Eolian - Past year experiences and the future

Shaping up our binding generation infrastructure

Daniel Kolesa
Samsung Open Source Group
d.kolesa@osg.samsung.com
@octaforge
EFL Developer Day 2015
Introduction

For those living under a rock
What is Eolian?

- Most people are probably already aware of it
- It's several things
  - It's a declarative format for describing APIs
  - It's a C library to deal with these declarations
  - It's a generator for the core C API
What is Eolian?

- Most people are probably already aware of it
What is Eolian?

- Most people are probably already aware of it
- It’s several things
What is Eolian?

- Most people are probably already aware of it
- It’s several things
- It’s a declarative format for describing APIs
What is Eolian?

- Most people are probably already aware of it
- It’s several things
- It’s a declarative format for describing APIs
- It’s a C library to deal with these declarations
What is Eolian?

- Most people are probably already aware of it
- It’s several things
- It’s a declarative format for describing APIs
- It’s a C library to deal with these declarations
- It’s a generator for the core C API
Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language
- Improved documentation possibilities
- Better tooling
- The possibilities are endless
Why is Eolian useful?

- Language independent API descriptions
Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language
Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language
- Improved documentation possibilities
Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language
- Improved documentation possibilities
- Better tooling

The possibilities are endless
Why is Eolian useful?

- Language independent API descriptions
- Automatic generation of bindings for any language
- Improved documentation possibilities
- Better tooling
- The possibilities are endless
Past and present
How was Eolian and how is it now?
Last year Eolian
How was it?

- Very basic
- Supported C generation and nothing else
- A messy parser/lexer created with a generator
- A lot of design problems
How was it?

- Very basic
How was it?

- Very basic
- Supported C generation and nothing else
How was it?

- Very basic
- Supported C generation and nothing else
- A messy parser/lexer created with a generator
How was it?

- Very basic
- Supported C generation and nothing else
- A messy parser/lexer created with a generator
- A lot of design problems
How did we fix it?

- First project: write a new parser from scratch
- Gradual rewrite of Eolian from ground up
- Fixed design problems one by one
- Through iterative approach, we got where we are currently
How did we fix it?

- First project: write a new parser from scratch
How did we fix it?

- First project: write a new parser from scratch
- Gradual rewrite of Eolian from ground up
How did we fix it?

- First project: write a new parser from scratch
- Gradual rewrite of Eolian from ground up
- Fixed design problems one by one
How did we fix it?

- First project: write a new parser from scratch
- Gradual rewrite of Eolian from ground up
- Fixed design problems one by one
- Through iterative approach, we got where we are currently
Main challenges of last year and the present
Eolian type system

Old Eolian used C strings to do types
No verification
Not suitable for other languages
We needed our own type system
Eolian type system

- Old Eolian used C strings to do types
Eolian type system

- Old Eolian used C strings to do types
- No verification
Eolian type system

- Old Eolian used C strings to do types
- No verification
- Not suitable for other languages
Eolian type system

- Old Eolian used C strings to do types
- No verification
- Not suitable for other languages
- We needed our own type system
Real types represented in our in-memory database

Basic type checking/verification (usage contexts etc.)

Integration with variables/constants

Structures and enumerations

Type aliases

Extern types and other smaller features
Real types represented in our in-memory database
- Real types represented in our in-memory database
- Basic type checking/verification (usage contexts etc.)
- Real types represented in our in-memory database
- Basic type checking/verification (usage contexts etc.)
- Integration with variables/ constants
- Real types represented in our in-memory database
- Basic type checking/verification (usage contexts etc.)
- Integration with variables/constants
- Structures and enumerations
- Real types represented in our in-memory database
- Basic type checking/verification (usage contexts etc.)
- Integration with variables/constants
- Structures and enumerations
- Type aliases
▶ Real types represented in our in-memory database
▶ Basic type checking/verification (usage contexts etc.)
▶ Integration with variables/constants
▶ Structures and enumerations
▶ Type aliases
▶ Extern types and other smaller features
Custom constructors

Old Eolian supported custom constructors for classes

We decided this is a bad idea

We generate constructors using constructing functions now
Custom constructors

- Old Eolian supported custom constructors for classes
Custom constructors

- Old Eolian supported custom constructors for classes
- We decided this is a bad idea
Custom constructors

- Old Eolian supported custom constructors for classes
- We decided this is a bad idea
- We generate constructors using constructing functions now
Syntax revision

Old syntax was very messy and inconsistent

Complete refactoring of syntax was done

Unification and stricter parsing

Focus on consistency
Syntax revision

- Old syntax was very messy and inconsistent
Syntax revision

- Old syntax was very messy and inconsistent
- Complete refactoring of syntax was done
Syntax revision

- Old syntax was very messy and inconsistent
- Complete refactoring of syntax was done
- Unification and stricter parsing
Syntax revision

- Old syntax was very messy and inconsistent
- Complete refactoring of syntax was done
- Unification and stricter parsing
- Focus on consistency
Complex types

At first, we had generalized complex types

This proved to be a bad idea

Provided several built-in complex types (array, hash etc.)

Greatly simplified generation and syntax
Complex types

- At first, we had generalized complex types
Complex types

- At first, we had generalized complex types
- This proved to be a bad idea
Complex types

- At first, we had generalized complex types
- This proved to be a bad idea
- Provided several built-in complex types (array, hash etc.)
Complex types

- At first, we had generalized complex types
- This proved to be a bad idea
- Provided several built-in complex types (array, hash etc.)
- Greatly simplified generation and syntax
Data sharing

- We needed to share typedefs etc. between several classes
- Introduced .eot (eo type) files
- And introduced an import statement
Data sharing

- We needed to share typedefs etc. between several classes
Data sharing

- We needed to share typedefs etc. between several classes
- Introduced .eot (eo type) files
Data sharing

- We needed to share typedefs etc. between several classes
- Introduced .eot (eo type) files
- And introduced an import statement
Documentation
Documentation

▶ Ongoing issue
Documentation

- Ongoing issue
- Doxygen will no longer cut it (specific to C/C++)

Still being extended and improved
Documentation

- Ongoing issue
- Doxygen will no longer cut it (specific to C/C++)
- Designed a custom format
Ongoing issue

Doxygen will no longer cut it (specific to C/C++)

Designed a custom format

Removed documentation comments, introduced special syntax
Documentation

- Ongoing issue
- Doxygen will no longer cut it (specific to C/C++)
- Designed a custom format
- Removed documentation comments, introduced special syntax
- Converted all EFL/Elementary docs to the new format
Ongoing issue
Doxygen will no longer cut it (specific to C/C++)
Designed a custom format
Removed documentation comments, introduced special syntax
Converted all EFL/Elementary docs to the new format
Still being extended and improved
Current situation

▶ We're mostly done
▶ API is nice and clean
▶ All major issues are solved
▶ Documentation still ongoing
▶ Ownership handling still ongoing
Current situation

- We’re mostly done
Current situation

- We’re mostly done
- API is nice and clean
Current situation

- We’re mostly done
- API is nice and clean
- All major issues are solved
Current situation

- We’re mostly done
- API is nice and clean
- All major issues are solved
- Documentation still ongoing
Current situation

- We’re mostly done
- API is nice and clean
- All major issues are solved
- Documentation still ongoing
- Ownership handling still ongoing
The future

What are we going to do now?
Stabilize!

Stabilization is the primary goal
Not happening for a few more releases
We need to finalize documentation
We need to handle ownership
Refactor the implementation and fix all quirks
Stabilize!

- Stabilization is the primary goal
Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases
Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases
- We need to finalize documentation
Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases
- We need to finalize documentation
- We need to handle ownership
Stabilize!

- Stabilization is the primary goal
- Not happening for a few more releases
- We need to finalize documentation
- We need to handle ownership
- Refactor the implementation and fix all quirks
More testing and changes
More testing and changes

- We need more generators to help us test
More testing and changes

- We need more generators to help us test
- We also need to update all of EFL eo files
More testing and changes

- We need more generators to help us test
- We also need to update all of EFL eo files
- This should help uncover any potential problems
Thank you.

Daniel Kolesa
Samsung Open Source Group
d.kolesa@osg.samsung.com
@octaforge
EFL Developer Day 2015