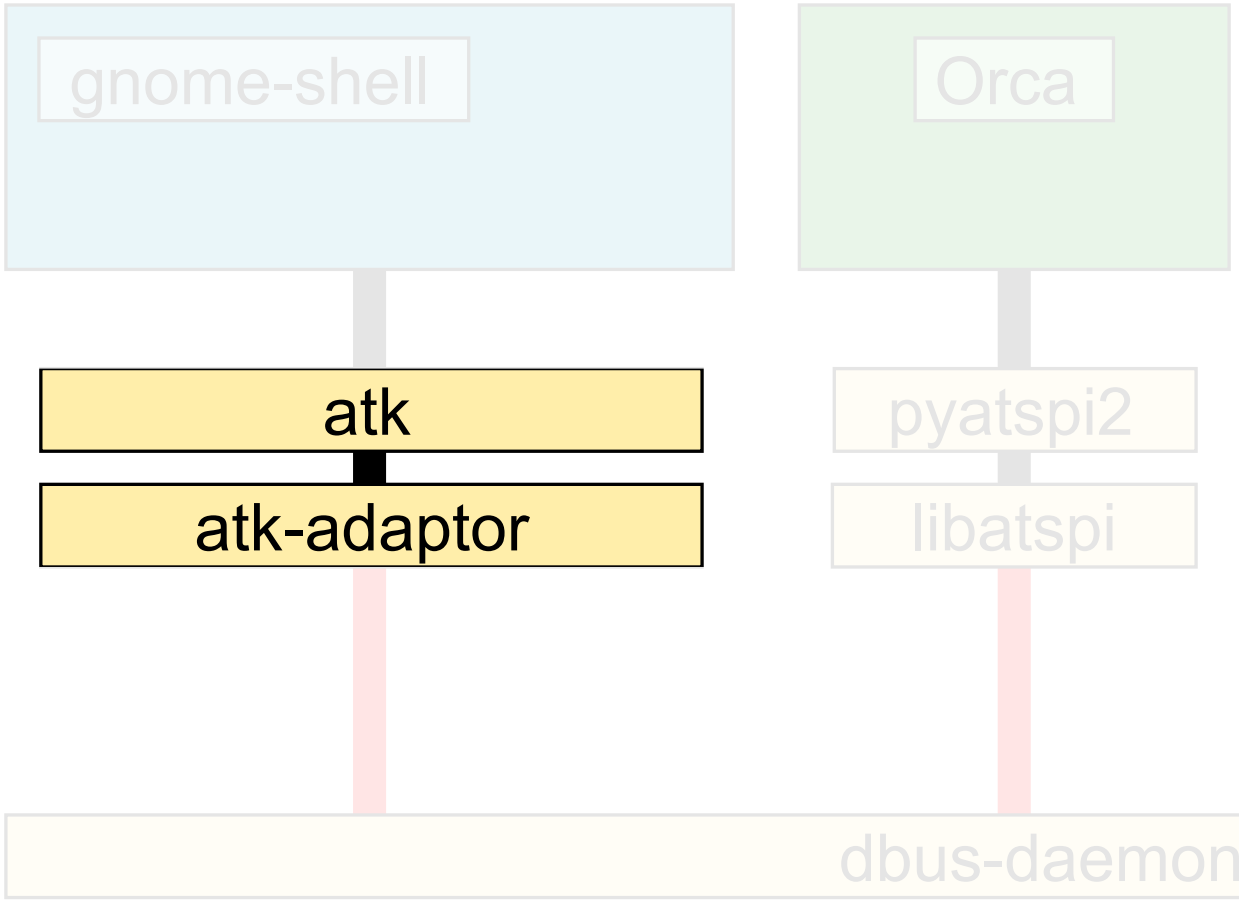
Los toolkits (como st en gnome-shell) implementan las interfases.

“Enumera tus descendientes”

“Dame el nombre accesible”

“Dame el rol accesible”

“Ejecuta una acción”



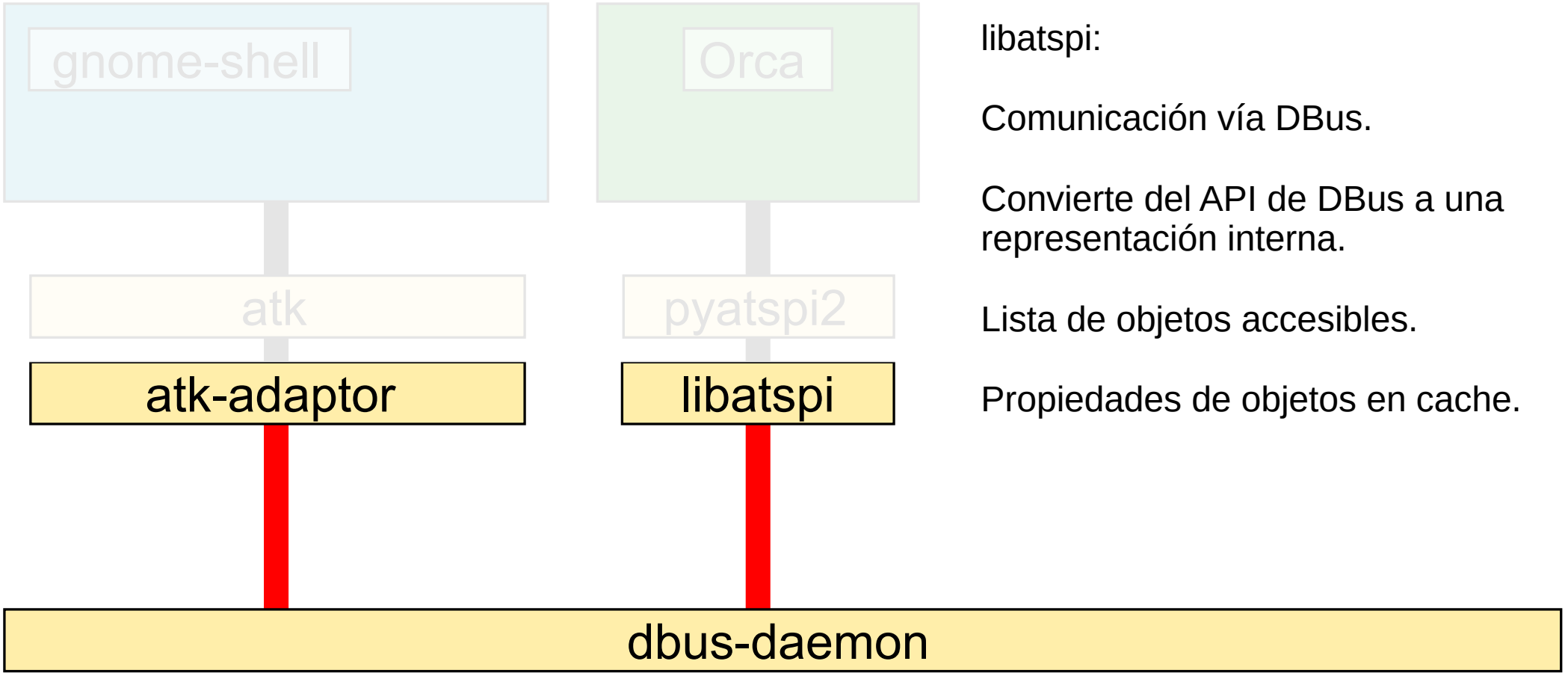
atk-adaptor:

Convierte de la representación de atk a la representación del API que se usa sobre DBus.

Cachear algunas cosas para que no haya que preguntarle otra vez al toolkit.

Implementa las interfases de DBus en términos de atk.

Comunicación vía DBus.



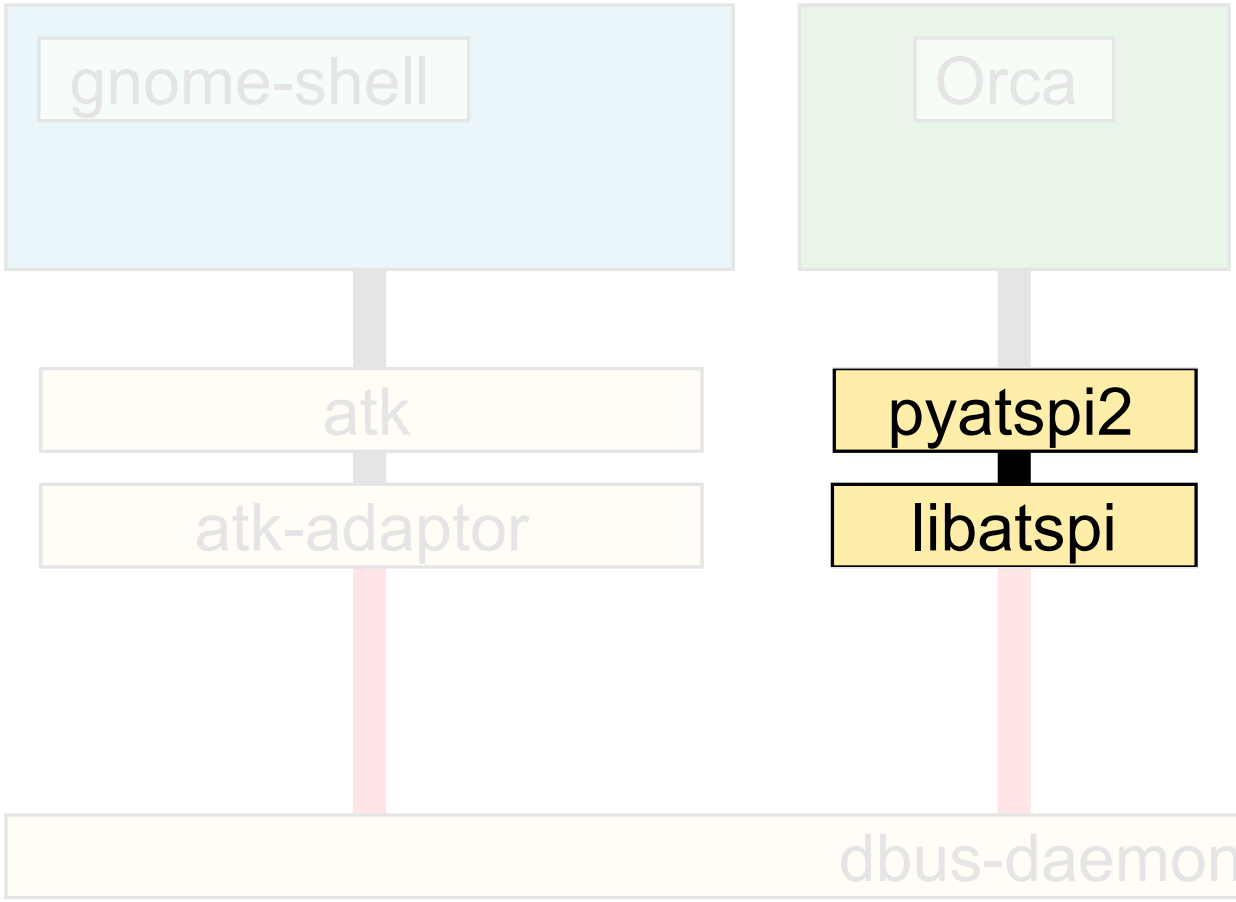
libatspi:

Comunicación vía DBus.

Convierte del API de DBus a una representación interna.

Lista de objetos accesibles.

Propiedades de objetos en cache.

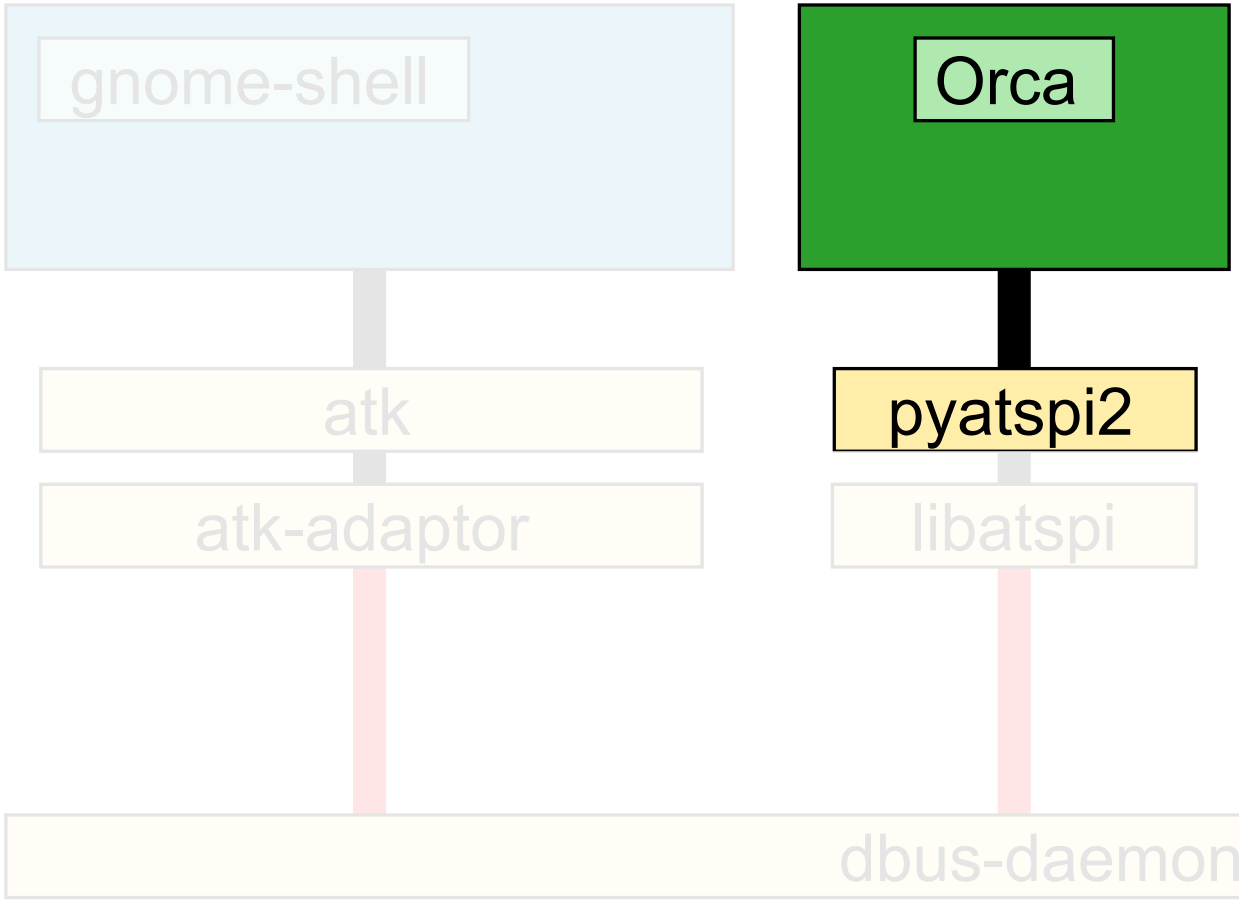


pyatspi2:

“API viejo en Python”

“API nuevo” (GObject Introspection)

Es un binding para Python sobre libatspi.

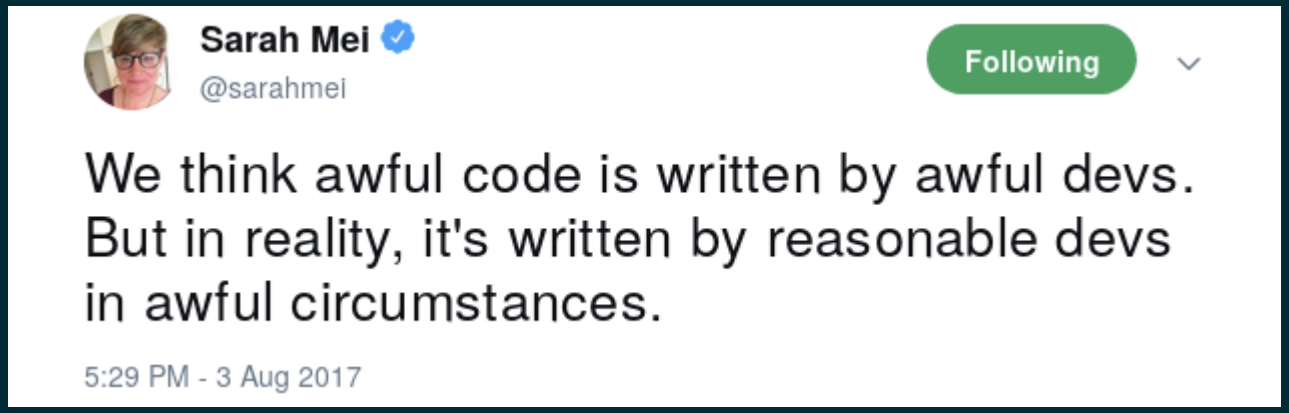


Orca:

Lector de pantalla para ciegos.

120K líneas de Python (!)

Usa el “API viejo” de pyatspi2...
pero no por mucho tiempo.



https://twitter.com/sarahmei/status/893237308316565505

“Creemos que el código horrible fue escrito por personas horribles. Pero en realidad, lo escriben personas razonables en circunstancias horribles.”

¿Cuáles circunstancias horribles?

1997-2018

- cvs.gnome.org - 1997
- bugzilla.gnome.org - 1998
- Tara Hernandez inventa la Integración Contínua (CI) en Mozilla - 1998 (Tinderbox): https://web.archive.org/web/20120414004306/https://build-doctor.com/2012/03/14/the-godmother-of-continuous-integation-an-interview-with-tara-hernandez/
- 2001-2002 - Sun contribuye con la infraestructura de accesibilidad para GNOME 2 ”Archaeology of Accessibility” (Arqueología de la Accesibilidad) (Emmanuele Bassi, GUADEC 2020): https://lwn.net/Articles/826738/ https://www.youtube.com/watch?v=eNh0Xg8abj0
- svn.gnome.org - 2006?
- gtestutils (glib) - 2007
- git.gnome.org - 2008

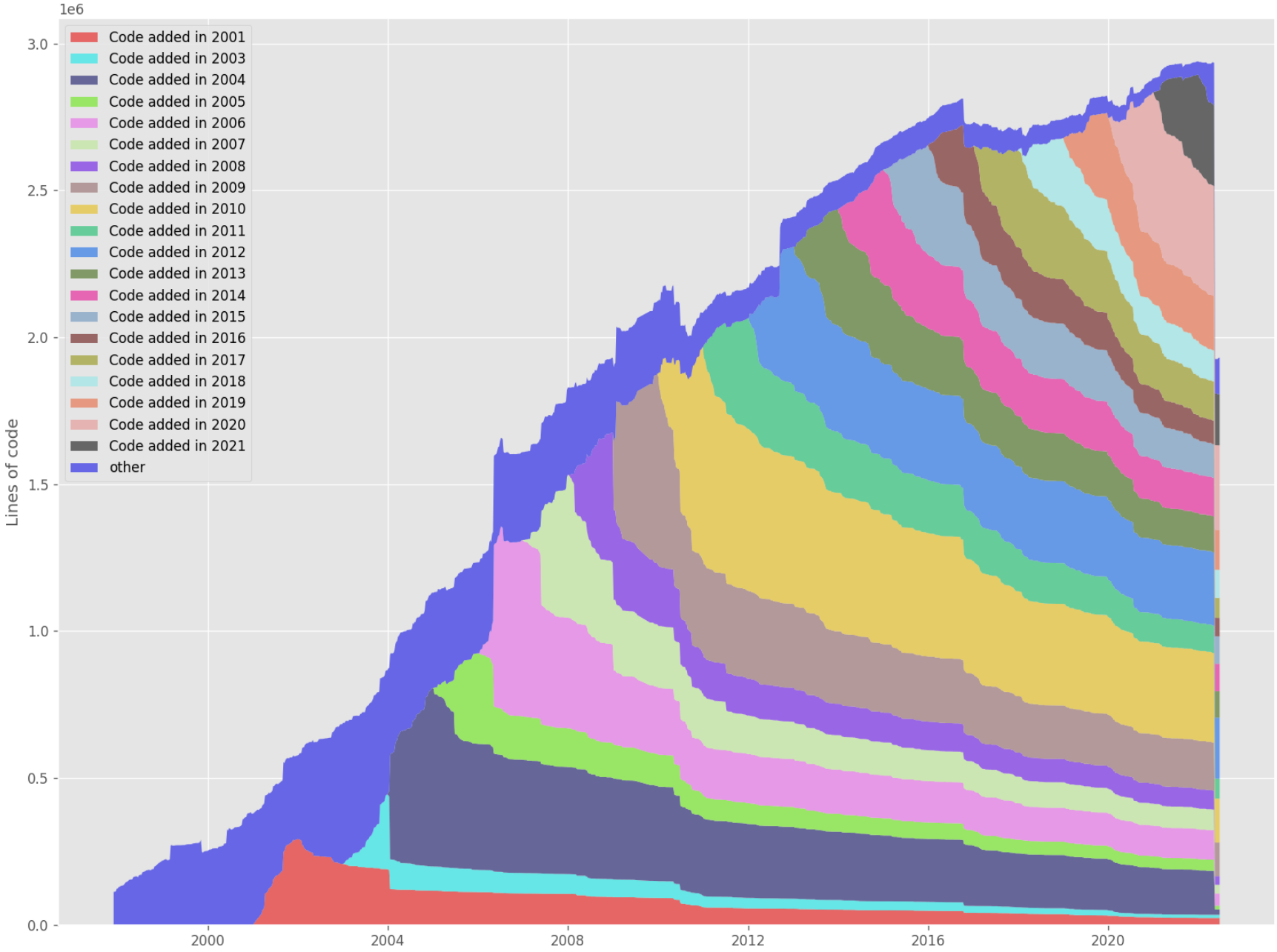
- **Oracle compra a Sun; el equipo de Accesibilidad desaparece (2010-2011)**

- GitHub - 2008; Travis CI - 2011
- contenedores - Jessie Frazelle y Docker - 2013 - 2015?
- contenedores sin ser root - 2016
- gitlab.gnome.org - 2018

El estado típico del código heredado

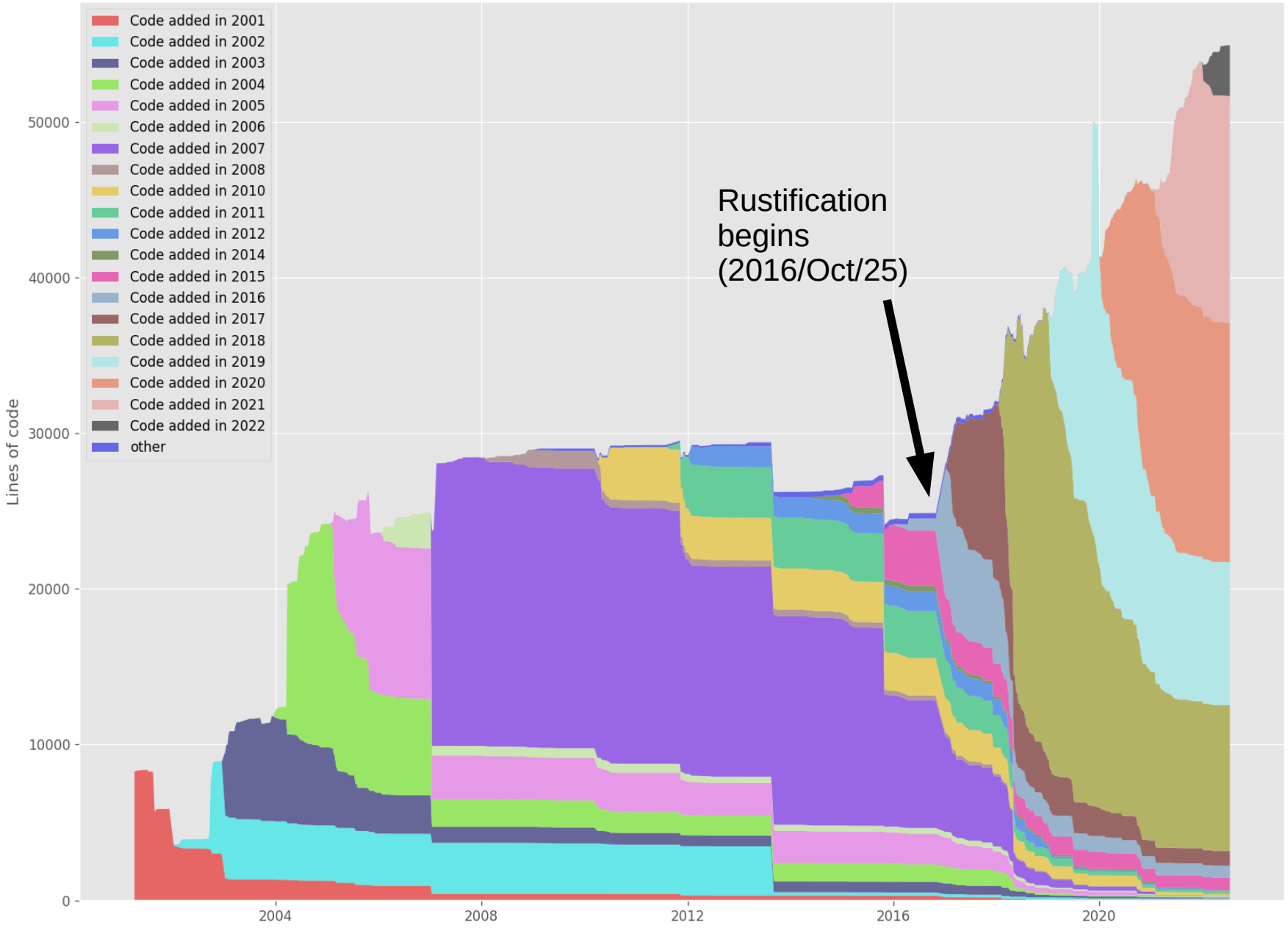
- Librsvg:
  ~~pocas pruebas que no funcionan bien, sin CI, sin entorno reproducible.~~
- Accesibilidad:
  pocas pruebas que no funcionan bien, sin CI, sin entorno reproducible.
- Yelp:
  pocas pruebas que no funcionan bien, sin CI, sin entorno reproducible.
- gnome-session:
- pocas pruebas que no funcionan bien, sin CI, sin entorno reproducible.

Líneas de código en GTK



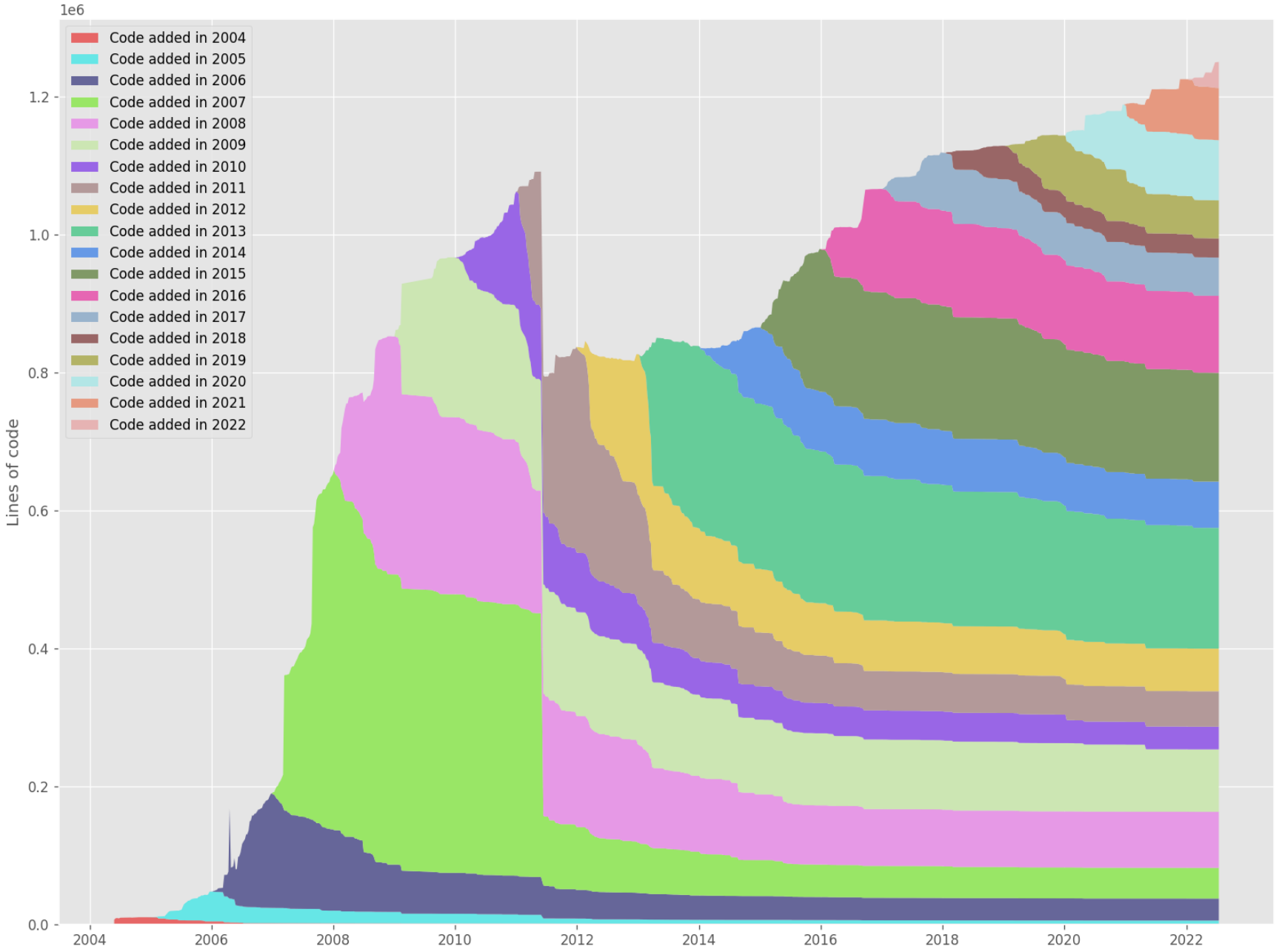
https://github.com/erikbern/git-of-theseus

Líneas de código en librsvg

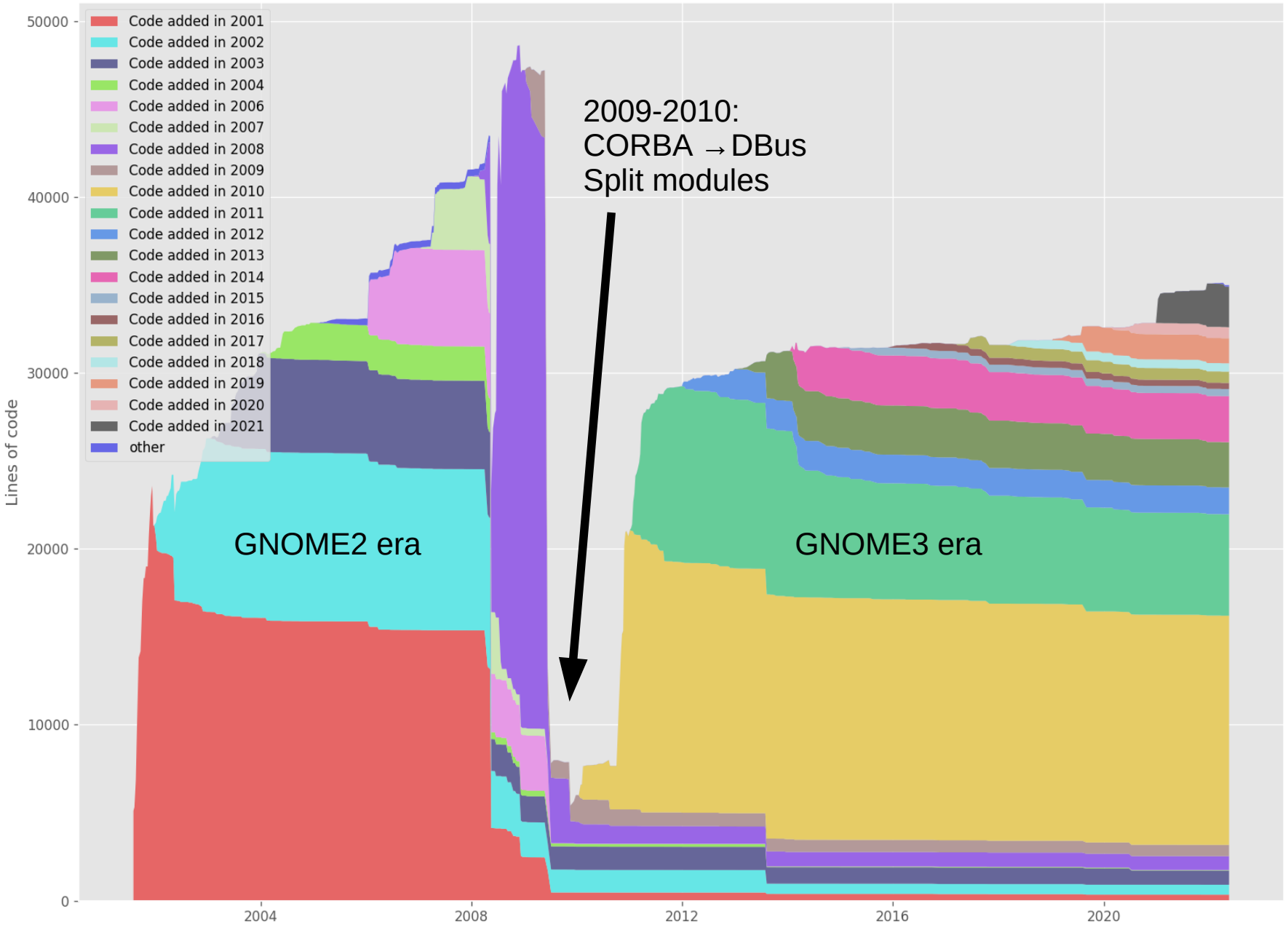


https://github.com/erikbern/git-of-theseus

Líneas de código en Orca



Líneas de código en at-spi2-core

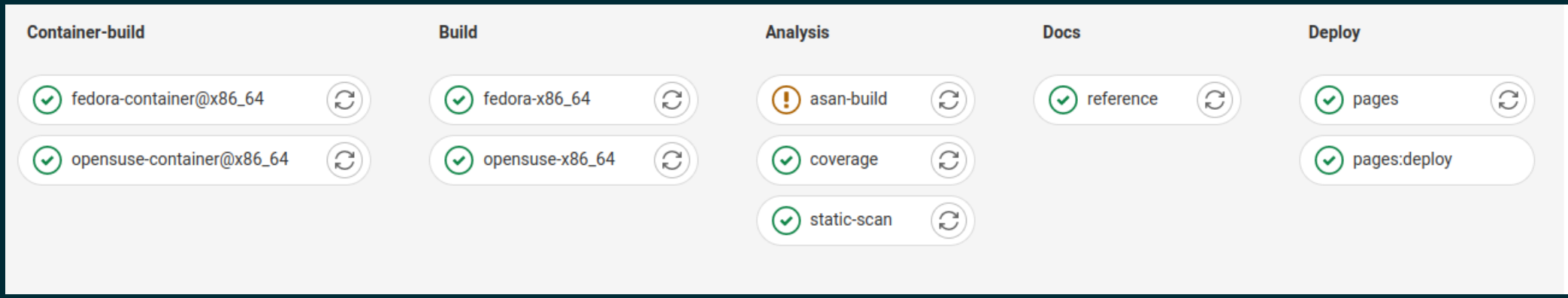


https://github.com/erikbern/git-of-theseus

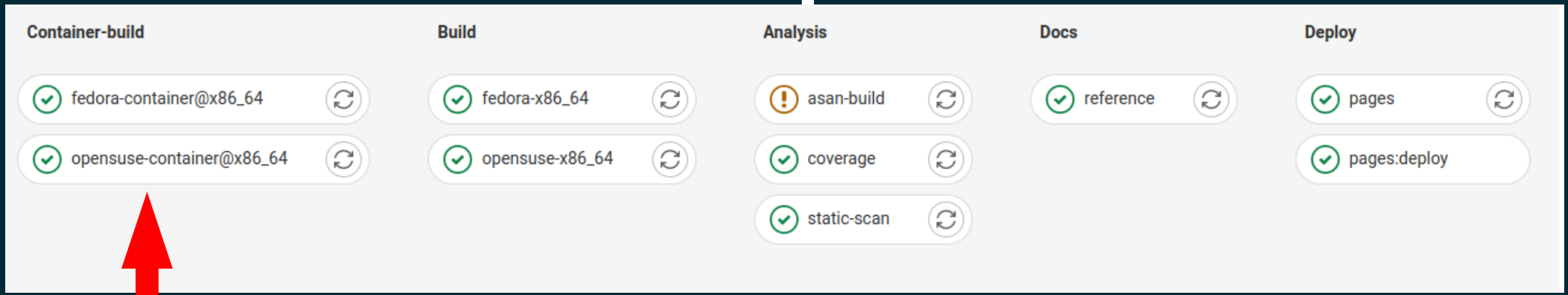
Primero: añadir Integración Contínua

- Copié el .gitlab-ci.yml de libgweather (¡gracias, ebassi!)
- Quería un proyecto con Meson y en C con cosas bonitas: análisis estático, address sanitizer, gi-docgen, reportes de cobertura de pruebas.
- Le puse los Freedesktop CI Templates (copia de librsvg - ¡gracias, alatiera!)

El pipeline de CI de at-spi2-core



Construir un entorno de forma reproducible

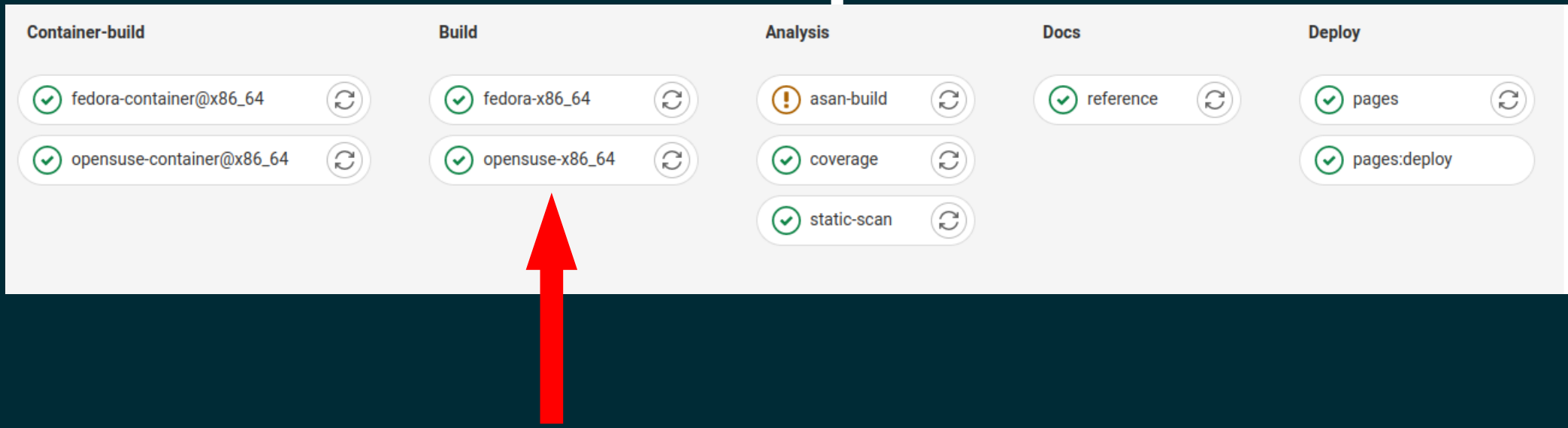


Construye un entorno de forma reproducible (imágenes de contenedores)

Sube la imagen de forma automática al registro de contenedores de gitlab.gnome.org

Freedesktop CI Templates: https://gitlab.freedesktop.org/freedesktop/ci-templates/

Compilar todo y correr las pruebas

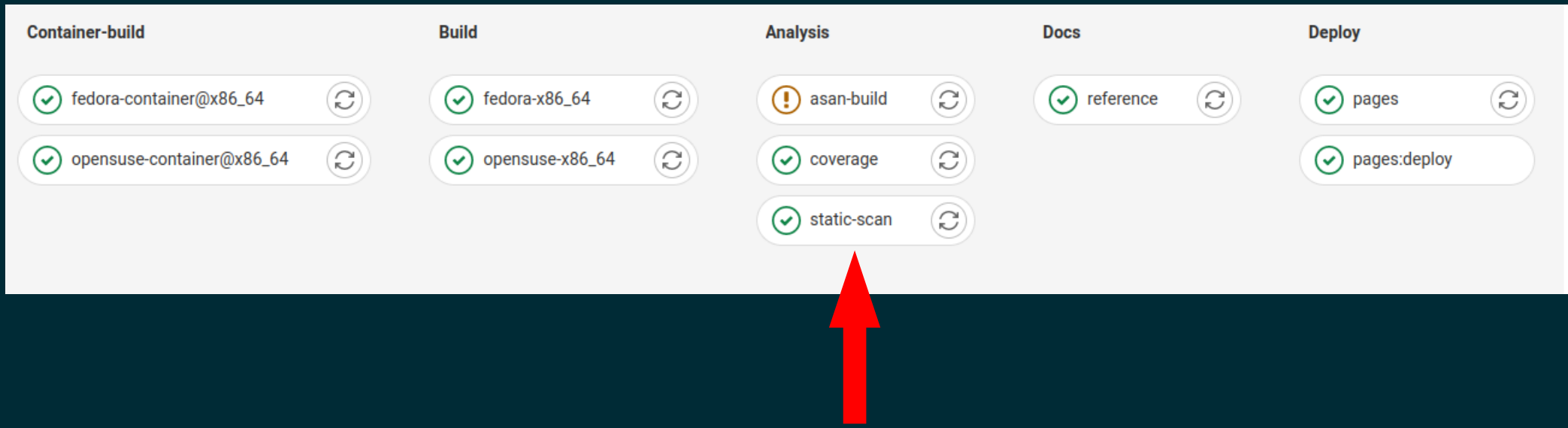


Compila el proyecto

Corre las pruebas (sólo en openSUSE / dbus-daemon)

SE BUSCA AYUDA para Fedora / dbus-broker / ¿Necesitamos una VM?

Los compiladores modernos son maravillosos

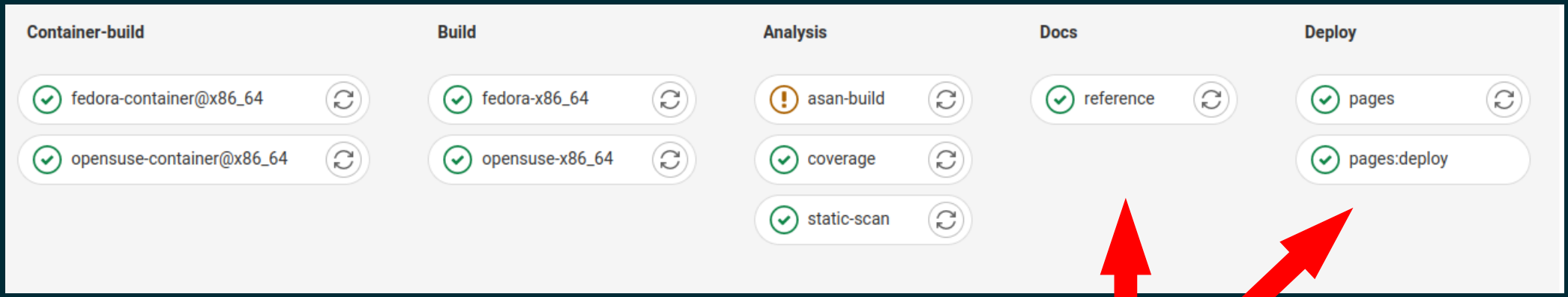


Address sanitizer - para un lenguaje sin manejo seguro de memoria

Análisis estático - porque los compiladores son mucho mejores que en 2005

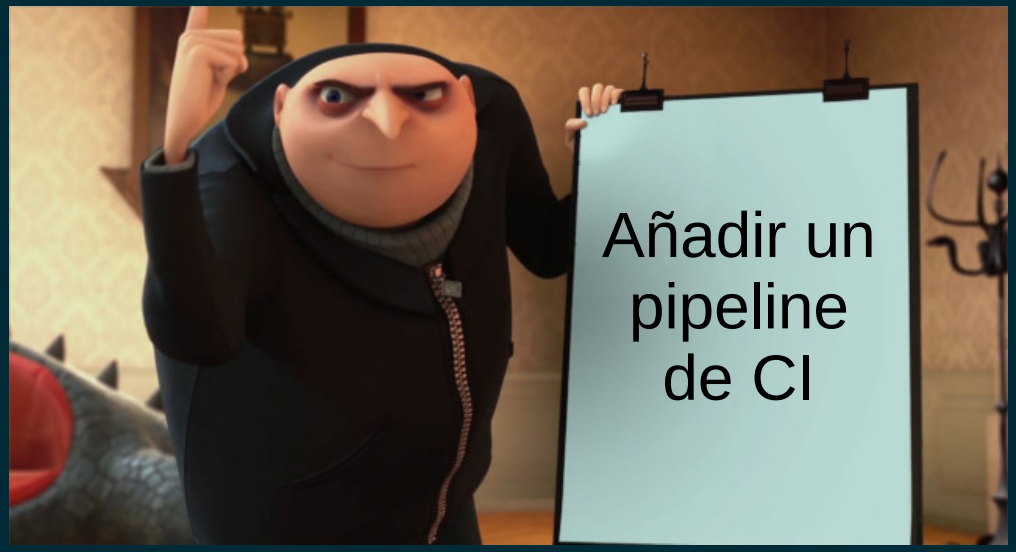
Reportes de cobertura de pruebas - “¿Qué código sí se ejecutó?”

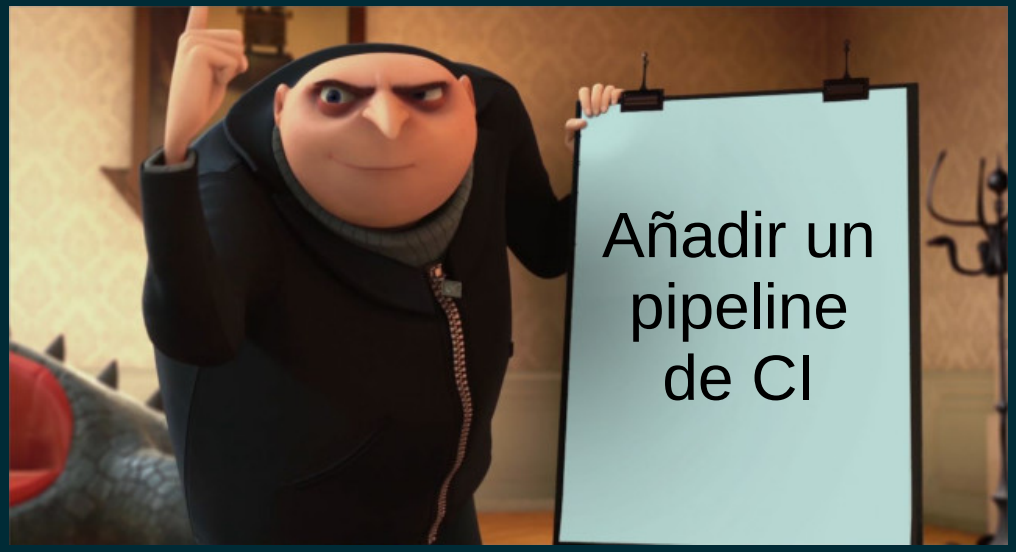
Generar HTML, publicarlo

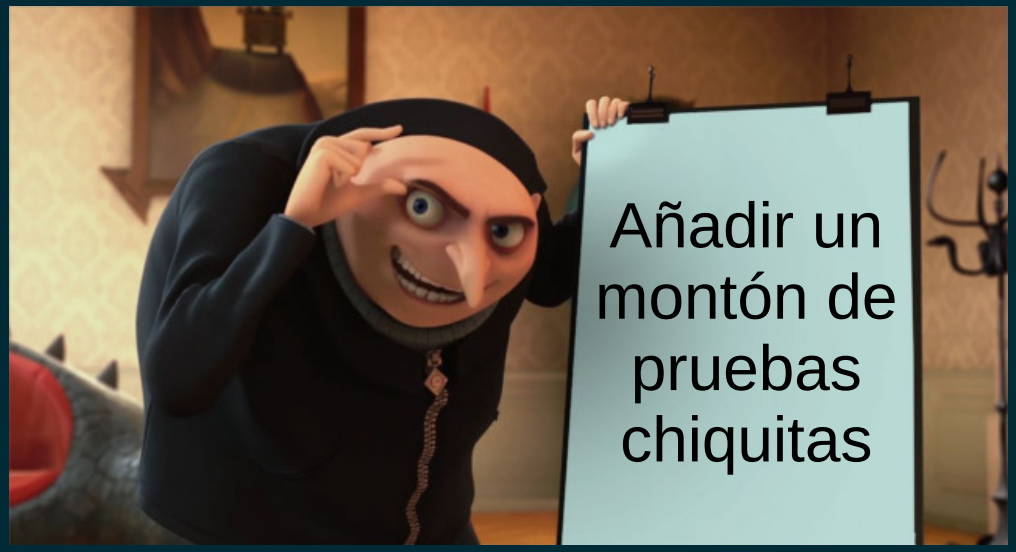


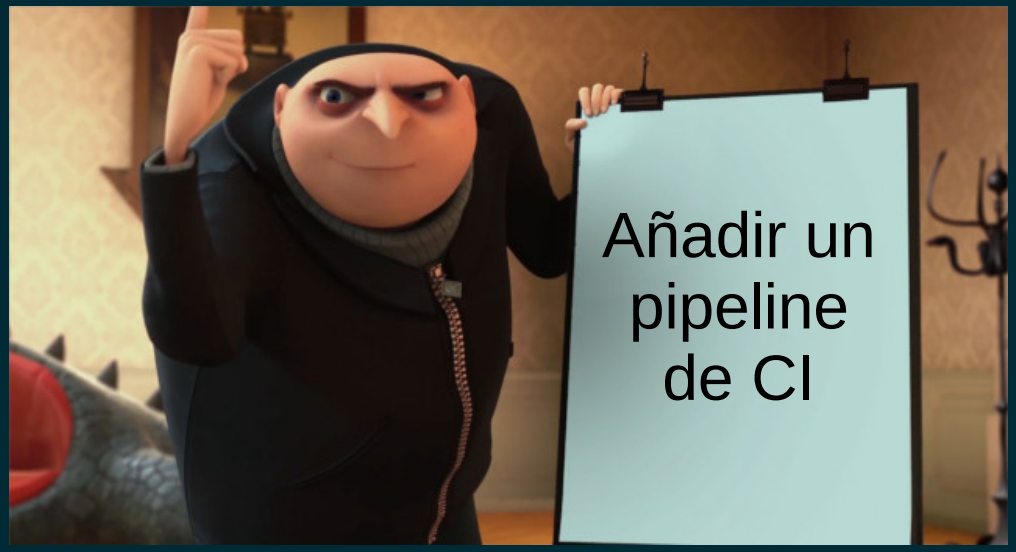
Generar la documentación

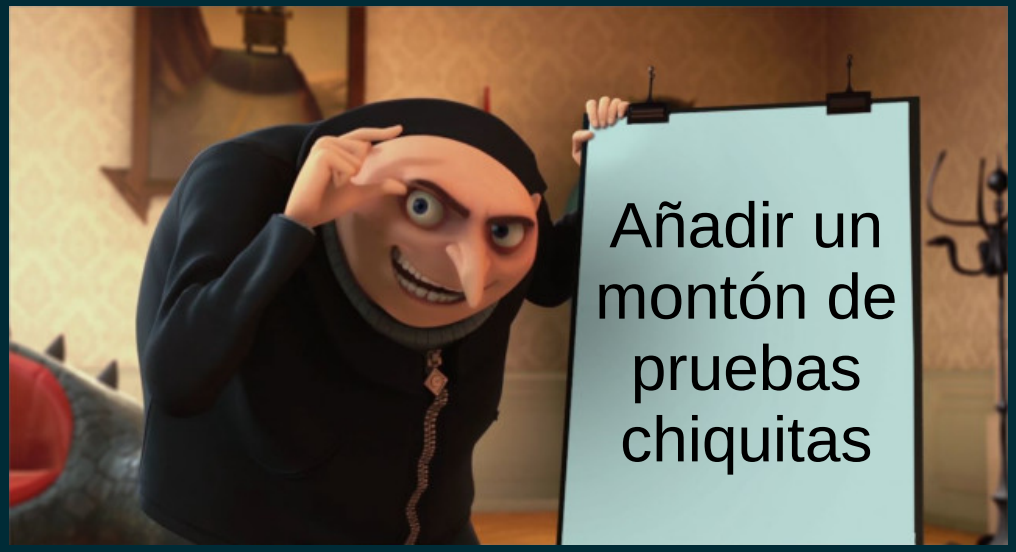
Publicar la documentación y el reporte de cobertura en HTML a GitLab Pages.

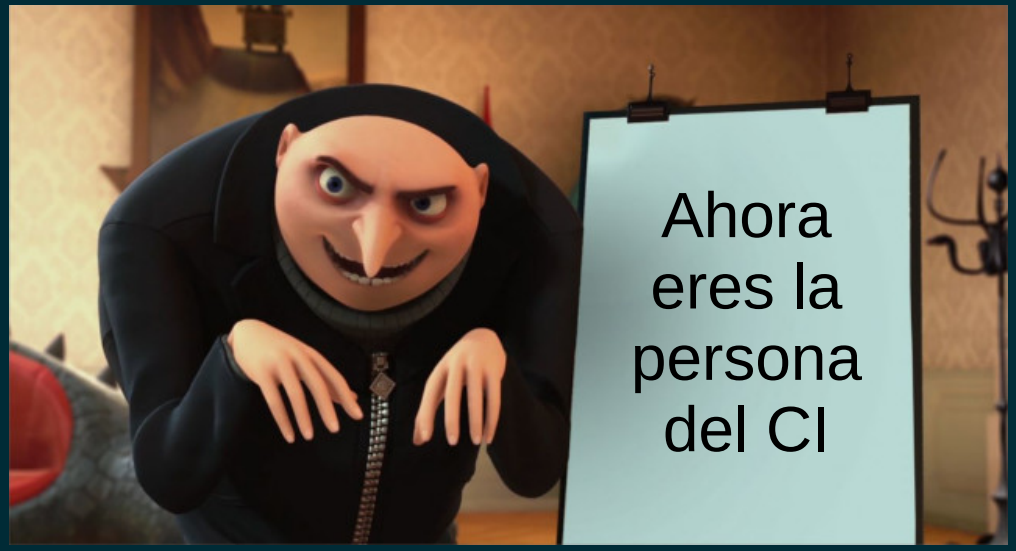

Añadir un
pipeline
de CI

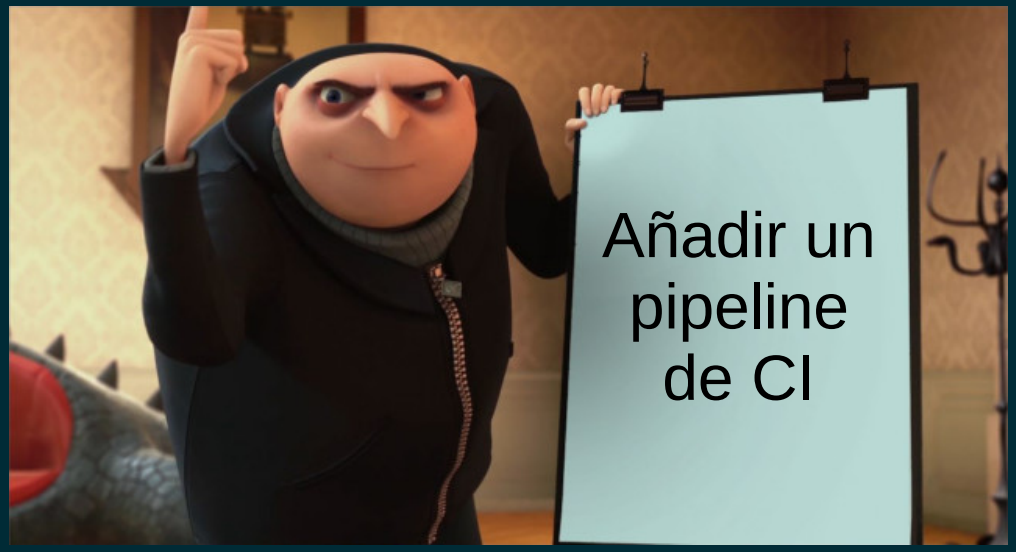

Añadir un pipeline de CI

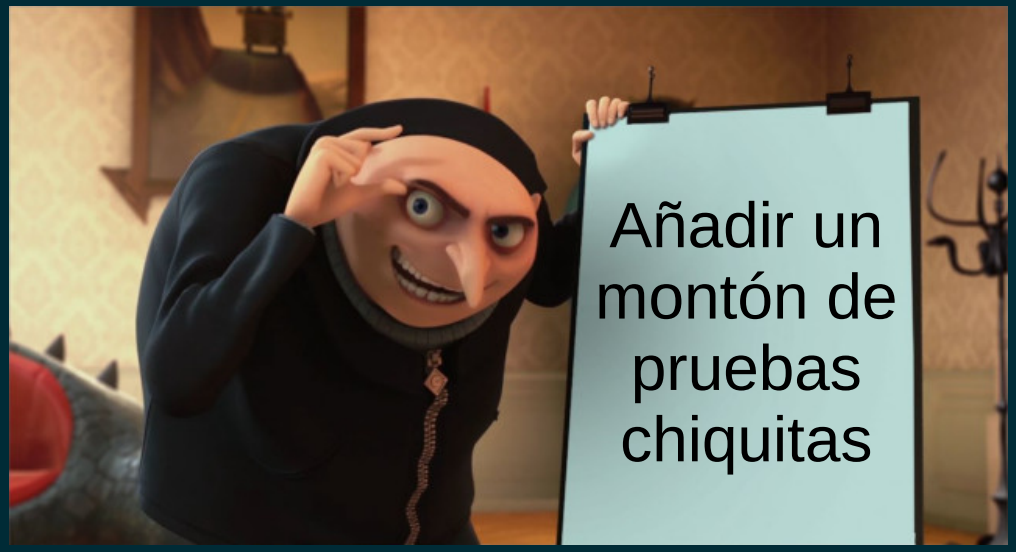

Añadir un montón de pruebas chiquitas

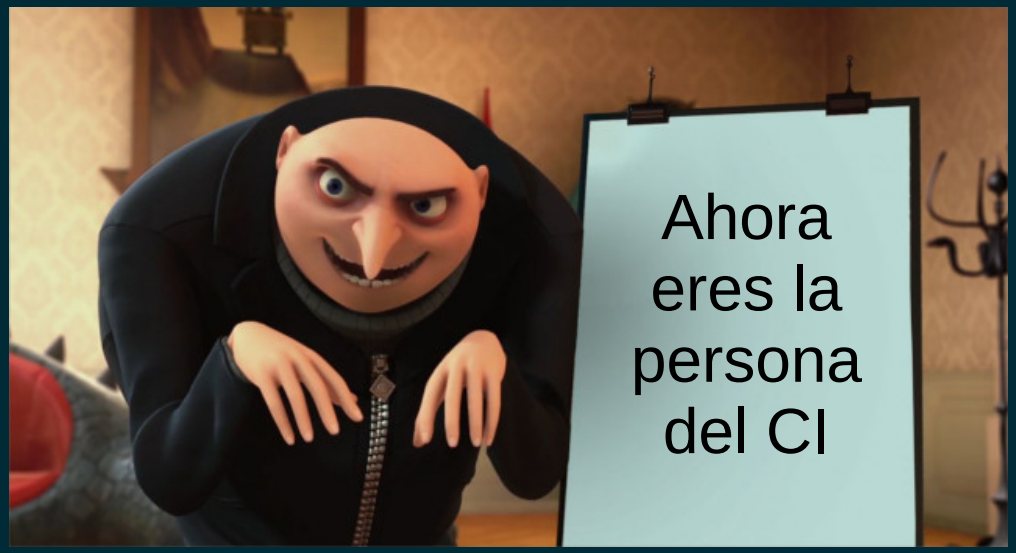

Añadir un pipeline de CI

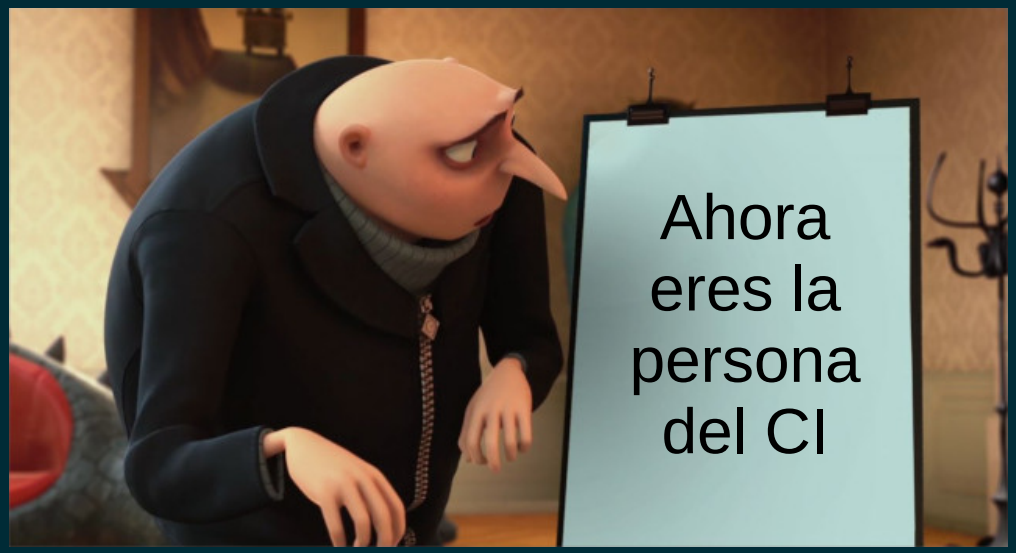

Añadir un montón de pruebas chiquitas


Ahora eres la persona del CI


Añadir un pipeline de CI


Añadir un montón de pruebas chiquitas


Ahora eres la persona del CI


Ahora eres la persona del CI

Cobertura de pruebas en at-spi2-core

LINES	FUNCTIONS	BRANCHES
60.76 %	67.96 %	36.95 %

Directory	Line Coverage		Functions		Branches	
atk	56.38%	1521 / 2698	62.77%	236 / 376	33.63%	957 / 2846
atk-adaptor	50.28%	986 / 1961	56.67%	85 / 150	26.69%	265 / 993
atk-adaptor/adaptors	59.96%	1520 / 2535	76.58%	170 / 222	35.63%	790 / 2217
atk-adaptor/gtk-2.0	0%	0 / 13	0%	0 / 4	0%	0 / 2
atspi	53.66%	2517 / 4691	63.24%	375 / 593	35.21%	780 / 2215
bus	56.23%	194 / 345	60%	15 / 25	34.42%	53 / 154
dbind	66.67%	472 / 708	79.41%	27 / 34	45.59%	124 / 272
droute	57.73%	239 / 414	63.83%	30 / 47	41.18%	56 / 136
registryd	36.51%	831 / 2276	46.23%	92 / 199	21.05%	216 / 1026
tests/at-spi2-atk	97.57%	2089 / 2141	98.99%	196 / 198	53.6%	699 / 1304
tests/at-spi2-atk/dummyatk	85.02%	1283 / 1509	77.11%	192 / 249	52.12%	591 / 1134
tests/atk	89.31%	234 / 262	88.24%	30 / 34	43.48%	80 / 184
tests/atspi	53.49%	46 / 86	71.43%	5 / 7	42.31%	11 / 26

Cobertura de pruebas en librsvg

LINES	FUNCTIONS	BRANCHES
91.83 %	47 %	42.7 %

Directory	Line Coverage		Functions		Branches	
_build/target/debug/build/target-lexicon-057b023953dd6122/out	0%	0 / 21	0%	0 / 6	100%	0 / 0
librsvg-c	97.3%	36 / 37	100%	2 / 2	53.93%	96 / 178
librsvg-c/src	86.17%	1003 / 1164	32.33%	182 / 563	38.24%	785 / 2053
rsvg	93.55%	29 / 31	100%	18 / 18	45.45%	50 / 110
rsvg/src	93.7%	10816 / 11543	42.48%	5162 / 12151	42.75%	12580 / 29427
rsvg/src/filters	89.35%	2089 / 2338	25.75%	380 / 1476	44.17%	1901 / 4304
rsvg/src/surface_utils	94.39%	825 / 874	30.28%	182 / 601	48.99%	700 / 1429
rsvg/src/test_utils	81.86%	167 / 204	64.8%	162 / 250	38.46%	280 / 728
rsvg/src/xml	87.2%	613 / 703	49.13%	169 / 344	47.47%	713 / 1502
rsvg/tests	96.72%	886 / 916	98.43%	1761 / 1789	42.85%	1079 / 2518
rsvg_convert	100%	3 / 3	100%	2 / 2	100%	0 / 0
rsvg_convert/src	93.16%	708 / 760	34.76%	130 / 374	44.46%	758 / 1705
rsvg_convert/tests	99.81%	538 / 539	100%	183 / 183	39.88%	768 / 1926
rsvg_convert/tests/internal_predicates	53.83%	204 / 379	61.29%	57 / 93	18.09%	87 / 481

Cobertura de pruebas en glib

		Hit	Total	Coverage
Current view:	**top level**			
Test:	**unnamed**			
Date:	**2023-05-23 17:30:54**			
	Lines:	**211899**	**252903**	**83.8 %**
	Functions:	**20317**	**22786**	**89.2 %**
	Branches:	**63098**	**99414**	**63.5 %**

Directory	Line Coverage		Functions		Branches	
glib	**83.7 %**	30046 / 35877	**93.9 %**	2784 / 2966	**70.0 %**	14939 / 21343
glib/deprecated	**78.8 %**	523 / 664	**68.7 %**	68 / 99	**66.1 %**	148 / 224
glib/gio	**66.6 %**	39387 / 59119	**77.9 %**	4826 / 6193	**51.9 %**	14283 / 27499
glib/gio/inotify	**94.7 %**	554 / 585	**100.0 %**	58 / 58	**75.8 %**	210 / 277
glib/gio/tests	**87.7 %**	31587 / 36030	**85.8 %**	1996 / 2327	**58.3 %**	3769 / 6460
glib/gio/tests/modules	**60.0 %**	12 / 20	**57.1 %**	8 / 14	**50.0 %**	6 / 12
glib/glib	**88.1 %**	29664 / 33663	**96.9 %**	2785 / 2875	**75.0 %**	14791 / 19719
glib/glib/deprecated	**78.8 %**	524 / 665	**68.7 %**	68 / 99	**67.0 %**	150 / 224
glib/glib/tests	**96.8 %**	32612 / 33693	**97.3 %**	2918 / 2998	**69.3 %**	5408 / 7809
glib/glib/tests/path-test-subdir	**100.0 %**	3 / 3	**100.0 %**	1 / 1	**-**	0 / 0
glib/gmodule	**70.3 %**	218 / 310	**79.2 %**	19 / 24	**56.1 %**	92 / 164
glib/gmodule/tests	**87.3 %**	137 / 157	**100.0 %**	19 / 19	**48.1 %**	25 / 52
glib/gobject	**81.0 %**	7921 / 9776	**90.2 %**	960 / 1064	**64.5 %**	3199 / 4960
glib/gobject/tests	**92.8 %**	7292 / 7860	**91.6 %**	878 / 958	**60.8 %**	1237 / 2034
glib/gobject/tests/performance	**89.5 %**	570 / 637	**96.2 %**	100 / 104	**68.4 %**	108 / 158
glib/gthread	**100.0 %**	8 / 8	**100.0 %**	2 / 2	**100.0 %**	2 / 2
glib/gthread/tests	**100.0 %**	18 / 18	**100.0 %**	3 / 3	**-**	0 / 0
glib/tests	**91.2 %**	30823 / 33815	**94.7 %**	2824 / 2981	**55.8 %**	4731 / 8477
glib/tests/path-test-subdir	**0.0 %**	0 / 3	**0.0 %**	0 / 1	**-**	0 / 0


709 static DBusMessage *
710 2 impl_GetChildren (DBusMessage * message, SpiRegistry *registry)
711 {
712 2 DBusMessage *reply = NULL;
713 DBusMessageIter iter, iter_array;
714 int i;
715
716 2 reply = dbus_message_new_method_return (message);
717
718 2 dbus_message_iter_init_append (reply, &iter);
719 2 dbus_message_iter_open_container(&iter, DBUS_TYPE_ARRAY, "(so)", &iter_array);
720 3 for (i=0; i < registry->apps->len; i++)
721 {
722 1 SpiReference *current = g_ptr_array_index (registry->apps, i);
723 1 append_reference (&iter_array, current->name, current->path);
724 }
725 2 dbus_message_iter_close_container(&iter, &iter_array);
726 2 return reply;
727 }
728
729 static DBusMessage *
730 impl_GetIndexInParent (DBusMessage * message, SpiRegistry *registry)
731 {
732 DBusMessage *reply;
733 dbus_uint32_t rv = 0;
734
735 reply = dbus_message_new_method_return (message);
736 dbus_message_append_args (reply, DBUS_TYPE_INT32, &rv, DBUS_TYPE_INVALID);
737 return reply;
738 }

```python
def test_accessible_iface_properties(registry_root, session_manager):
    values = [
        ('Name',        'main'),
        ('Description', ''),
        ('Parent',      ('', '/org/a11y/atspi/null')),
        ('ChildCount',  0),
    ]

    for prop_name, expected in values:
        assert get_property(registry_root, ACCESSIBLE_IFACE, prop_name) == expected
```

```
1417    if (!strcmp (prop_iface, SPI_DBUS_INTERFACE_ACCESSIBLE))
          {
1417        if      (!strcmp (prop_member, "Name"))
2             impl_get_Name (&iter, registry);
1415        else if (!strcmp (prop_member, "Description"))
1             impl_get_Description (&iter, registry);
1414        else if (!strcmp (prop_member, "Parent"))
1             impl_get_Parent (&iter, registry);
1413        else if (!strcmp (prop_member, "ChildCount"))
1413          impl_get_ChildCount (&iter, registry);
            else
              {
                dbus_message_unref (reply);
                reply = dbus_message_new_error (message, DBUS_ERROR_FAILED, "Property unavailable");
              }
          }
```

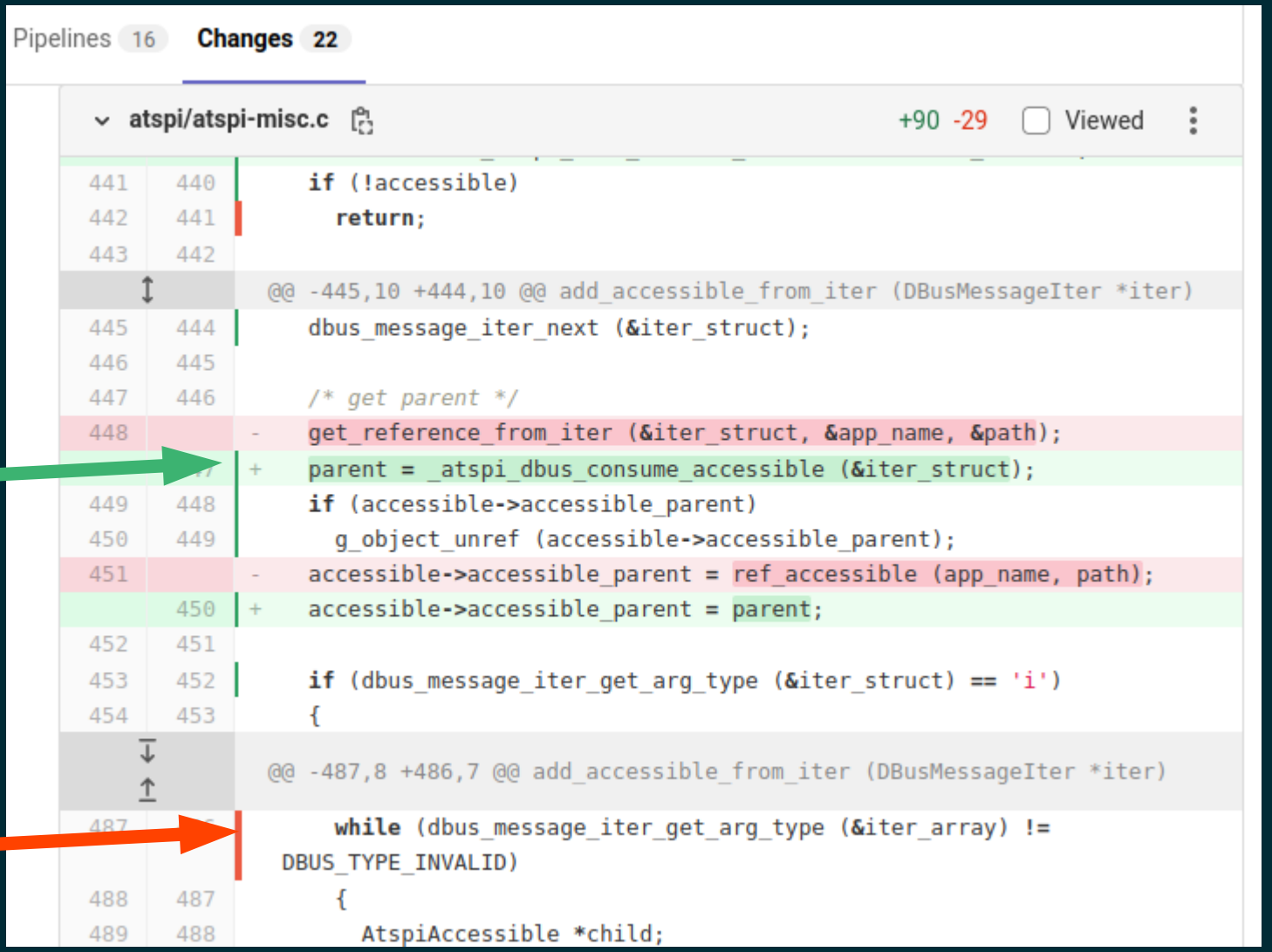
```python
def test_accessible_iface_properties(registry_root, session_manager):
    values = [
        ('Name',        'main'),
        ('Description', ''),
        ('Parent',      ('', '/org/a11y/atspi/null')),
        ('ChildCount',  0),
    ]

    for prop_name, expected in values:
        assert get_property(registry_root, ACCESSIBLE_IFACE, prop_name) == expected

def test_unknown_property_yields_error(registry_root, session_manager):
    with pytest.raises(dbus.exceptions.DBusException):
        get_property(registry_root, ACCESSIBLE_IFACE, 'NonexistentProperty')
```

```
1555                if (!strcmp (prop_iface, SPI_DBUS_INTERFACE_ACCESSIBLE))
                      {
1554                    if      (!strcmp (prop_member, "Name"))
2                         impl_get_Name (&iter, registry);
1552                    else if (!strcmp (prop_member, "Description"))
1                         impl_get_Description (&iter, registry);
1551                    else if (!strcmp (prop_member, "Parent"))
2                         impl_get_Parent (&iter, registry);
1549                    else if (!strcmp (prop_member, "ChildCount"))
1548                      impl_get_ChildCount (&iter, registry);
                        else
                          {
1                           dbus_message_unref (reply);
1                           reply = dbus_message_new_error (message, DBUS_ERROR_FAILED, "Property unavailable");
                          }
                      }
```

Cobertura en los diffs de los merge requests

Sí se ejecutó (verde)

No se ejecutó (rojo)



El hechizo para poner cobertura en los diffs

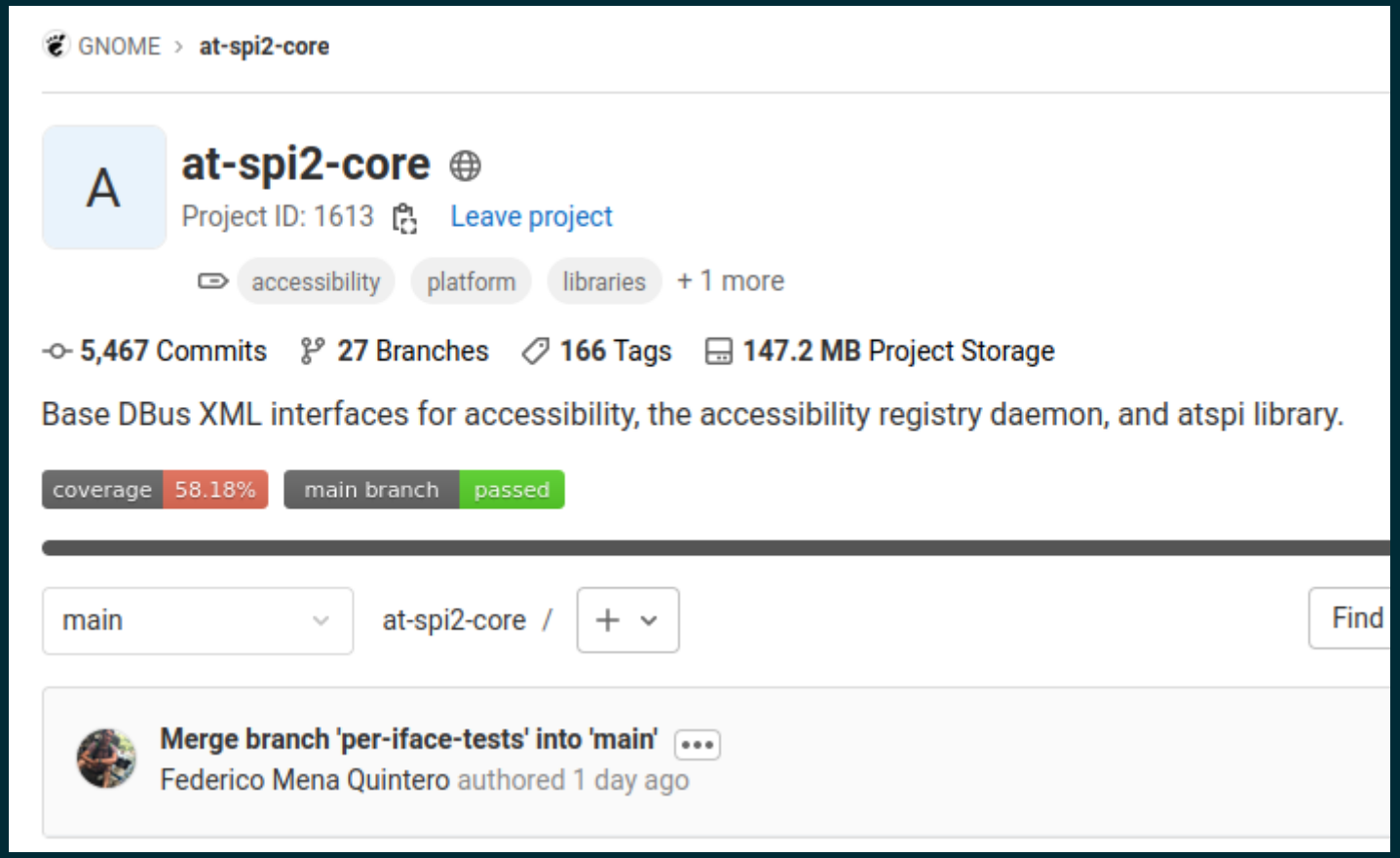
```
artifacts:
  name: "at-spi2-core-${CI_JOB_NAME}-${CI_COMMIT_REF_NAME}"
  expire_in: 2 days
  when: always
  reports:
    coverage_report:
      coverage_format: cobertura
      path: coverage.xml
```

```
grcov _build --source-dir ./ --prefix-dir ../ --output-type cobertura --branch \
        --ignore-not-existing -o coverage.xml
```

```
725     2       dbus_message_iter_close_container(&iter, &iter_array);
726     2       return reply;
727             }
728
729             static DBusMessage *
730             impl_GetIndexInParent (DBusMessage * message, SpiRegistry *registry)
731             {
732               DBusMessage *reply;
733               dbus_uint32_t rv = 0;
```

```
<div class="columns p-0 m-0" role="row">
    <div
        class="column is-1 is-narrow p-0 has-text-centered"
        id="{{ item.0 }}"
        role="cell">
        <a href="#{{ item.0 }}">{{ item.0 }}</a>
    </div>
    <div
        class="column is-1 is-narrow p-0 has-text-centered has-text-{{ highlight_light }}"
        role="cell" aria-label="{{ aria_label }}">
        {{ count }}
    </div>
    <div class="column has-background-{{ highlight_light }} p-0"
        role="cell">
        <pre class="has-background-{{ highlight_light }} py-0 px-2">{{ item.2 }}</pre>
    </div>
</div>
```

alt=”Project badge”



https://gitlab.com/gitlab-org/gitlab/-/issues/364210

Detalle importante sobre los reportes de cobertura

- ¡¡¡Tus procesos deben terminar de manera limpia!!!

- Los runtimes de gcc/clang escriben el reporte de cobertura al terminar el proceso.

- Si terminas con SIGTERM / SEGV / etc, no tendrás datos de cobertura.

Demonios: bus-launcher y registryd

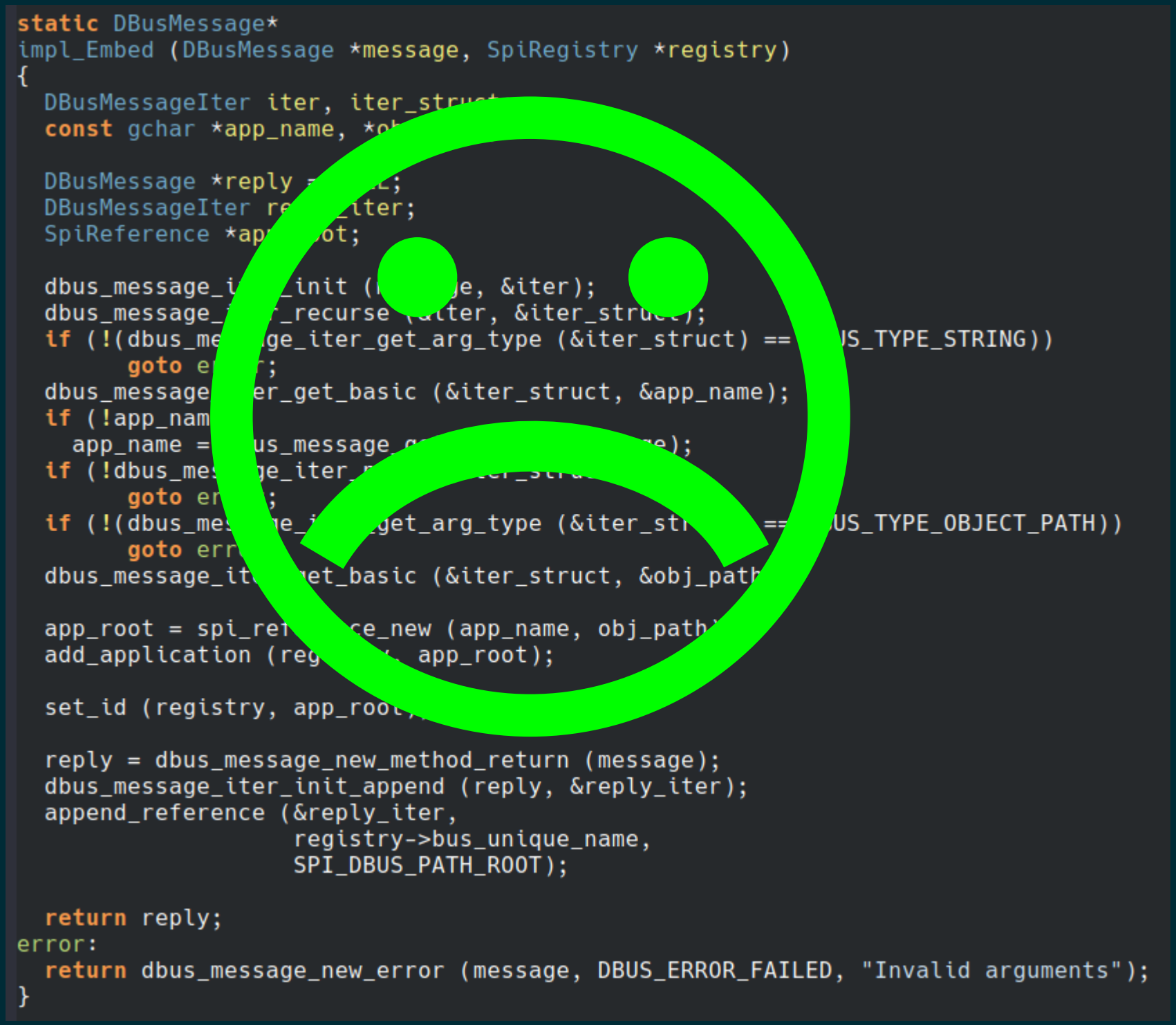
- Su tiempo de vida se controla desde el manejador de sesiones.
- Pero durante las pruebas no hay manejador de sesiones.
- Escribí un mock de gnome-session, 80 líneas.
- ¡python-dbusmock de Martin Pitt es maravilloso!

pytest

- El objeto de pruebas (fixture) es “un demonio del registro de accesibilidad conectado a un manejador de sesiones”.
- Pruebas vía DBus, todo en Python.
- Destruir el objeto de pruebas (fixture teardown) es “decirle al mock del manejador de sesiones que haga Logout”.
- Esto hace que los demonios terminen de forma limpia → obtenemos datos de cobertura.

```python
@pytest.fixture
def session_manager():
    bus = dbus.SessionBus()
    mock_session = bus.get_object('org.gnome.SessionManager', '/org/gnome/SessionManager')

    yield mock_session

    # Tell all session clients to terminate
    mock_session.Logout(0, dbus_interface='org.gnome.SessionManager')
```

```c
static DBusMessage*
impl_Embed (DBusMessage *message, SpiRegistry *registry)
{
  DBusMessageIter iter, iter_struct;
  const gchar *app_name, *obj_path;

  DBusMessage *reply = NULL;
  DBusMessageIter reply_iter;
  SpiReference *app_root;

  dbus_message_iter_init (message, &iter);
  dbus_message_iter_recurse (&iter, &iter_struct);
  if (!(dbus_message_iter_get_arg_type (&iter_struct) == DBUS_TYPE_STRING))
        goto error;
  dbus_message_iter_get_basic (&iter_struct, &app_name);
  if (!app_name)
    app_name = dbus_message_get_sender (message);
  if (!dbus_message_iter_next (&iter_struct))
        goto error;
  if (!(dbus_message_iter_get_arg_type (&iter_struct) == DBUS_TYPE_OBJECT_PATH))
        goto error;
  dbus_message_iter_get_basic (&iter_struct, &obj_path);

  app_root = spi_reference_new (app_name, obj_path);
  add_application (registry, app_root);

  set_id (registry, app_root);

  reply = dbus_message_new_method_return (message);
  dbus_message_iter_init_append (reply, &reply_iter);
  append_reference (&reply_iter,
                    registry->bus_unique_name,
                    SPI_DBUS_PATH_ROOT);

  return reply;
error:
  return dbus_message_new_error (message, DBUS_ERROR_FAILED, "Invalid arguments");
}
```


static DBusMessage*
impl_Embed (DBusMessage *message, SpiRegistry *registry)
{
DBusMessageIter iter, iter_struct
const gchar *app_name, *ob
DBusMessage *reply
DBusMessageIter re iter;
SpiReference *ap ot;
dbus_message_i _init (e, &iter);
dbus_message_ _recurse (&iter, &iter_struct);
if (!(dbus_me ge_iter_get_arg_type (&iter_struct) == US_TYPE_STRING))
goto e ;
dbus_message er_get_basic (&iter_struct, &app_name);
if (!app_nam
app_name = us_message_g e);
if (!dbus_me ge_iter_
goto er ;
if (!(dbus_me ge_i get_arg_type (&iter_st == US_TYPE_OBJECT_PATH))
goto err
dbus_message_it et_basic (&iter_struct, &obj_path
app_root = spi_ref ce_new (app_name, obj_path
add_application (reg app_root);
set_id (registry, app_root
reply = dbus_message_new_method_return (message);
dbus_message_iter_init_append (reply, &reply_iter);
append_reference (&reply_iter,
registry->bus_unique_name,
SPI_DBUS_PATH_ROOT);
return reply;
error:
return dbus_message_new_error (message, DBUS_ERROR_FAILED, "Invalid arguments");
}

obtener parámetros de DBus

```
static DBusMessage*
impl_Embed (DBusMessage *message, SpiRegistry *registry)
{
  DBusMessageIter iter, iter_struct;
  const gchar *app_name, *obj_path;

  DBusMessage *reply = NULL;
  DBusMessageIter reply_iter;
  SpiReference *app_root;

  dbus_message_iter_init (message, &iter);
  dbus_message_iter_recurse (&iter, &iter_struct);
  if (!(dbus_message_iter_get_arg_type (&iter_struct) == DBUS_TYPE_STRING))
      goto error;
  dbus_message_iter_get_basic (&iter_struct, &app_name);
  if (!app_name)
    app_name = dbus_message_get_sender (message);
  if (!dbus_message_iter_next (&iter_struct))
      goto error;
  if (!(dbus_message_iter_get_arg_type (&iter_struct) == DBUS_TYPE_OBJECT_PATH))
      goto error;
  dbus_message_iter_get_basic (&iter_struct, &obj_path);

  app_root = spi_reference_new (app_name, obj_path);
  add_application (registry, app_root);

  set_id (registry, app_root);

  reply = dbus_message_new_method_return (message);
  dbus_message_iter_init_append (reply, &reply_iter);
  append_reference (&reply_iter,
                    registry->bus_unique_name,
                    SPI_DBUS_PATH_ROOT);

  return reply;
error:
  return dbus_message_new_error (message, DBUS_ERROR_FAILED, "Invalid arguments");
}
```

```
static DBusMessage*
impl_Embed (DBusMessage *message, SpiRegistry *registry)
{
  DBusMessageIter iter, iter_struct;
  const gchar *app_name, *obj_path;

  DBusMessage *reply = NULL;
  DBusMessageIter reply_iter;
  SpiReference *app_root;

  dbus_message_iter_init (message, &iter);
  dbus_message_iter_recurse (&iter, &iter_struct);
  if (!(dbus_message_iter_get_arg_type (&iter_struct) == DBUS_TYPE_STRING))
      goto error;
  dbus_message_iter_get_basic (&iter_struct, &app_name);
  if (!app_name)
    app_name = dbus_message_get_sender (message);
  if (!dbus_message_iter_next (&iter_struct))
      goto error;
  if (!(dbus_message_iter_get_arg_type (&iter_struct) == DBUS_TYPE_OBJECT_PATH))
      goto error;
  dbus_message_iter_get_basic (&iter_struct, &obj_path);

  app_root = spi_reference_new (app_name, obj_path);
  add_application (registry, app_root);
  set_id (registry, app_root);

  reply = dbus_message_new_method_return (message);
  dbus_message_iter_init_append (reply, &reply_iter);
  append_reference (&reply_iter,
                    registry->bus_unique_name,
                    SPI_DBUS_PATH_ROOT);

  return reply;
error:
  return dbus_message_new_error (message, DBUS_ERROR_FAILED, "Invalid arguments");
}
```

obtener parámetros de DBus

un poco de maquinaria real

```
static DBusMessage*
impl_Embed (DBusMessage *message, SpiRegistry *registry)
{
  DBusMessageIter iter, iter_struct;
  const gchar *app_name, *obj_path;

  DBusMessage *reply = NULL;
  DBusMessageIter reply_iter;
  SpiReference *app_root;

  dbus_message_iter_init (message, &iter);
  dbus_message_iter_recurse (&iter, &iter_struct);
  if (!(dbus_message_iter_get_arg_type (&iter_struct) == DBUS_TYPE_STRING))
      goto error;
  dbus_message_iter_get_basic (&iter_struct, &app_name);
  if (!app_name)
    app_name = dbus_message_get_sender (message);
  if (!dbus_message_iter_next (&iter_struct))
      goto error;
  if (!(dbus_message_iter_get_arg_type (&iter_struct) == DBUS_TYPE_OBJECT_PATH))
      goto error;
  dbus_message_iter_get_basic (&iter_struct, &obj_path);

  app_root = spi_reference_new (app_name, obj_path);
  add_application (registry, app_root);
  set_id (registry, app_root);

  reply = dbus_message_new_method_return (message);
  dbus_message_iter_init_append (reply, &reply_iter);
  append_ref (&reply_iter,
              registry->bus_unique_name,
              SPI_DBUS_PATH_ROOT);
  return reply;
error:
  return dbus_message_new_error (message, DBUS_ERROR_FAILED, "Invalid arguments");
}
```

obtener parámetros de DBus

un poco de maquinaria real

construir el valor de retorno para DBus

extraer la maquinaria real

```
static SpiReference *
socket_embed (SpiRegistry *registry, SpiReference *app_root)
{
  add_application (registry, app_root);
  set_id (registry, app_root);
  return spi_reference_new (registry->bus_unique_name, SPI_DBUS_PATH_ROOT);
}
```

```
static DBusMessage*
impl_Embed (DBusMessage *message, SpiRegistry *registry)
{
  SpiReference *app_root = NULL;
  SpiReference *result;
  DBusMessage *reply = NULL;
  DBusMessageIter reply_iter;

  if (socket_embed_demarshal (message, &app_root) != DEMARSHAL_STATUS_SUCCESS)
    {
      return dbus_message_new_error (message, DBUS_ERROR_FAILED, "Invalid arguments");
    }

  result = socket_embed (registry, app_root); /* takes ownership of the app_root */

  reply = dbus_message_new_method_return (message);
  dbus_message_iter_init_append (reply, &reply_iter);
  append_reference (&reply_iter, result->name, result->path);
  spi_reference_free (result);

  return reply;
}
```

Cosas que hemos aprendido a lo largo de los años

- Entornos reproducibles para compilación y pruebas ~~“funciona en mi máquina”~~
- Reportes de cobertura de pruebas.
- Compilación automatizada.
- Integración contínua.
- Dejemos que los robots hagan el trabajo aburrido.
- “The Not Rocket Science Rule of Software Engineering” https://graydon2.dreamwidth.org/1597.html
- Usamos marge-bot para eso. 

Coordinar el XML con el código

```
<interface name="org.a11y.atspi.Cache">
  <method name="GetItems">
    <arg direction="out"
         name="nodes"
         type="a((so)(so)(so)iiassusau)"/>
  </method>
</interface>

struct Item {
  object:      ObjectReference,  // (so)
  app:         ObjectReference,  // (so)
  parent:      ObjectReference,  // (so)
  index:       i32,              // i
  child_count: i32,              // i
  interfaces:  Vec<String>,      // as
  name:        String,           // s
  role:        u32,              // u
  description: String,           // s
  states:      StateSet,         // au
}
```

Limpiar los módulos en Python

- Orca, pyatspi2, dogtail, accerciser
- “desaparecer” pyatspi2
- Añadir Integración Contínua (CI)
- Formateo y lints: flake8 o ruff
- Chequeo de tipos: mypy, typeguard
- etc. Todo lo de Hypermodern Python

¡Gracias por venir!