Building a Graphical IDE in Elm/Purescript

for a Distributed PLC Language Compiling to BEAM

by @doppioslash

04/11/2016 - Codemesh - London

DIPL. PHYS. PEER STRITZINGER GMBH

Hi, I'm

Claudia Doppioslash

Functional Programmer & **Game Developer**

@doppioslash
www.lambdacat.com

DIPL. PHYS. PEER STRITZINGER GMBH

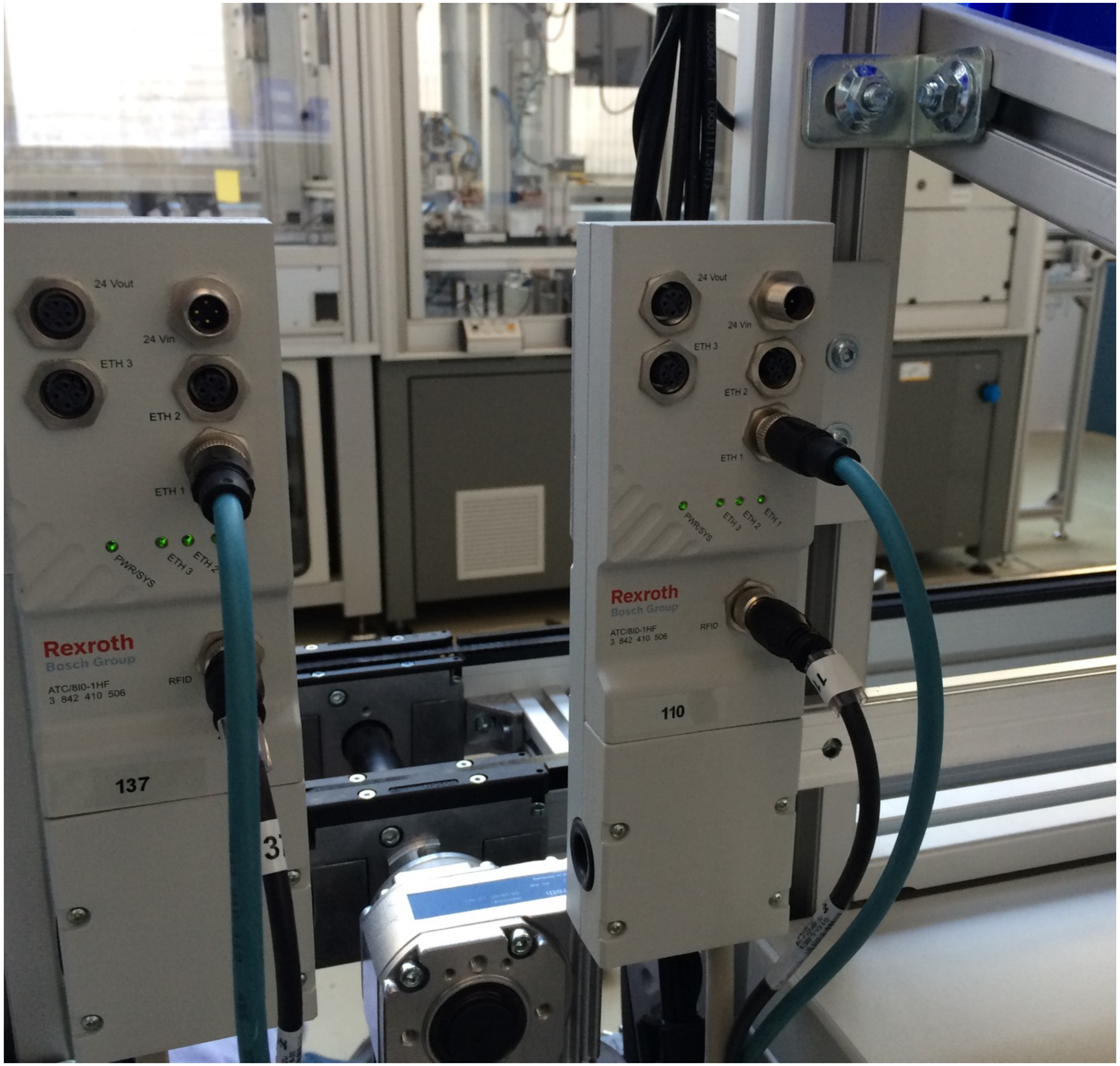
Peer Stritzinger GmbH

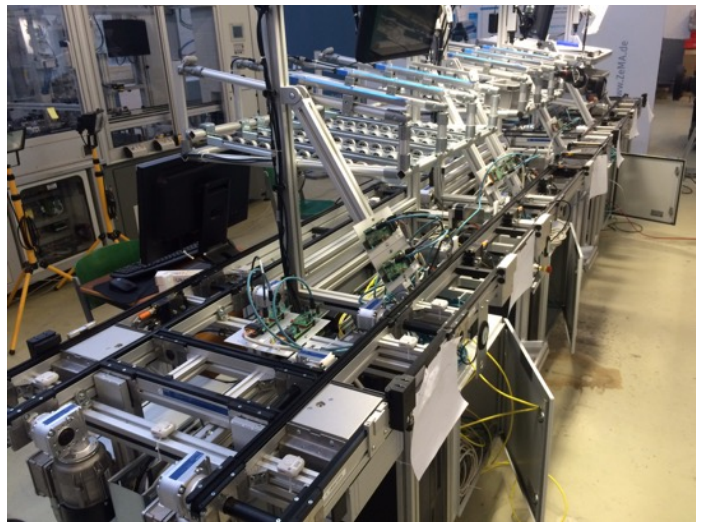
Functional and Failure Tolerant Programming for Embedded, Industrial Control and Automotive

DIPL. PHYS. PEER STRITZINGER GMBH

www.stritzinger.com

DIPL. PHYS. PEER STRITZINGER GMBH

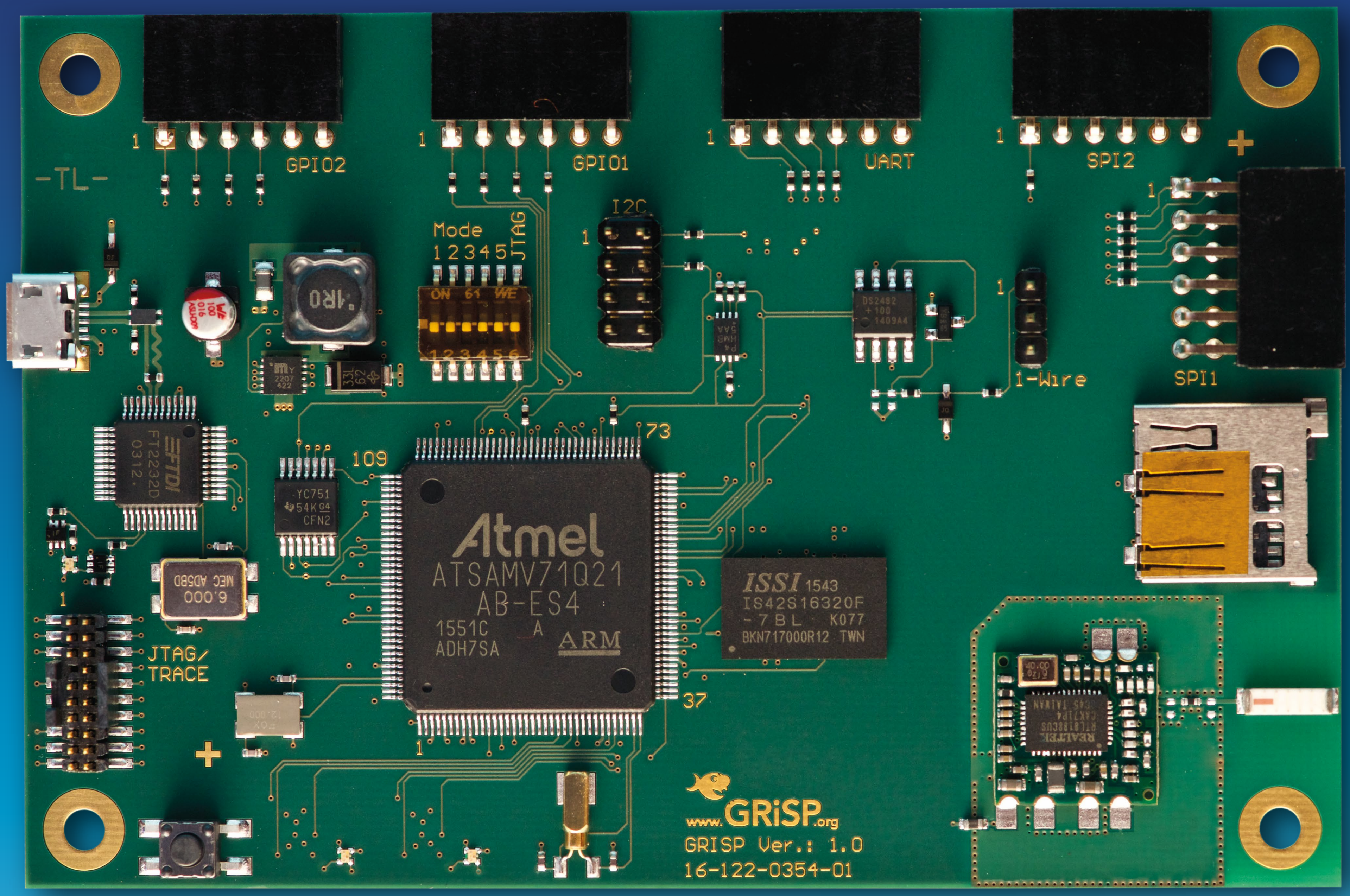

24 Vout
24 Vin
ETH 3
ETH 2
ETH 1
PWR/SYS
Rexroth
Bosch Group
ATC/8I0-1HF
3 842 410 506
RFID
137
110



CeBIT
future talk
Intelligente Cyber-physische IT-Systeme

Rexroth

DIPL. PHYS. PEER STRITZINGER GMBH



www.grisp.org

ERLANG
Embedded With
RTEMS
www.rtems.com
GRiSP

LISP *flavored* ERLANG

ERLANG

GRiSP

elixir

luerl

Why are you here?

“I need to get some frontend code done, and I hate Javascript”

Interested in Haskell-like languages

Undecided between Elm and Purescript

What are you getting

This is a WIP-mortem:

- why we made the choices we made
- what went right/wrong
- enough Elm to understand what’s going on
- our experience of porting from Elm to Purescript

Not an Elm or Purescript guide, also not latest Elm version.

0.16? 0.17?

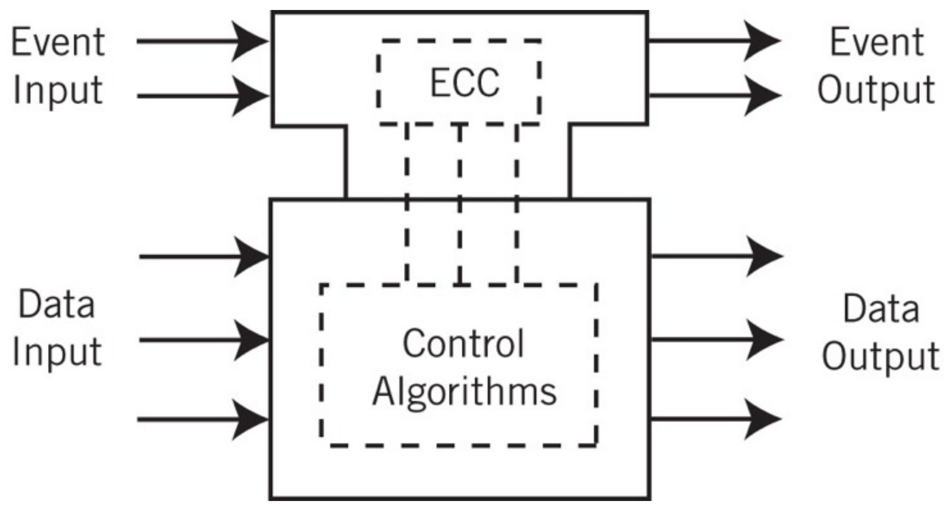
The jump from 0.16 and 0.17 in Elm

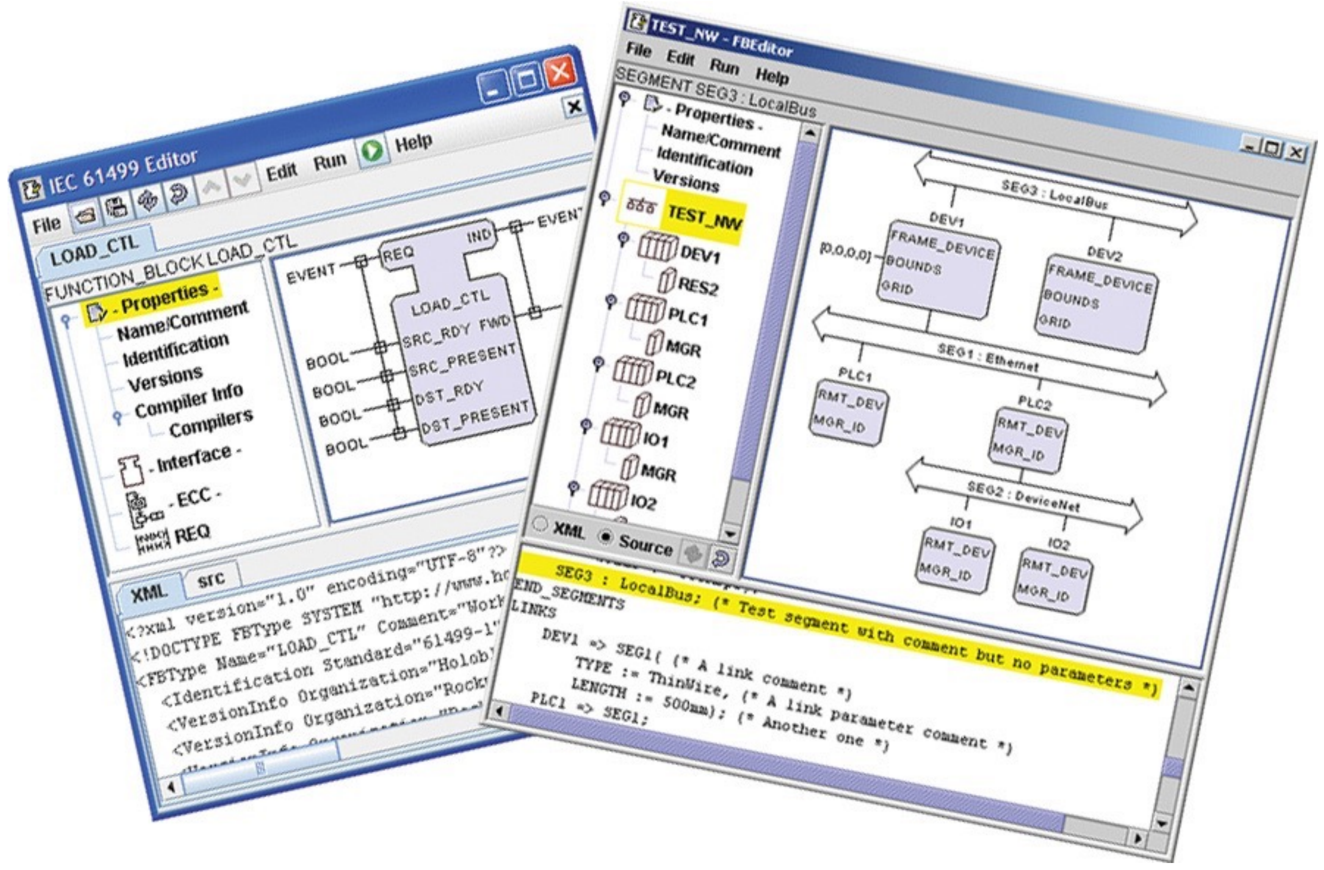
0.16	**0.17**
FRP	
mailboxes	
addresses	
signals	
foldp	

Our Project

Visual IDE for PLC language **IEC61499**

“A programmable logic controller, PLC, or programmable controller is a digital computer used for automation"





```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "http://www.hc
<FBType Name="LOAD_CTL" Comment="Work
<Identification Standard="61499-1"
  <VersionInfo Organization="Holob
  <VersionInfo Organization="Rock
```

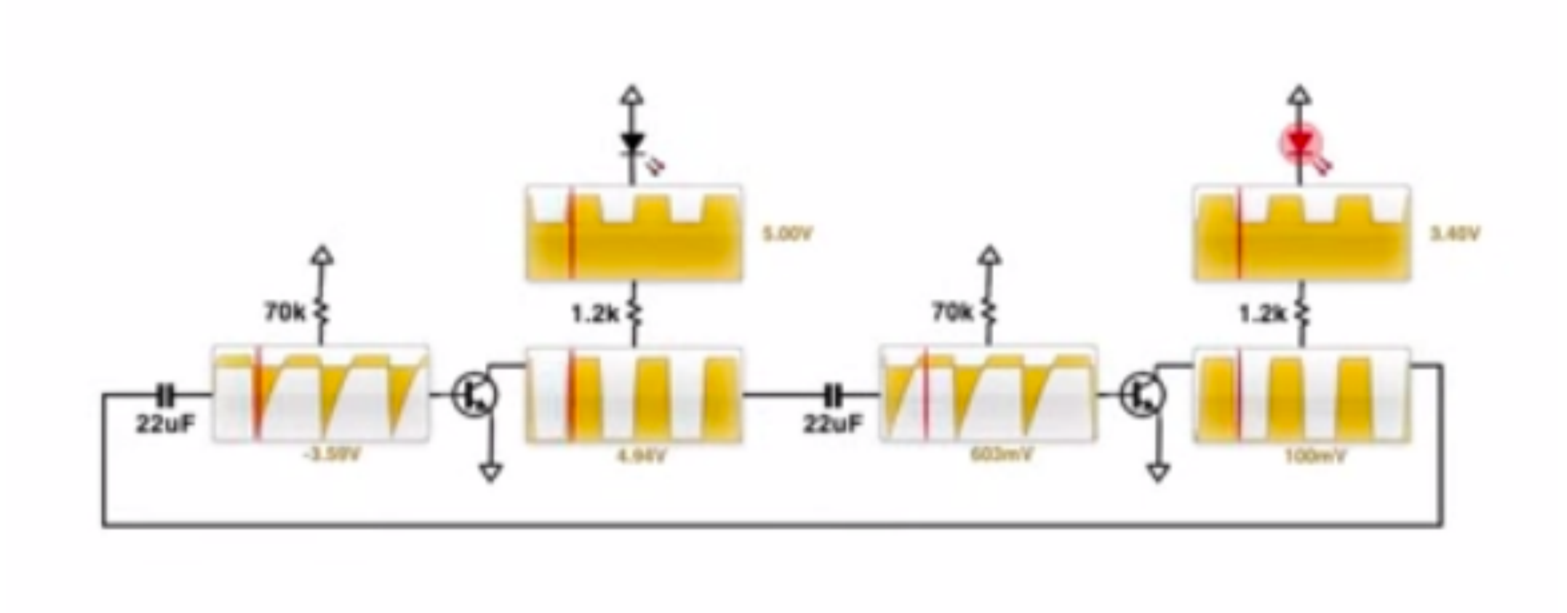
```
  SEG3 : LocalBus; (* Test segment with comment but no parameters *)
END_SEGMENTS
LINKS
   DEV1 => SEG1( (* A link comment *)
      TYPE := ThinWire, (* A link parameter comment *)
      LENGTH := 500mm); (* Another one *)
   PLC1 => SEG1;
```

(images from http://www.controldesign.com/articles/2007/202/?show=all)

DIPL. PHYS. PEER STRITZINGER GMBH

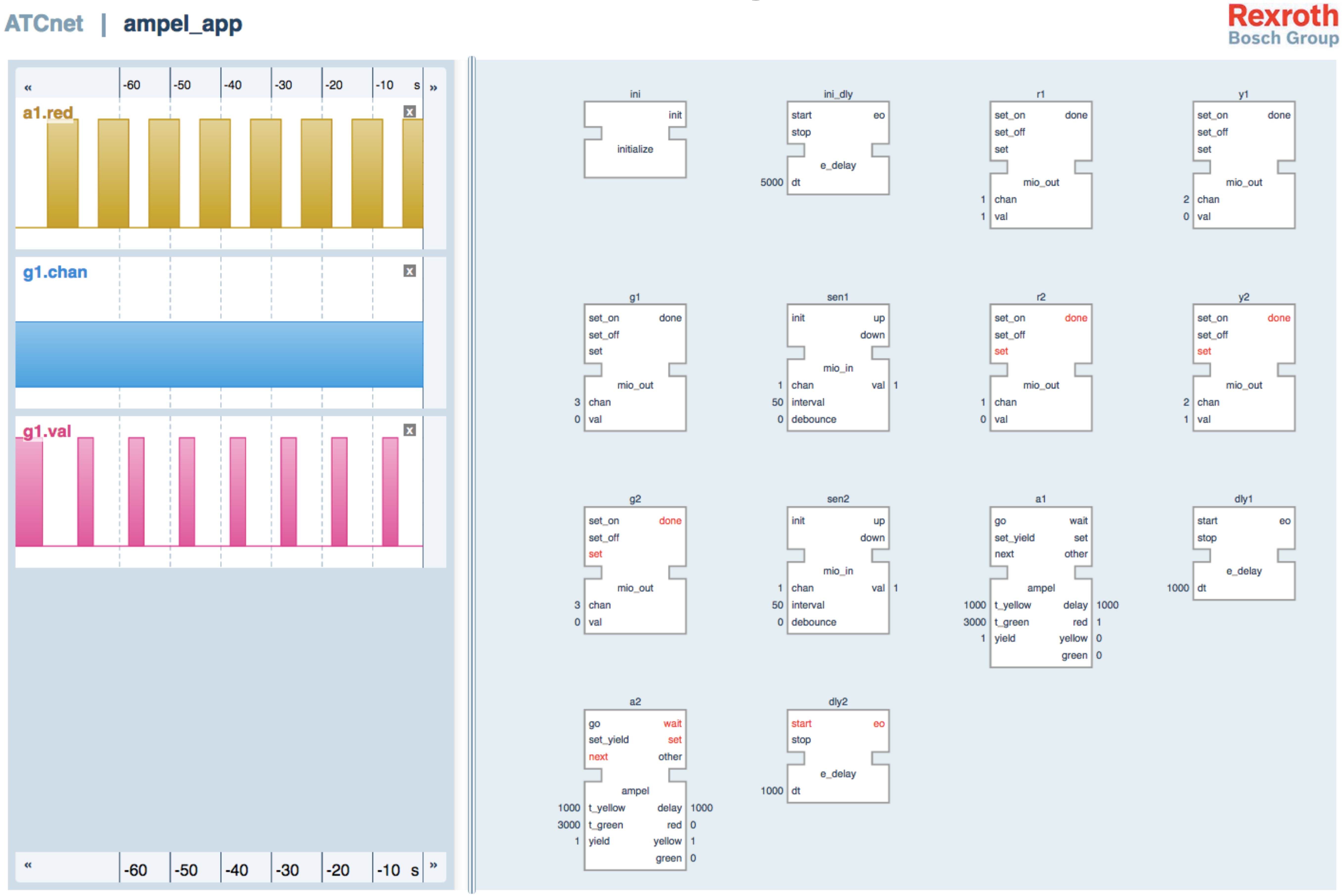
Our Project

Inspired by Bret Victor's "Inventing on Principle" talk:

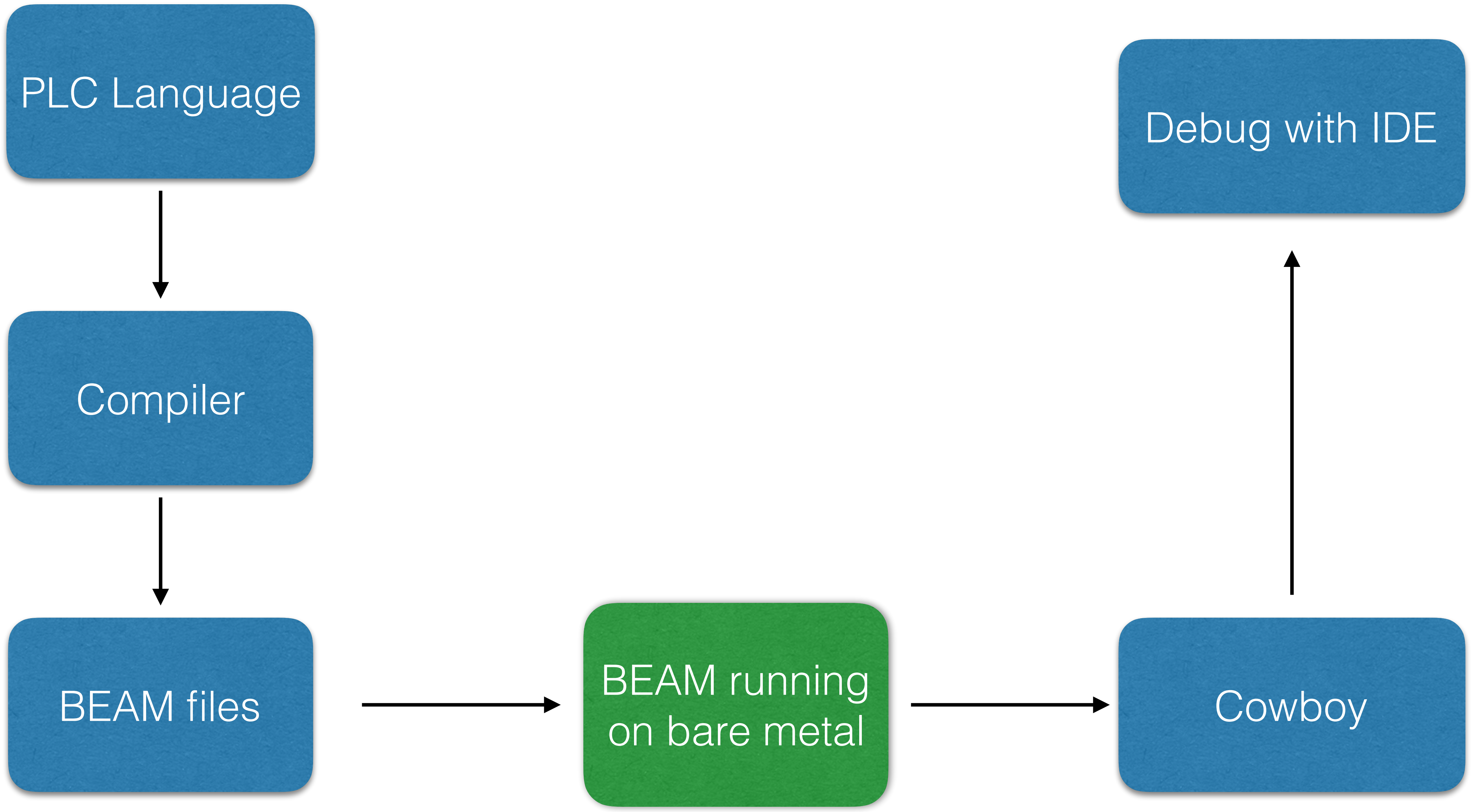


DIPL. PHYS. PEER STRITZINGER GMBH

Our Project


ATCnet | ampel_app
Rexroth Bosch Group
« -60 -50 -40 -30 -20 -10 s »
a1.red
g1.chan
g1.val
ini: init, initialize
ini_dly: start, stop, eo, e_delay, 5000 dt
r1: set_on, set_off, set, done, mio_out, 1 chan, 1 val
y1: set_on, set_off, set, done, mio_out, 2 chan, 0 val
g1: set_on, set_off, set, done, mio_out, 3 chan, 0 val
sen1: init, up, down, mio_in, 1 chan, val 1, 50 interval, 0 debounce
r2: set_on, set_off, set, done, mio_out, 1 chan, 0 val
y2: set_on, set_off, set, done, mio_out, 2 chan, 1 val
g2: set_on, set_off, set, done, mio_out, 3 chan, 0 val
sen2: init, up, down, mio_in, 1 chan, val 1, 50 interval, 0 debounce
a1: go, set_yield, next, wait, set, other, ampel, 1000 t_yellow, 3000 t_green, 1 yield, delay 1000, red 1, yellow 0, green 0
dly1: start, stop, eo, e_delay, 1000 dt
a2: go, set_yield, next, wait, set, other, ampel, 1000 t_yellow, 3000 t_green, 1 yield, delay 1000, red 0, yellow 1, green 0
dly2: start, stop, eo, e_delay, 1000 dt

Our Project



DIPL. PHYS. PEER STRITZINGER GMBH

Requirements

Many platforms to support

All PC OSs & iPad Pro

Decent performance

Needs to be interactive
~30fps should be fine

Frontend Tech Choice

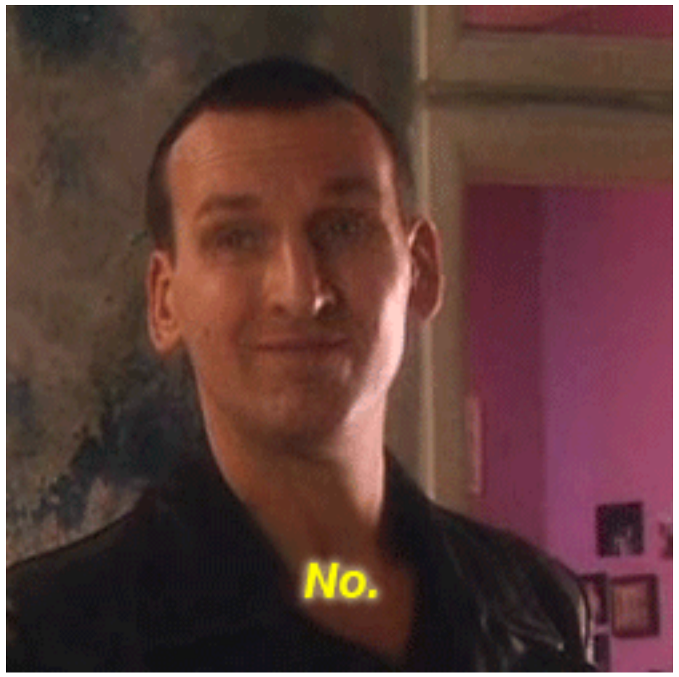
Web Technologies because cross-platform

Hence: **Javascript, CSS, Svg**

Wait a minute, Javascript?

DIPL. PHYS. PEER STRITZINGER GMBH

Wait a minute, Javascript?



…let's not.

Possible Choices, Then

Ready at the time:

Clojurescript

CoffeScript

Elm

Typescript

Why did we chose Elm?

Functional Reactive Programming

(it’s gone now though)

Good error messages

(so good everyone is imitating them)

Some concepts somewhat similar to Erlang

(e.g. Mailboxes)

What is Elm?

Pure Functional

Strongly Typed

Eagerly evaluated

Compiles to **Javascript**

Functional Reactive Programming (< 0.17)

Haskell-like syntax

Very **small**

Optimised for **learning curve** (>0.16)

Similar to Haskell but no advanced types

Elm package manager enforces **semantic versioning**

DIPL. PHYS. PEER STRITZINGER GMBH

Elm Pros compared to JS

If it compiles, it works (90% of the time)

Confident **refactoring**

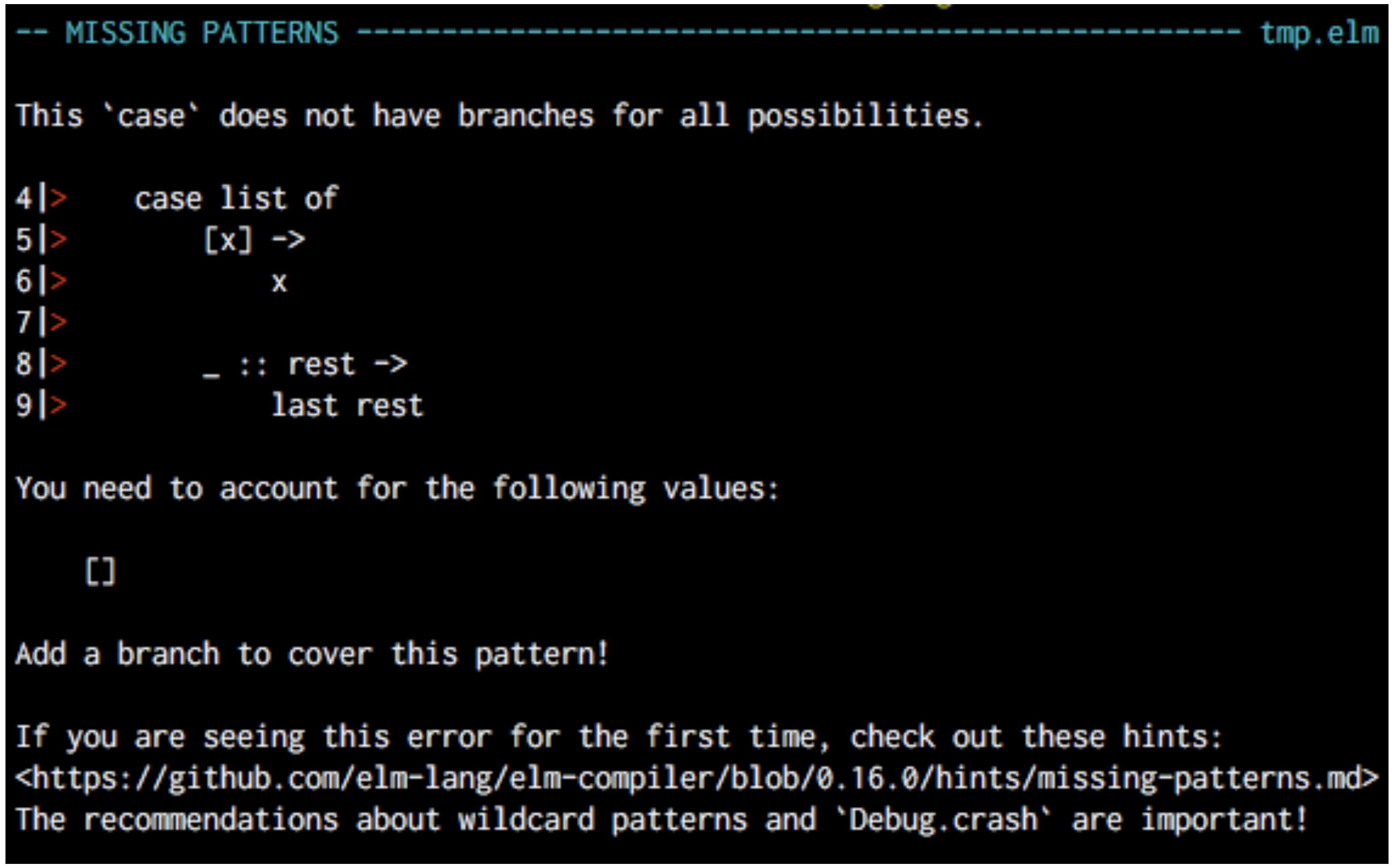
Clean

Much fewer LOC

The famous great error messages
(better than undefined is not a function)

The famous Elm errors

- contextual
- correct common errors



```
-- MISSING PATTERNS ------------------------------------------------------ tmp.elm

This `case` does not have branches for all possibilities.

4|>    case list of
5|>        [x] ->
6|>            x
7|>
8|>        _ :: rest ->
9|>            last rest

You need to account for the following values:

    []

Add a branch to cover this pattern!

If you are seeing this error for the first time, check out these hints:
<https://github.com/elm-lang/elm-compiler/blob/0.16.0/hints/missing-patterns.md>
The recommendations about wildcard patterns and `Debug.crash` are important!
```

The famous Elm errors

How do they do it?

- make it a priority
- carefully tracked on a git repo
- type system complexity
  (simpler = easier to have good errors)

The famous Elm errors

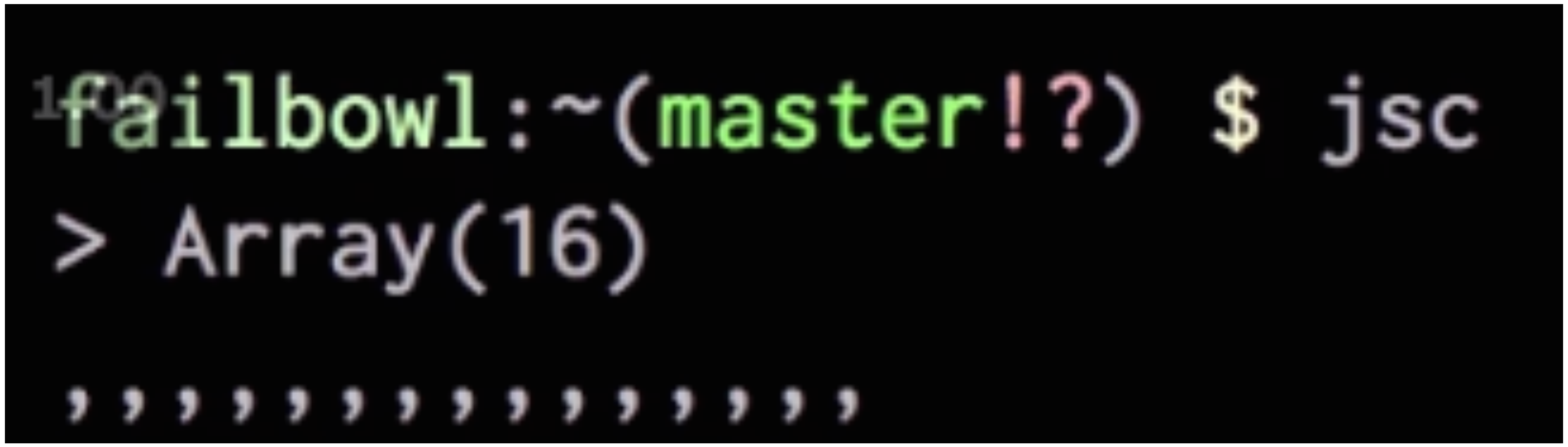
But

you can **call** something wrong
or **define** something wrong

and sometimes it thinks it’s a wrong definition
when it is actually incorrect use

Elm Pros compared to JS

Elm actually makes sense (seen the ‘**Wat**’ talk?)



```
failbowl:~(master!?) $ jsc
> Array(16)

,,,,,,,,,,,,,,,
```

DIPL. PHYS. PEER STRITZINGER GMBH

Elm Cons compared to JS

Javascript **interop inflexible**

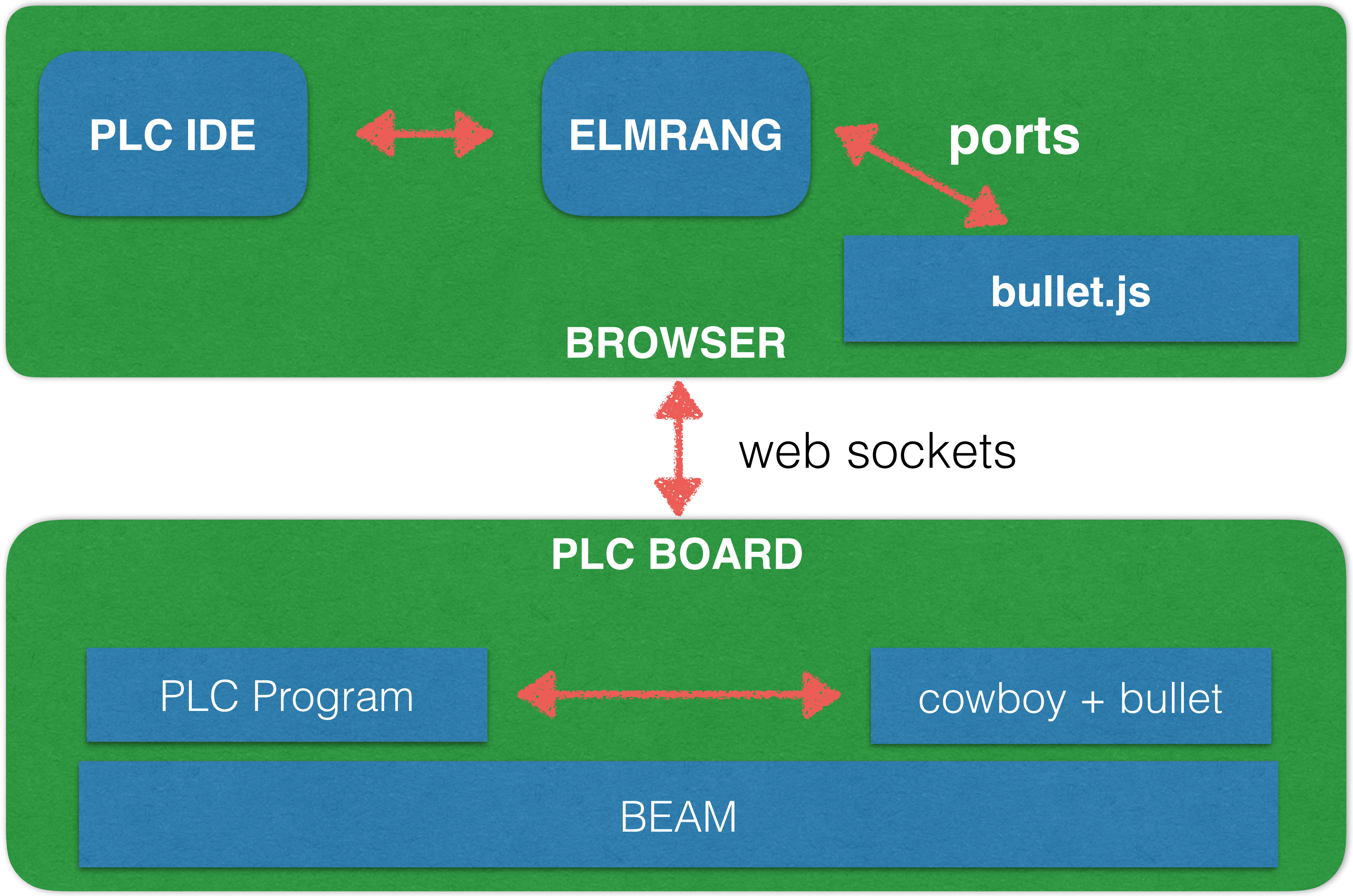
(less in 0.17)

new language, still 0.x

…so, not that much.

DIPL. PHYS. PEER STRITZINGER GMBH

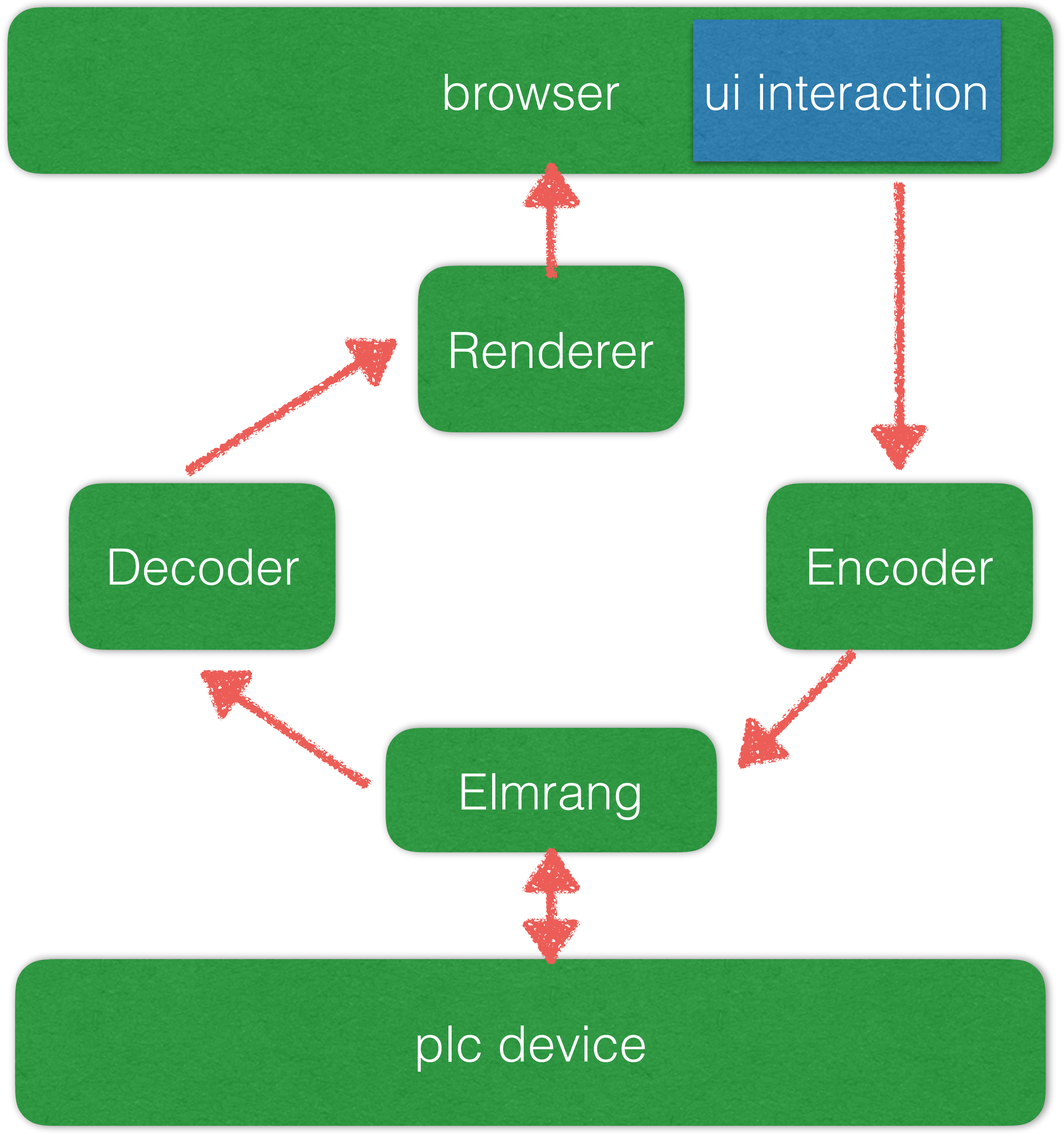
Our Project



DIPL. PHYS. PEER STRITZINGER GMBH

Demo

PLC IDE Structure



DIPL. PHYS. PEER STRITZINGER GMBH

What is StartApp?

Implementation of **The Elm Architecture** for **0.16**

In 0.17 it **is** the language

Action **Model** **Update** **View**

Beware: this is different in 0.17

What is StartApp?

Action

```
type Action
= Increment
| Decrement
```

Just a Union Type (aka ADT, etc)

What is StartApp?

Model

```
type alias Model = Int
```

A type alias

What is StartApp?

Update

```
update : Action -> Model -> Model
update action model =
  case action of
    Increment -> model + 1
    Decrement -> model - 1
```

Returns the new model state

DIPL. PHYS. PEER STRITZINGER GMBH

What is StartApp?

View

```
view : Address -> Model -> Html
view address model =
  p [] [text model]
```

Returns html

DIPL. PHYS. PEER STRITZINGER GMBH

PLC IDE Structure

Four **StartApp** connected by **Mailboxes**

Wired into a parent StartApp, so nested StartApps

As in the structure invented by **foxdonut**

Easy to expand, add components

But no one ported it to 0.17 (may be impossible)

Elmrang can be a component using this structure

Why are we still on 0.16?

We use **FRP** heavily

Porting code might not be **cost effective**

Frustrated with **lack of communication**
(e.g. no deprecation warnings)

Waiting for Elm evolution to **stabilise**

DIPL. PHYS. PEER STRITZINGER GMBH

Production Problems

How to organise subcomponents in a big Elm app?

How to store deps not on elm-package?

How to include an Elm project into an Erlang app?

The file structure

Every component has:

```
component/Action.elm
component/Model.elm
component/View.elm
component/Update.elm
component/Feature.elm
```

Wired in in App.elm and fed to Main.elm

Non elm-package deps

- fetch it from repo
- store it in a subdir of the erlang project
- move only the elm files to a subdir of the elm project
- not under elm-stuff
- include the subdir in elm-package.json

Mixed Elm/Erlang Project

- /elm subdir in Erlang project
- compiler Elm files to /priv
- add the .js to your html file

Rendering

Choices we had:

- WebGL (2d rendering engine)
- SVG (w or w/o CSS layout and animations)
- Html (not ideal)

DIPL. PHYS. PEER STRITZINGER GMBH

Rendering

We use **Svg with CSS**

We try to do as much as we can with CSS

Animation in Elm can get complicated

CSS styles are in separate CSS files

We have an Svg & CSS expert on call

Rendering

elm-html and **elm-svg** have great syntax:

```
div [class “somecssclass”]
    [ p [] [text “a very well written paragraph”]
    , p [] [text “and another one”]
    ]
```

Based on virtualdom = fast

DIPL. PHYS. PEER STRITZINGER GMBH

Several words to the wise

Be aware of what Elm is good for.

An Elm program has to fit the Elm Architecture (which is good if it does fits, less if it doesn't)

Wrapping Javascript libraries

There is no path to get a library that wraps a javascript library on elm-package (e.g. elm-d3)

Several words to the wise

Elm is still experimental

Elm is still subject to big changes, expect to have to rewrite some of your code with a new version.

Elm lacks a roadmap

There are short beta previews, and you can keep up by looking at the changes in the compiler.
Recently Evan started doing semi-regular updates of what he’s up to in the mailing list

What next?

We're going to skip 0.17

Maybe come back when Elm is nearer to 1.0

Meanwhile taking another language for a spin, porting a portion of our project to it

Possible Choices, Now

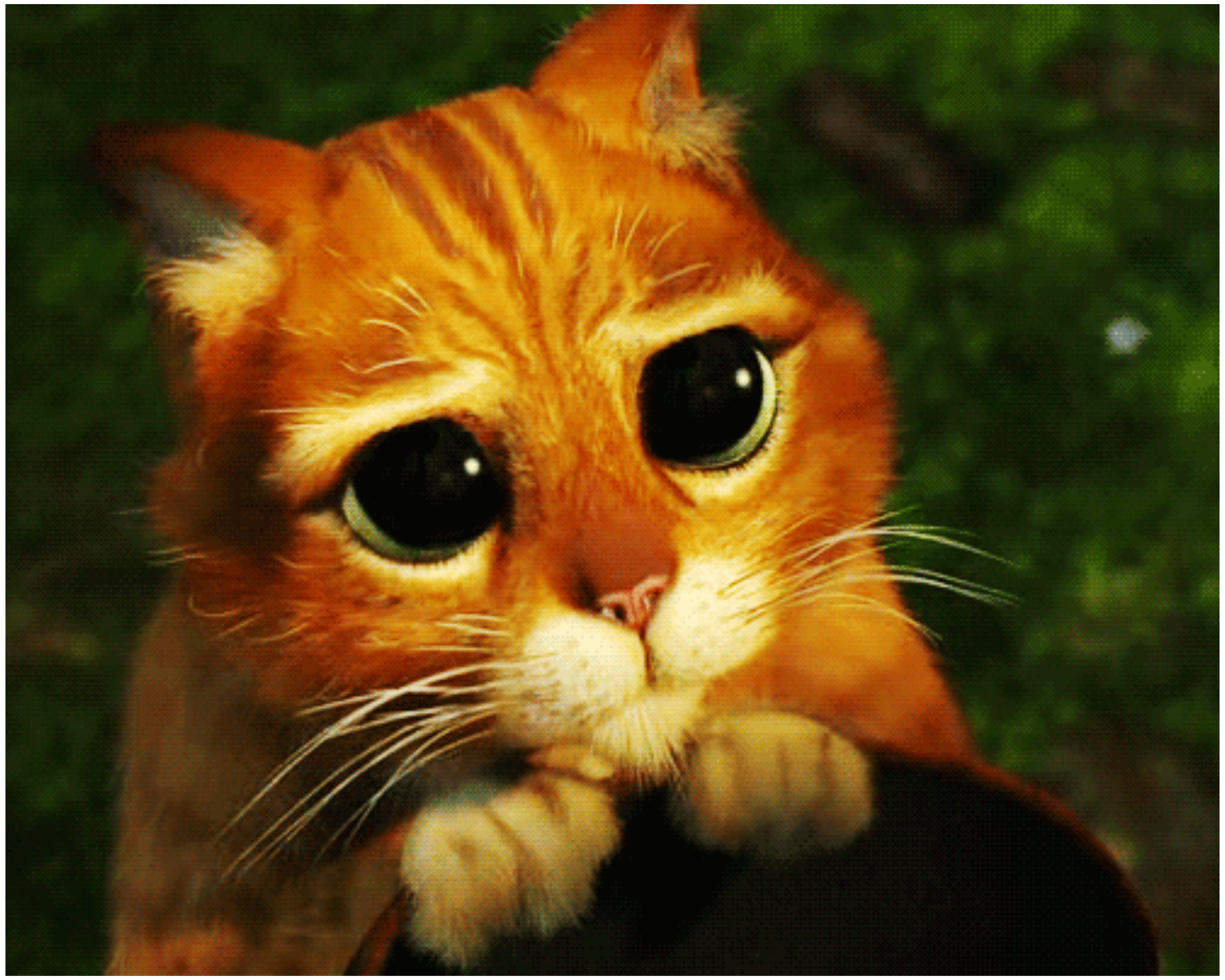
Ready now:

Bucklescript	Purescript
Clojurescript	Elm
CoffeScript	Typescript
Reason	Fable

YOU MUST CHOOSE...
BUT CHOOSE WISELY.
quickmeme.com

Our First Choice

: “Please adopt me…”



“…I swear I won’t mention Monads”

DIPL. PHYS. PEER STRITZINGER GMBH

Our Second Choice

Purescript : **“you’re free to do anything…**

…if you can cope with the types”

DIPL. PHYS. PEER STRITZINGER GMBH

What is Purescript?

Reminds you of anything?

Pure Functional

Strongly Typed

Eagerly evaluated

Compiles to Javascript

Advanced Types

Haskell-like syntax (with all the squiggles)

Generates readable Javascript, has no runtime

Open community, a bit of a roadmap

Philosophy Differences

Elm is made to be simple above anything else, have a quick learning curve

In Purescript you have most of the type features you have in Haskell, longer learning curve

Philosophy Differences

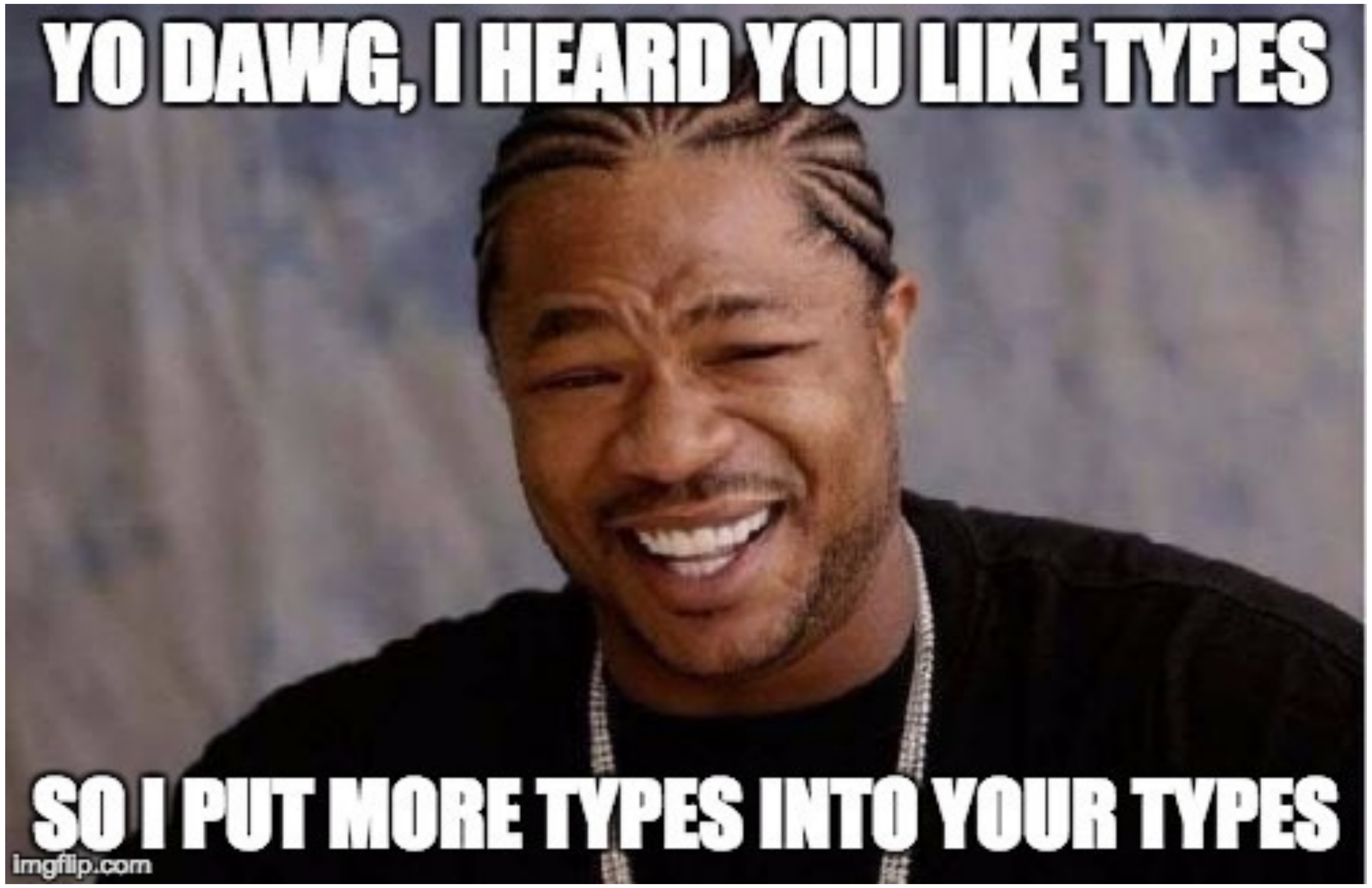
Elm gives you only one possible program structure (Elm arch)

In Purescript there are many possible ways of structuring your app

Why Purescript after Elm?

Why Purescript after Elm?

Exhibit 1: the type system is a great feature of Elm


YO DAWG, I HEARD YOU LIKE TYPES
SO I PUT MORE TYPES INTO YOUR TYPES
imgflip.com

Purescript's has more features. (Simplicity vs Power)

The Elm tradeoff

Preferring simpler types begets:

- smooth learning curve
- more boilerplate

Why Purescript after Elm?

- it’s similar enough that porting code is relatively straightforward
- once you get restless with Elm’s boilerplate, you’re likely ready for more powerful abstractions
- It's possible to implement Elm in it, but not the other way around
- It benefits from the hindsight of following Haskell from a distance
- Small, open community, communication still works

DIPL. PHYS. PEER STRITZINGER GMBH

Pros compared to Elm

Pursuit (search libs by type signature)

Clearer direction

Can work a lot with REPL

Type holes!!

All (well, many) of the cool abstractions

Cons

Takes time to learn the cool abstractions

DIPL. PHYS. PEER STRITZINGER GMBH

Reflection on Elm - Purescript - Haskell

- Simplest
- Focused on UX
- One way to do things
- Removes all historical baggage
- Great entry level language

- Most sensible
- UX is fairly good
- Still a lot of power
- Eagerly evaluated, hence simpler

- Research language
- Most powerful
- Least good UX
- Most historical baggage
- Laziness adds complexity

DIPL. PHYS. PEER STRITZINGER GMBH

Warning, 0.10 has just landed

It brings cool stuff, but breakage occurs while important libraries are still being ported

My advice: stick to 0.9.3 until 0.10 porting is complete, but still your deps will mismatch all over the place

DIPL. PHYS. PEER STRITZINGER GMBH

Bower

At the moment Purescript is relying on bower, which makes the time after a new release particularly annoying

But Phil's working on a new package manager

(Also, please everyone, let's try not to use github to host all our dependencies any more. It's asking for trouble and DDoSs)

Frameworks Overview

Wrapping React

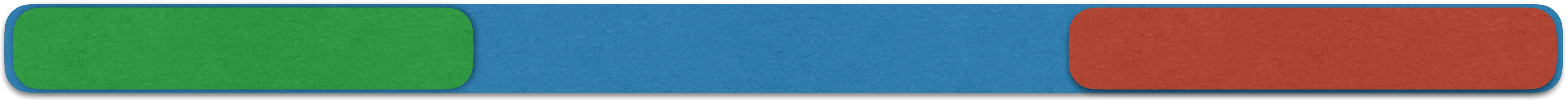
- Pux
- Thermite
- purescript-react

Pure

- Halogen
- Flare
- Optic UI

Frameworks

Type Complexity continuum



Why Flare?

- Great to start with
- Easy to make cool interactive graphs

Why not?

- Limited to a specific use case
- Need to understand applicative functor syntax:
  `thing <$> thing <*> thing`

DIPL. PHYS. PEER STRITZINGER GMBH

Why Pux?

Similar to the Elm architecture

Svg support already included

Interactive React debugger can be wired in

Probably the simplest Purescript framework

Why not?

React dependencies /0\

On the pain of installing React

(Though the React interactive debugger is nice)

Pux Structure

State	Action	update	inputs
		view	Effects

Compare with the Elm Architecture (0.16)

Model	Action	update	inputs
		view	Aff

Why Pux?

```
data Action = Increment | Decrement

type State = Int

update :: Action -> State -> State
update Increment state = state + 1
update Decrement state = state - 1

view :: State -> Html Action
view state =
  div
    []
    [ button [ onClick (const Increment) ] [ text "Increment" ]
    , span [] [ text (show state) ]
    , button [ onClick (const Decrement) ] [ text "Decrement" ]
    ]
```

DIPL. PHYS. PEER STRITZINGER GMBH

Thermite

Wraps React

Lenses and stuff

Optic UI

Pure Purescript

Lenses and stuff

DIPL. PHYS. PEER STRITZINGER GMBH

Why Halogen?

I'd rather not have to install the 300 React tools

It's used in production by Slamdata, on a pretty impressive app

> 1 people developing it

Nice Html DSL

Why not?

Argh, the types!! My eyes burn!

aka it's just a bit hard

Halogen Structure

State	Query action request	Component	eval render	main

Compare with StartApp (0.16)

Model	Action		update view	input Effects

Halogen Structure

```
-- | The state of the component
type State = { on :: Boolean }

-- | The query algebra for the component
data Query a
  = ToggleState a
  | GetState (Boolean -> a)

-- | The component definition
myComponent :: forall g. Component State Query g
myComponent = component { render, eval }
  where

  render :: State -> ComponentHTML Query
  render state =
    H.div_
      [ H.h1_
          [ H.text "Toggle Button" ]
      , H.button
          [ E.onClick (E.input_ ToggleState) ]
          [ H.text (if state.on then "On" else "Off") ]
      ]
```

DIPL. PHYS. PEER STRITZINGER GMBH

Getting started with Purescript

1. get it from npm or psvm
2. start reading Purescript by Example
3. read purescript-compat-elm
4. try out Pux or Flare
5. come on #purescript on freenode
6. come to the video meetup
7. try out Halogen
8. ???
9. PROFIT!

Purescript Conclusion

Powerful

Sensible

With all your favourite abstractions, and more

It will take time to learn, but similar enough to Elm to get a headstart

But you don’t have to know **everything** to start (with Pux)

It’s not obsessed about language UX, but it’s still good

DIPL. PHYS. PEER STRITZINGER GMBH

tl;dr

Elm works fine with Erlang
If Elm compiles, it works (mostly)
boilerplate can get annoying
never expect fancy types
Haskell syntax (with less squiggles)
Makes for a great entry level language into Haskellworld
unexpected removal of FRP was :/

tl;dr

Purescript works fine with Erlang
If Purescript compiles, it works (mostly)
types can get complicated
expect a longish learning curve
Haskell syntax, in all its squiggly glory
the roadmap is sensible
Makes a great second step into your road to Haskell
maybe use Pux to start with

www.stritzinger.com

@doppioslash

DIPL. PHYS. PEER STRITZINGER GMBH

Win One of 5 Boards by subscribing to the Newsletter during the conference until November 5th 2016

GRiSP

www.grisp.org

DIPL. PHYS. PEER STRITZINGER GMBH