Reinventing the Enlightenment Object System

stosb.com/talks

Tom Hacohen
Samsung Electronics Open Source Group
tom.hacohen@samsung.com
@TomHacohen

FOSDEM 2015
Main Goals

Unify Code
Main Goals

Unify Code

- Many different object systems → one
Main Goals

Unify Code

- Many different object systems → one
- Many different event/callback implementations → one
Main Goals

Unify Code

- Many different object systems → one
- Many different event/callback implementations → one
- Make objects compatible
Main Goals

Reducing our API

```
evas_object_image_file_set (obj , " blah . png ", " key ");
edje_object_file_set (obj , " blah . edj ", " group ");
evas_object_del ( obj );
ecore_timer_del ( obj );
ecore_animator_del ( obj );
```
Main Goals

Reducing our API

We have:

evas_object_image_file_set(obj, "blah.png", "key");
edje_object_file_set(obj, "blah.edj", "group");

evas_object_del(obj);
ecore_timer_del(obj);
ecore_animator_del(obj);
Main Goals

Bindings Generation

Be able to automatically generate for most popular languages

Correctly handle ref counting, buffer ownership and etc.
Main Goals

Bindings Generation

- Be able to automatically generate for most popular languages
Main Goals

Bindings Generation

- Be able to automatically generate for most popular languages
- Correctly handle ref counting, buffer ownership and etc.
Main Goals

Not Hurt Performance
Main Goals

Not Hurt Performance

- Not easily measurable – many changes in EFL
Other Object Systems

Other Languages

- C++ – our developers hate it
- Objective C – quite ugly and not really common in OSS world

We considered using just the runtime
Other Object Systems

Other Languages

- C++ – our developers hate it
Other Object Systems

Other Languages

- C++ – our developers hate it
- Objective C – quite ugly and not really common in OSS world
Other Object Systems

Other Languages

- C++ – our developers hate it
- Objective C – quite ugly and not really common in OSS world
  - We considered using just the runtime
Other Object Systems

GObject

Good:
▶ Fast
▶ Has a “C feel”

Bad:
▶ Doesn’t offer a stable ABI
▶ Funny, full of casting syntax
▶ “G tech” dependencies
▶ Didn’t exactly fit our needs
Other Object Systems

GObject

Good:
- Fast

Bad:
- Doesn't offer a stable ABI
- Funny, full of casting syntax
- "G tech" dependencies
- Didn't exactly fit our needs
Other Object Systems

**GObject**

**Good:**
- Fast
- Has a “C feel”
Other Object Systems

GObject

Good:
- Fast
- Has a “C feel”
Other Object Systems

GObject

Good:

▶ Fast
▶ Has a “C feel”

Bad:

▶ Doesn’t offer a stable ABI
Other Object Systems

GObject

Good:

- Fast
- Has a “C feel”

Bad:

- Doesn’t offer a stable ABI
- Funny, full of casting syntax
Other Object Systems

GObject

Good:
- Fast
- Has a “C feel”

Bad:
- Doesn’t offer a stable ABI
- Funny, full of casting syntax
- “G tech” dependencies
Other Object Systems

GObject

Good:

▶ Fast
▶ Has a “C feel”

Bad:

▶ Doesn’t offer a stable ABI
▶ Funny, full of casting syntax
▶ “G tech” dependencies
▶ Didn’t exactly fit our needs
Other Object Systems

systemd-objectd

Good:
▶ Exposes a dbus API
▶ Clean isolated daemon

Bad:
▶ Linux only
Other Object Systems

systemd-objectd

Good:

- Exposes a dbus API

Bad:

- Linux only
systemd-objectd

Good:

▶ Exposes a dbus API
▶ Clean isolated daemon
Other Object Systems

systemd-objectd

**Good:**
- Exposes a dbus API
- Clean isolated daemon

**Bad:**
- Linux only
Other Object Systems

systemd-objectd

Good:
- Exposes a dbus API
- Clean isolated daemon

Bad:
- Linux only
What is Eo?

Basics

It's Enlightenment's (fairly) new object system
Supports classes, abstract classes, mixins and interfaces
Completely written in C (no external preprocessor)
API/ABI stable
Portable
What is Eo?

Basics

- It’s Enlightenment’s (fairly) new object system
What is Eo?

Basics

- It’s Enlightenment’s (fairly) new object system
- Supports classes, abstract classes, mixins and interfaces
What is Eo?

Basics

- It’s Enlightenment’s (fairly) new object system
- Supports classes, abstract classes, mixins and interfaces
- Completely written in C (no external preprocessor)
What is Eo?

Basics

- It’s Enlightenment’s (fairly) new object system
- Supports classes, abstract classes, mixins and interfaces
- Completely written in C (no external preprocessor)
- API/ABI stable
What is Eo?

Basics

- It’s Enlightenment’s (fairly) new object system
- Supports classes, abstract classes, mixins and interfaces
- Completely written in C (no external preprocessor)
- API/ABI stable
- Portable
What is Eo?

Using Eo

```c
eo_do(obj, efl_file_set("file.eet", "key"));
if (eo_do(obj, elm_widget_enabled_get()))
    eo_do(obj, visible = elm_widget_visibility_get(),
          elm_widget_visibility_set(!visible));
    eo_do(obj, elm_widget_visibility_set(!elm_widget_visibility_get()));
static void size_multiply ( double f)
{
    int w, h;
evas_object_geometry_get (NULL , NULL , &w, &h);
evas_object_geometry_set (NULL , NULL , w * f, h * f);
}
eo_do (obj , size_multiply (3.5) );
```
What is Eo?

Using Eo

- `eo_do(obj, efl_file_set("file.eet", "key"));`

  ```c
  static void size_multiply ( double f)
  {
  int w, h;
  evas_object_geometry_get (NULL , NULL , &w, &h);
  evas_object_geometry_set (NULL , NULL , w * f, h * f);
  }
  eo_do (obj , size_multiply (3.5) );
  ```
What is Eo?

Using Eo

- eo_do(obj, efl_file_set("file.eet", "key"));
- if (eo_do(obj, elm_widget_enabled_get()))

static void size_multiply ( double f)
{    int w, h;
    evas_object_geometry_get (NULL , NULL , &w, &h);
    evas_object_geometry_set (NULL , NULL , w * f, h * f);
}

eo_do (obj , size_multiply (3.5) );
What is Eo?

Using Eo

- `eo_do(obj, efl_file_set("file.eet", "key"));`
- `if (eo_do(obj, elm_widget_enabled_get()))`
- `eo_do(obj, visible = elm_widget_visibility_get(), !visible = elm_widget_visibility_set(!visible));`
What is Eo?

Using Eo

- eo_do(obj, efl_file_set("file.eet", "key"));
- if (eo_do(obj, elm_widget_enabled_get()))
  eo_do(obj, visible = elm_widget_visibility_get(), !visible);
- eo_do(obj, elm_widget_visibility_set(!elm_widget_visibility_get()));
- static void size_multiply ( double f)
  int w, h;
  evas_object_geometry_get (NULL , NULL , &w, &h);
  evas_object_geometry_set (NULL , NULL , w * f, h * f);
  eo_do (obj , size_multiply (3.5) );
What is Eo?

Using Eo

```c
// eo_do(obj, efl_file_set("file.eet", "key"));
if (eo_do(obj, elm_widget_enabled_get()))
    eo_do(obj, visible = elm_widget_visibility_get(),
           elm_widget_visibility_set(!visible));
else
    eo_do(obj, elm_widget_visibility_set(!elm_widget_visibility_get()));

static void size_multiply(double f)
{
    int w, h;
    evas_object_geometry_get(NULL, NULL, &w, &h);
    evas_object_geometry_set(NULL, NULL, w * f, h * f);
}

eo_do(obj, size_multiply(3.5));
```
What is Eo? | Internals

eo_do() – How It’s Done (simplified)
What is Eo? | Internals

eo_do() – How It’s Done (simplified)

```c
#define eo_do(eoid, clsid, ...) \
({ \
    const Eo * _eoid_ EO_DO_CLEANUP = eoid; \ 
    _eo_do_start(_eoid_, clsid); \ 
    __VA_ARGS__; \ 
})
```
Defining New Functions (simplified)
Defining New Functions (simplified)

EOAPI EO_FUNC_BODY(eo_parent_get, Eo *, NULL);
What is Eo? | Internals

Defining New Functions (simplified)

EOAPI EO_FUNC_BODY(eo_parent_get, Eo *, NULL);

#define EO_FUNC_BODY(Name, Ret, DefRet) 
Ret Name(void) 
{
    static Eo_Op op = EO_NOOP;
    if (op == EO_NOOP)
        op = _eo_api_op_id_get((void*) Name);
    if (!_eo_call_resolve(#Name, op, &call))
        return DefRet;
    _Eo_##Name##_func_func_ =
        (_Eo_##Name##_func) call.func;
    return _func_(call.obj, call.data);
}
Defining New Classes (simplified)

Populating a struct with some metadata
What is Eo?  |  Internals

Defining New Classes (simplified)

Populating a struct with some metadata

```c
static Eo_Op_Description _edge_object_op_desc[] = {
    EO_OP_FUNC(edge_obj_update_hints_set, _edje_object_update_hints_set),
    EO_OP_FUNC_OVERRIDE(eo_constructor, _edje_object_eo_base_constructor),
    EO_OP_CLASS_FUNC(eo_event_global_thaw, _eo_base_event_global_thaw),
    EO_OP_CLASS_OVERRIDE_FUNC(eo_event_global_thaw, _edje_object_eo_base_event_global_thaw)
};
```
What is Eo? | Internals

Event Identifiers

EOAPI const Eo_Event_Description ← EO_BASE_EVENT_CALLBACK_ADD ← EO_EVENT_DESCRIPTION (" callback , add ");
What is Eo? | Internals

Event Identifiers

```c
EOAPI const Eo_Event_Description ←
_EO_BASE_EVENT_CALLBACK_ADD = ←
EO_EVENT_DESCRIPTION("callback,add");
```
What is Eo?

Unique Features

- Pointer indirection (at least in C)
- Multiple calls in one context
- How we do constructors (setting properties, no constructors)
- Named ref-counting
- Composite objects
- Default return values
What is Eo?

Unique Features

- Pointer indirection (at least in C)
What is Eo?

Unique Features

- Pointer indirection (at least in C)
- Multiple calls in one context
What is Eo?

Unique Features

- Pointer indirection (at least in C)
- Multiple calls in one context
- How we do constructors (setting properties, no constructors)
What is Eo?

Unique Features

- Pointer indirection (at least in C)
- Multiple calls in one context
- How we do constructors (setting properties, no constructors)
- Named ref-counting
What is Eo?

Unique Features

- Pointer indirection (at least in C)
- Multiple calls in one context
- How we do constructors (setting properties, no constructors)
- Named ref-counting
- Composite objects
Unique Features

- Pointer indirection (at least in C)
- Multiple calls in one context
- How we do constructors (setting properties, no constructors)
- Named ref-counting
- Composite objects
- Default return values
Wash, Rinse, Repeat
Wash, Rinse, Repeat

- Eo1
Reception

Wash, Rinse, Repeat

- Eo1
- Eo2
Wash, Rinse, Repeat

- Eo1
- Eo2
- Eolian
Wash, Rinse, Repeat

- Eo1
- Eo2
- Eolian
- Eolian (improved)
Impact

Stability
Impact

Stability

- Pointer indirection saved us in many cases
Stability

- Pointer indirection saved us in many cases
- We caught a lot of errors that were not noticed before
Impact

Stability

- Pointer indirection saved us in many cases
- We caught a lot of errors that were not noticed before
- Single point of access for type checking makes it impossible to forget
Impact

Reduced API

Before:
- `evas_object_image_file_set(obj, "blah.png", "key");`
- `edje_object_file_set(obj, "blah.edj", "group");`
- `evas_object_del(obj);`
- `ecore_timer_del(obj);`
- `ecore_animator_del(obj);`

Now:
- `eo_do(obj, efl_file_set("blah.file", "key");`  
- `eo_del(obj);`
Impact

Reduced API

Before:

```c
evas_object_image_file_set(obj, "blah.png", "key");
edje_object_file_set(obj, "blah.edj", "group");

evas_object_del(obj);
ecore_timer_del(obj);
ecore_animator_del(obj);
```
Impact

Reduced API

Before:

evas_object_image_file_set(obj, "blah.png", "key");
edje_object_file_set(obj, "blah.edj", "group");

evas_object_del(obj);
ecore_timer_del(obj);
ecore_animator_del(obj);

Now:

eo_do(obj, efl_file_set("blah.file", "key");

eo_del(obj);
But writing objects in C is tedious!
But writing objects in C is tedious!

- The answer: Eolian
But writing objects in C is tedious!

- The answer: Eolian
- Eolian parses Eo API declarations
But writing objects in C is tedious!

- The answer: Eolian
- Eolian parses Eo API declarations
- Eolian allows for automated binding generators
But writing objects in C is tedious!

- The answer: Eolian
- Eolian parses Eo API declarations
- Eolian allows for automated binding generators
- Eolian is meant to be familiar for everyone
Eolian

A new format?
Eolian

A new format?

- Language independent $\rightarrow$ easy bindings
Eolian

A new format?

- Language independent → easy bindings
- Familiar syntax → easy to pick up
Eolian

A new format?

- Language independent → easy bindings
- Familiar syntax → easy to pick up
- Easy to read and write
Eolian

A new format?

- Language independent → easy bindings
- Familiar syntax → easy to pick up
- Easy to read and write
- Declarative and descriptive
class Namespace.Class (inherits) {
  methods { ... }
  properties { ... }
  events { ... }
  implements { ... }
  constructors { ... }
}

type Type_Name: Type_Def;
struct Struct_Name { ... }
enum Enum_Name { ... }
methods {
    method_name @class @protected {
        params {
            @in int x;
            @out const(char) *y;
        }
        return: own(char*);
    }
}
properties {
    property_name {
        keys {
            list<int>* x;
        }
        values {
            int v;
        }
        get {}
        set {}
    }
}

Eolian

Generators!

- Initial generator: C
- Further generators in core EFL: C++ and Lua
- Third party generators: Python, efforts being put into Rust, OCaml
- Future generators include JavaScript and others
Generators!

- Initial generator: C

Further generators in core EFL: C++ and Lua

Third party generators: Python, efforts being put into Rust, OCaml

Future generators include JavaScript and others
Generators!

- Initial generator: C
- Further generators in core EFL: C++ and Lua
Generators!

- Initial generator: C
- Further generators in core EFL: C++ and Lua
- Third party generators: Python, efforts being put into Rust, OCaml
Generators!

- Initial generator: C
- Further generators in core EFL: C++ and Lua
- Third party generators: Python, efforts being put into Rust, OCaml
- Future generators include JavaScript and others
Eolian

The Eolian library
The Eolian library

- C API: simple and easy to use
The Eolian library

- C API: simple and easy to use
- Minimum of non-standard data types → easy to bind
The Eolian library

- C API: simple and easy to use
- Minimum of non-standard data types → easy to bind
- Not only for generators (IDEs... )
Eolian

The Eolian library

- C API: simple and easy to use
- Minimum of non-standard data types → easy to bind
- Not only for generators (IDEs...)
- Simple database
However...
However...

- Some things still missing
However...

- Some things still missing
- Documentation?
However...

- Some things still missing
- Documentation?
- Value ownership
However...

- Some things still missing
- Documentation?
- Value ownership
- And possibly others
However...

- Some things still missing
- Documentation?
- Value ownership
- And possibly others
Eolian

However...

▶ Some things still missing
▶ Documentation?
▶ Value ownership
▶ And possibly others

And yet...
However...  

- Some things still missing  
- Documentation?  
- Value ownership  
- And possibly others  

And yet...  

- Very useful
However...

- Some things still missing
- Documentation?
- Value ownership
- And possibly others

And yet...

- Very useful
- Generic
However...

- Some things still missing
- Documentation?
- Value ownership
- And possibly others

And yet...

- Very useful
- Generic
- I’d like to get it adopted by others (non EFL)
Other Projects

Clouseau

▶ Application state inspector for the EFL
▶ Was not created following Eo (but greatly improved)
▶ Will get even better with Eolian
Clouseau

- Application state inspector for the EFL
Clouseau

- Application state inspector for the EFL
- Was not created following Eo (but greatly improved)
Other Projects

Clouseau

- Application state inspector for the EFL
- Was not created following Eo (but greatly improved)
- Will get even better with Eolian
Other Projects

Erigo

▶ EFL GUI builder
▶ Reads properties from Eolian
▶ Supports whatever version is installed on the system automatically
▶ Supports widgets that it has no notion of
▶ Has its own format that is processed by language specific code generators
Other Projects

Eriso

- EFL GUI builder
Other Projects | Erigo

- EFL GUI builder
- Reads properties from Eolian
Other Projects

Eriso

- EFL GUI builder
- Reads properties from Eolian
  - Supports whatever version is installed on the system automatically
Other Projects

Erigo

- EFL GUI builder
- Reads properties from Eolian
  - Supports whatever version is installed on the system automatically
  - Supports widgets that it has no notion of
Other Projects

EriGo

- EFL GUI builder
- Reads properties from Eolian
  - Supports whatever version is installed on the system automatically
  - Supports widgets that it has no notion of
- Has its own format that is processed by language specific code generators
Resources Attributions

- Nothing