Software Analyzers

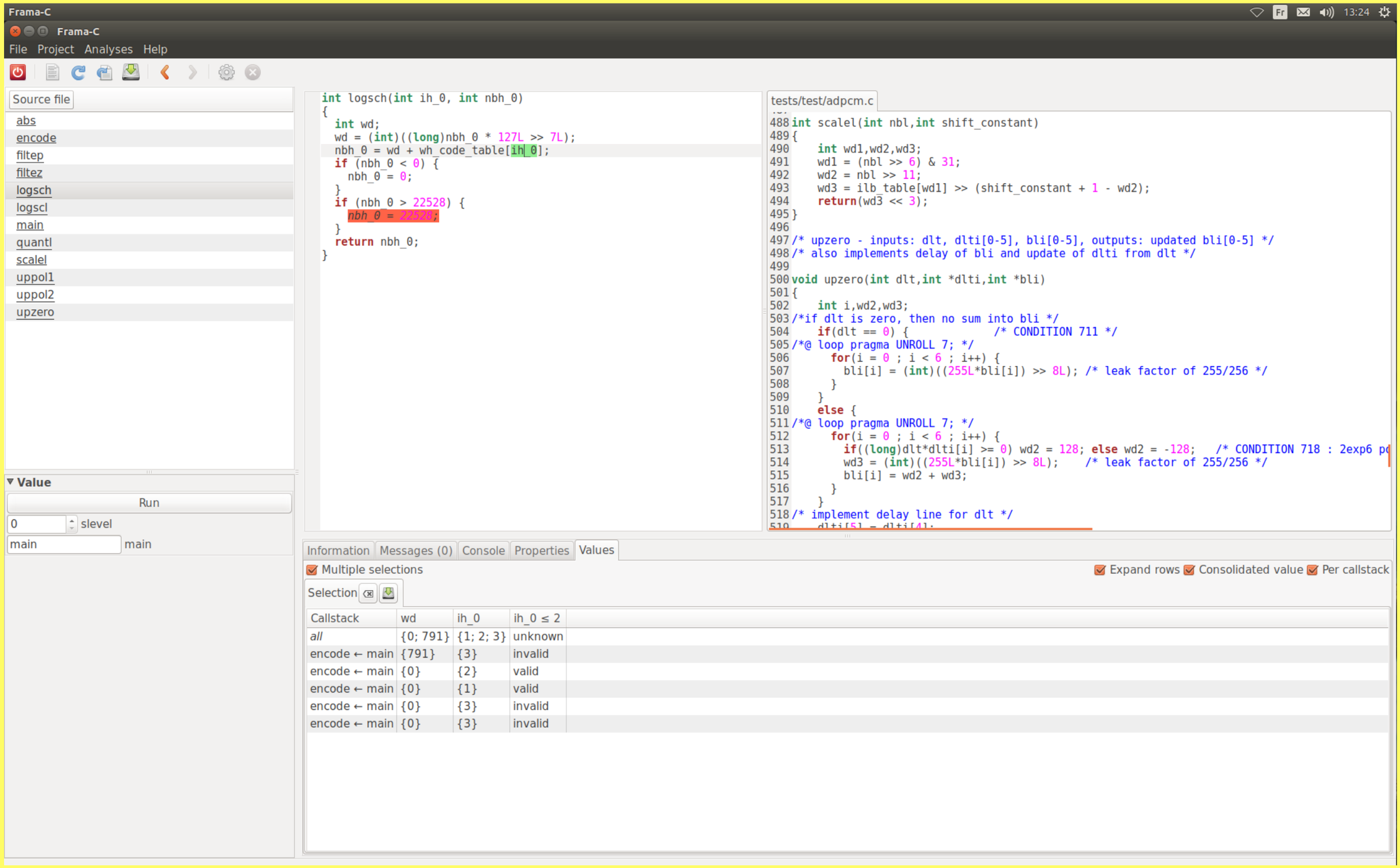
Ivette

An amazing new GUI for Frama-C

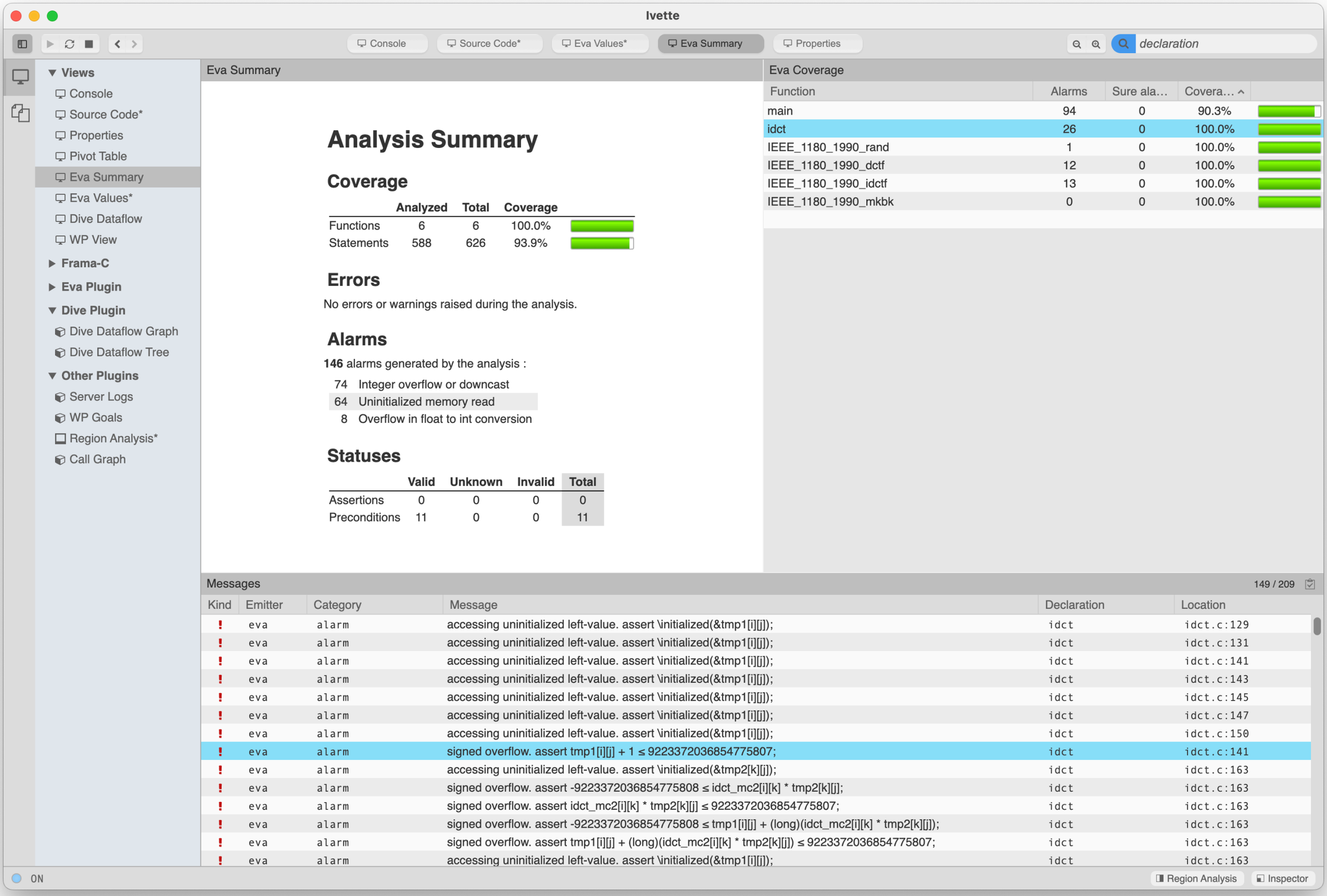
L. Correnson

CEA-List, Paris Saclay, Software Safety & Security Lab

Our good old GTK frama-c-gui …



… and the brand new Ivette !



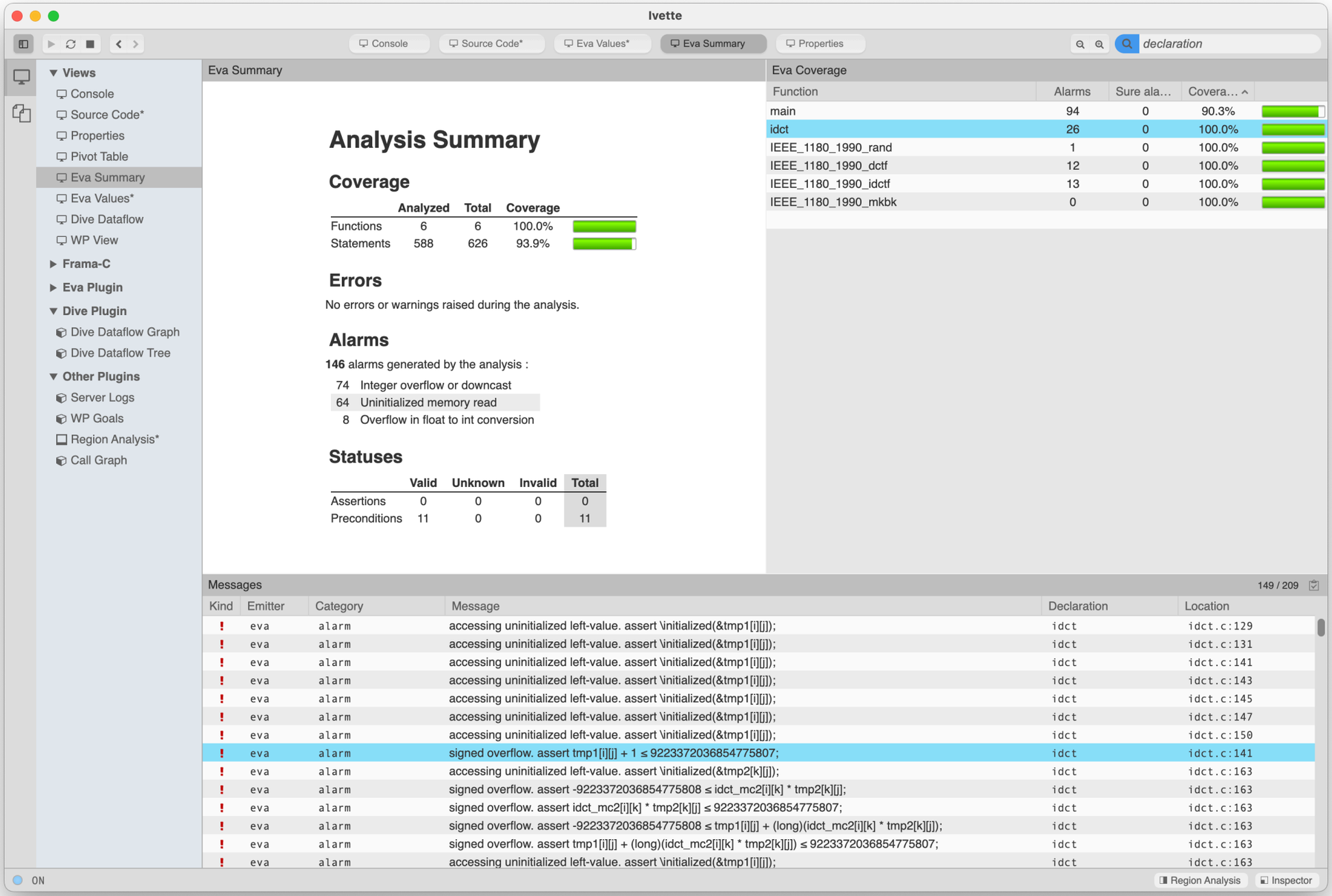
Software Analyzers

What’s New ?

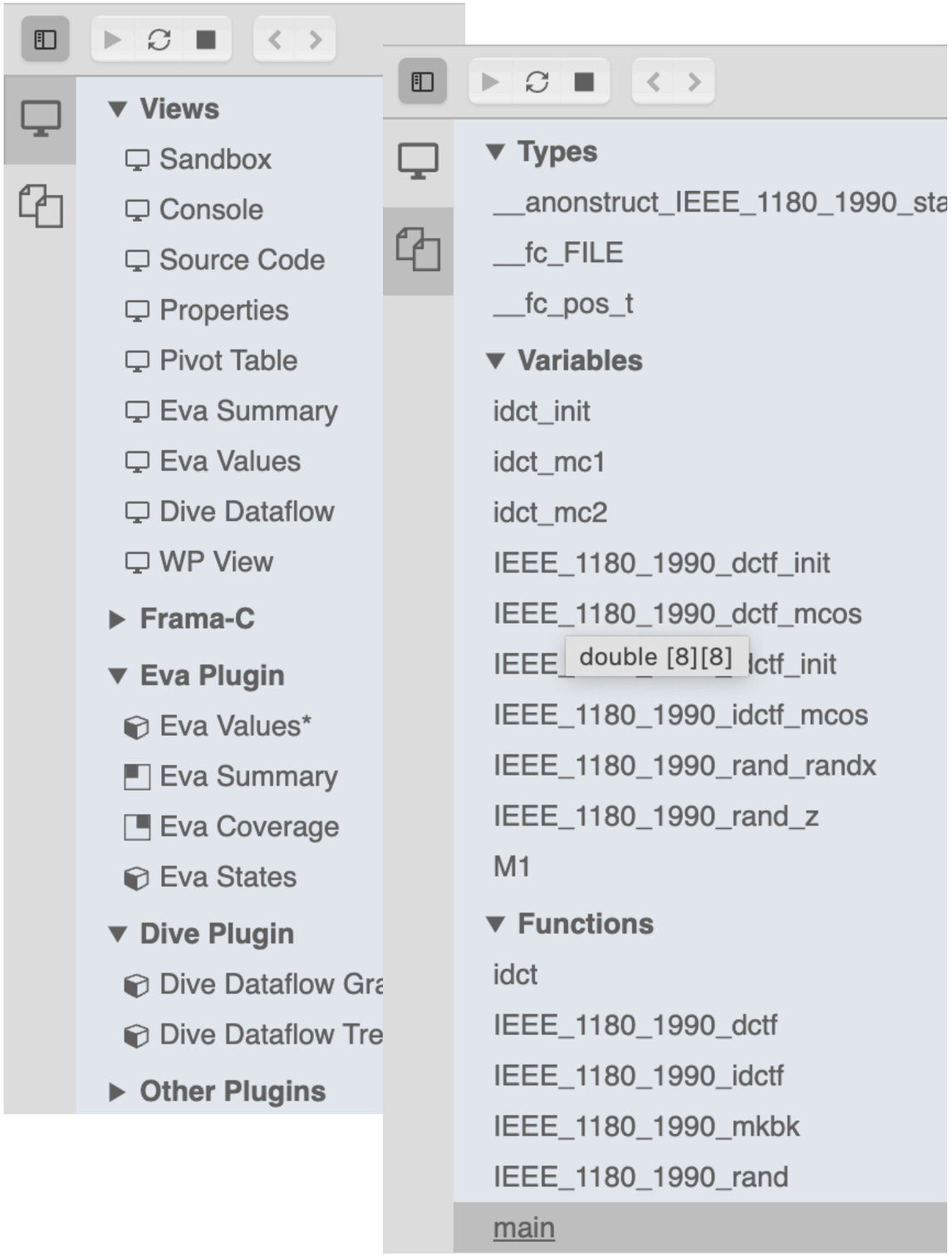
Organization

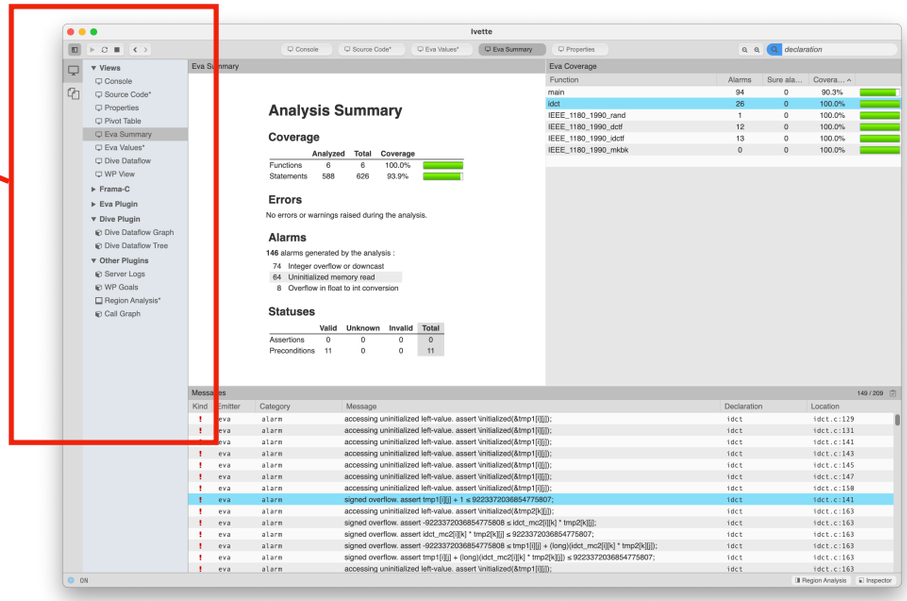
frama C
Software Analyzers

Organization

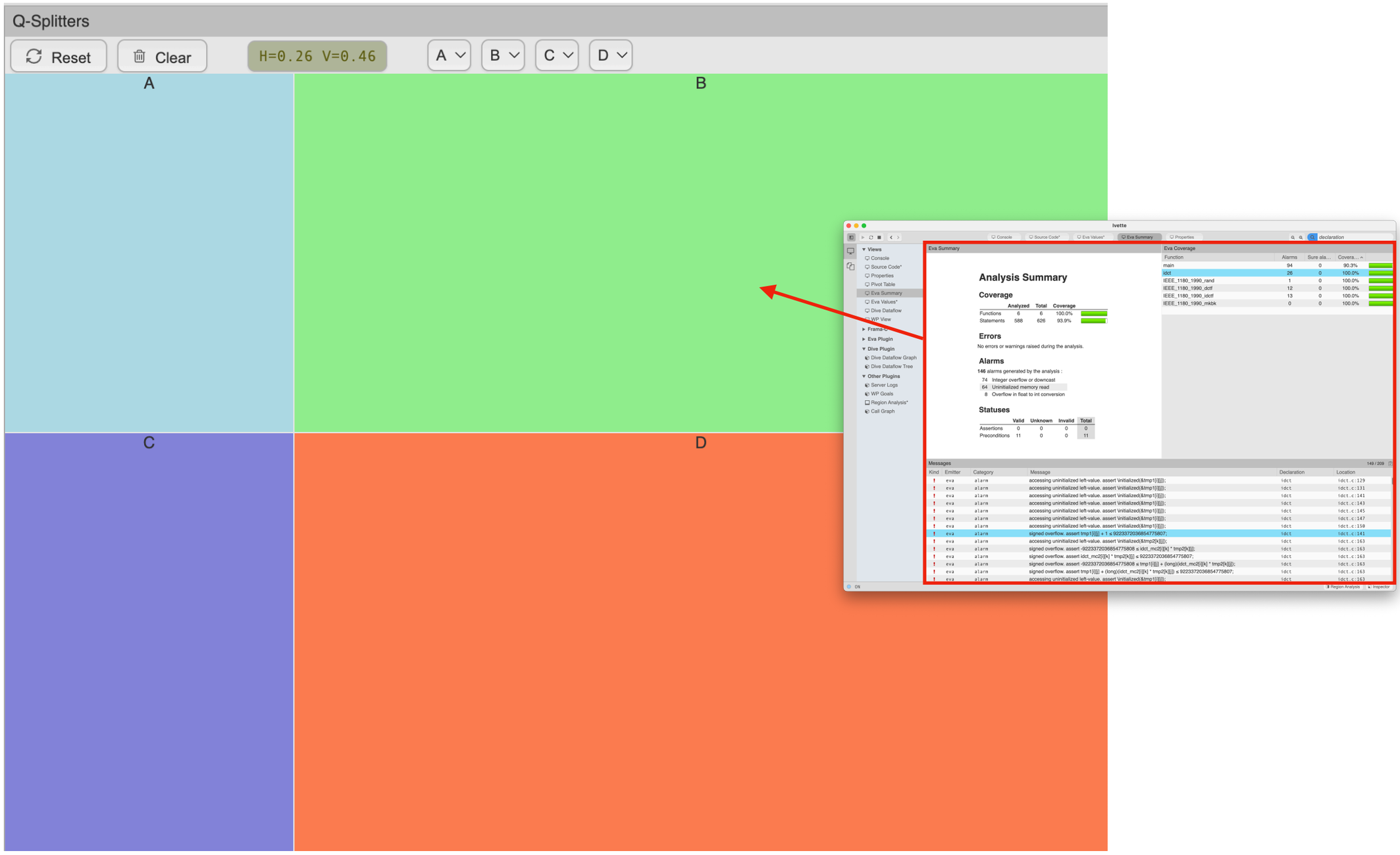


Controls — Navigation — Sidebar(s)

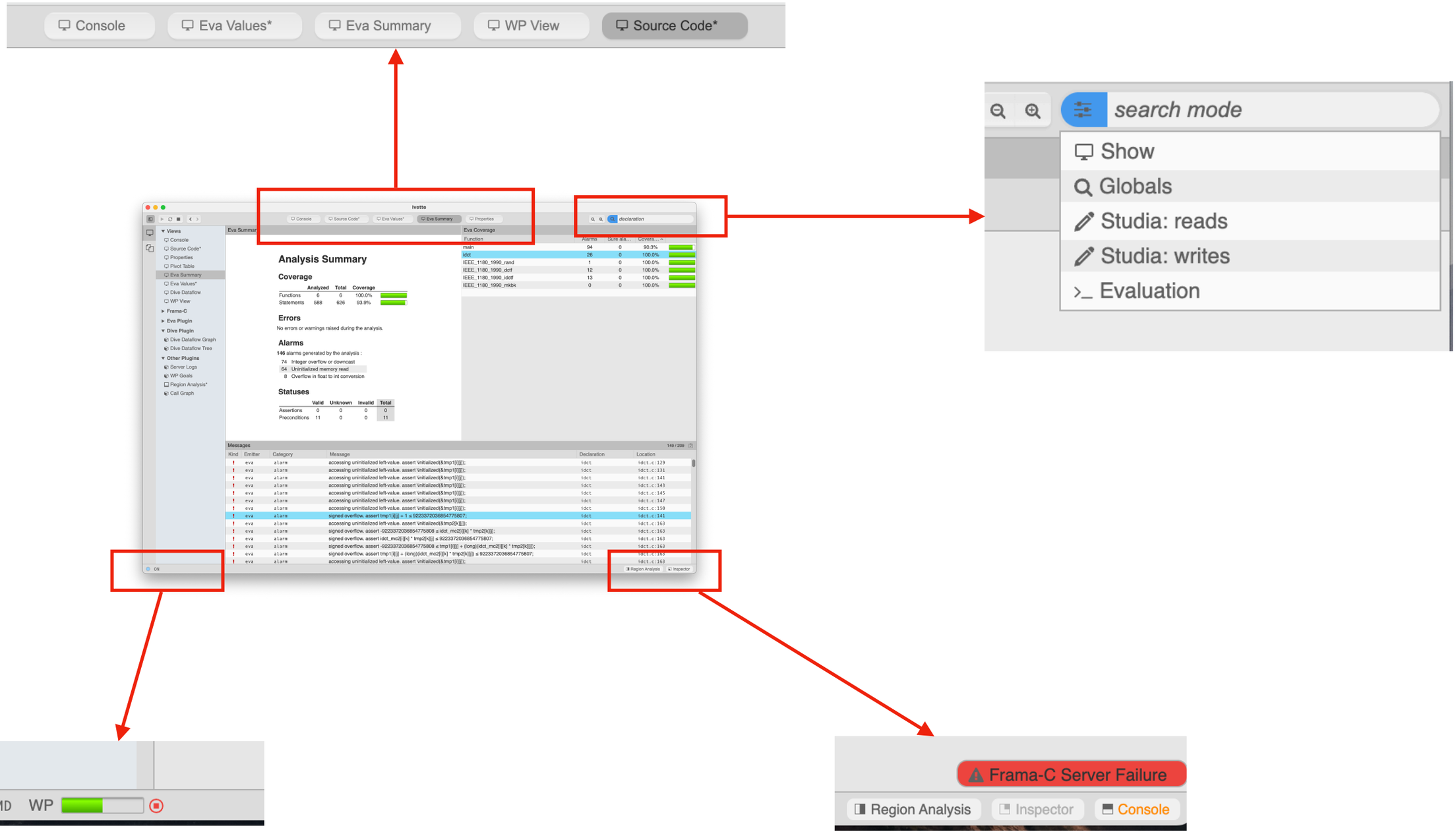




4-Quarters Central Workspace

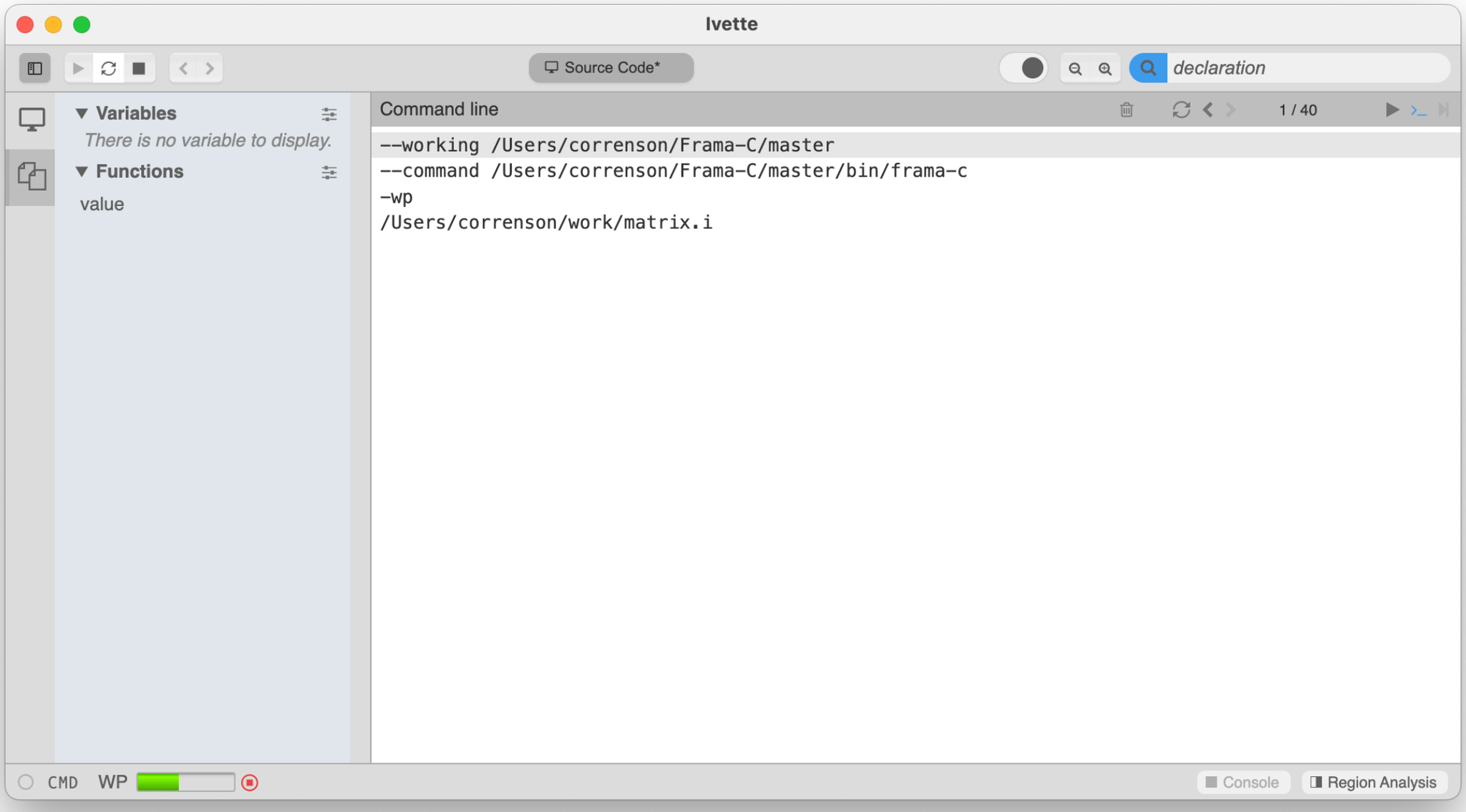


View Bar — Search Bar — Status Bar — Dock



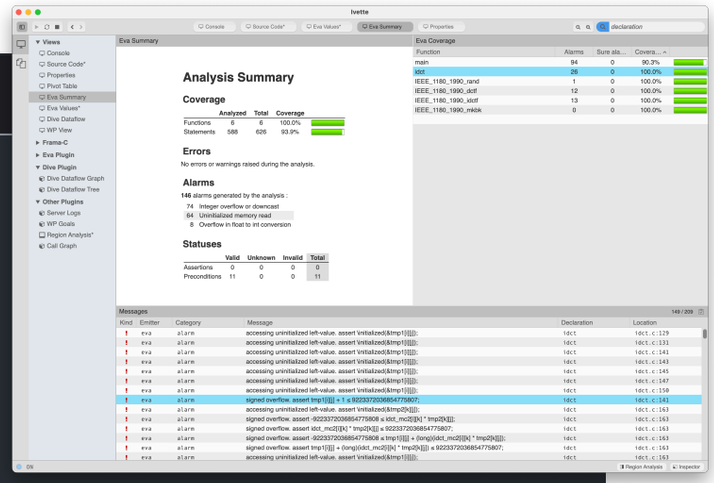
CLI + UI Console

```
[ ~/work ]
$ ivette -wp matrix.i
```



frama C
Software Analyzers

Extensible ! Everywhere !



Ivette Documentation - v29.0.0

- Ivette Documentation - v29.0.0
- dome
- frama-c
- ivette
  - ComponentProps
  - ContentProps
  - Hint
  - ItemProps
  - SearchProps
  - SidebarProps
  - TitleBarProps
  - ToolProps
  - ViewLayoutProps
  - Layout
  - Layout1
  - Layout2
  - Layout3
  - Layout4
  - LayoutPosition
  - compId
  - TitleBar
  - focusSearchMode
  - registerComponent
  - registerGroup
  - registerSandbox
  - registerSearchMode
  - registerSidebar

Ivette Documentation / ivette /

Module ivette

INDEX

Interfaces

- ComponentProps
- ContentProps
- Hint
- ItemProps
- SearchProps
- SidebarProps
- TitleBarProps
- ToolProps
- ViewLayoutProps

Type Aliases

- Layout
- Layout1
- Layout2
- Layout3
- Layout4
- LayoutPosition
- compId

Functions

- TitleBar
- focusSearchMode
- registerComponent
- registerGroup
- registerSandbox
- registerSearchMode
- registerSidebar
- registerStatusbar
- registerToolbar
- removeSearchMode
- selectSearchMode
- updateSearchMode
- useSearchMode

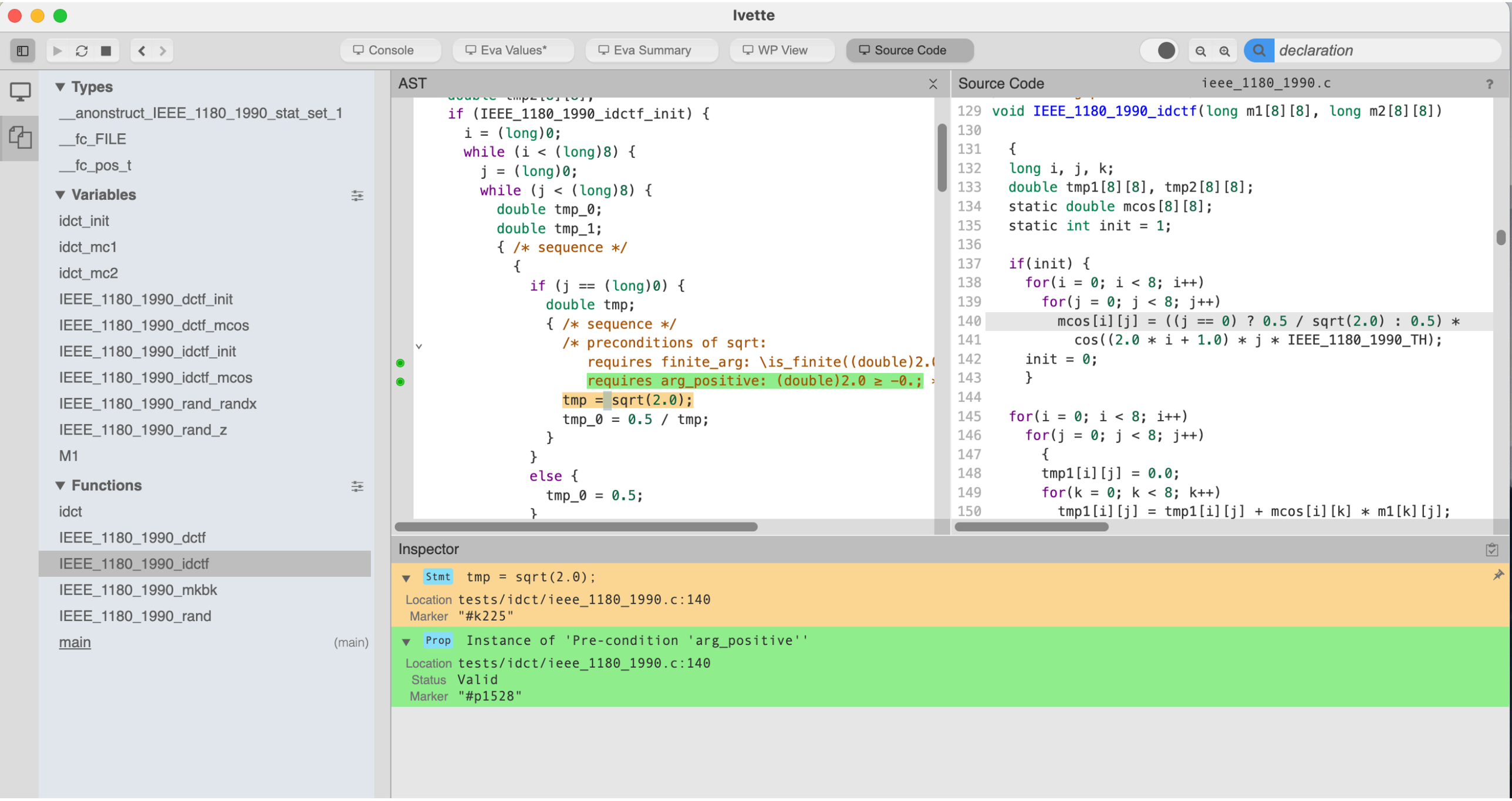
Software Analyzers

What’s In ?

« Old » components… but smarter !

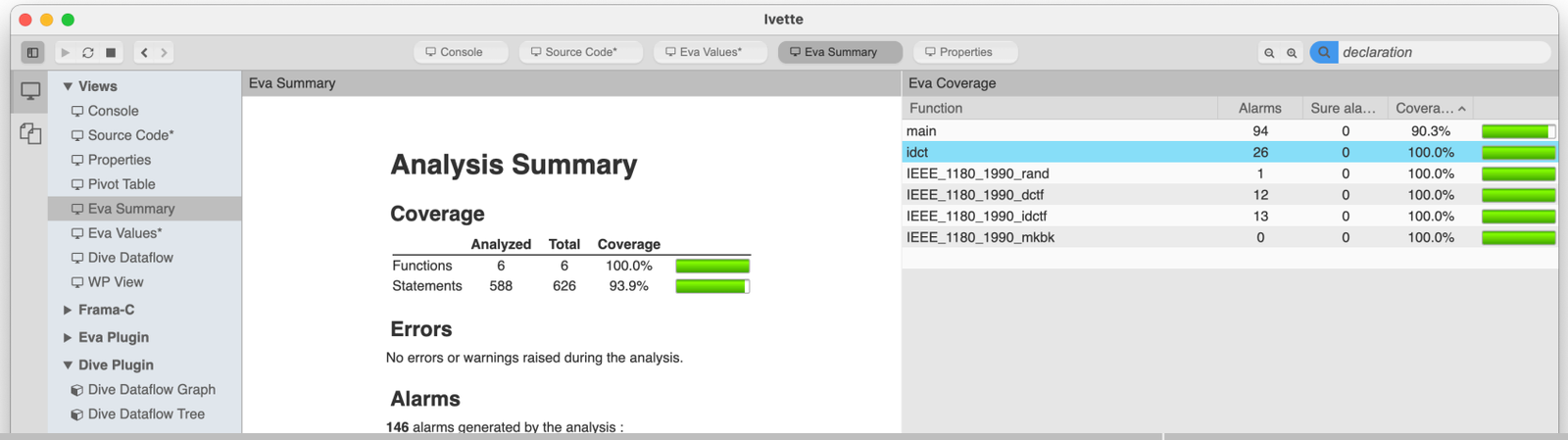
And new ones !

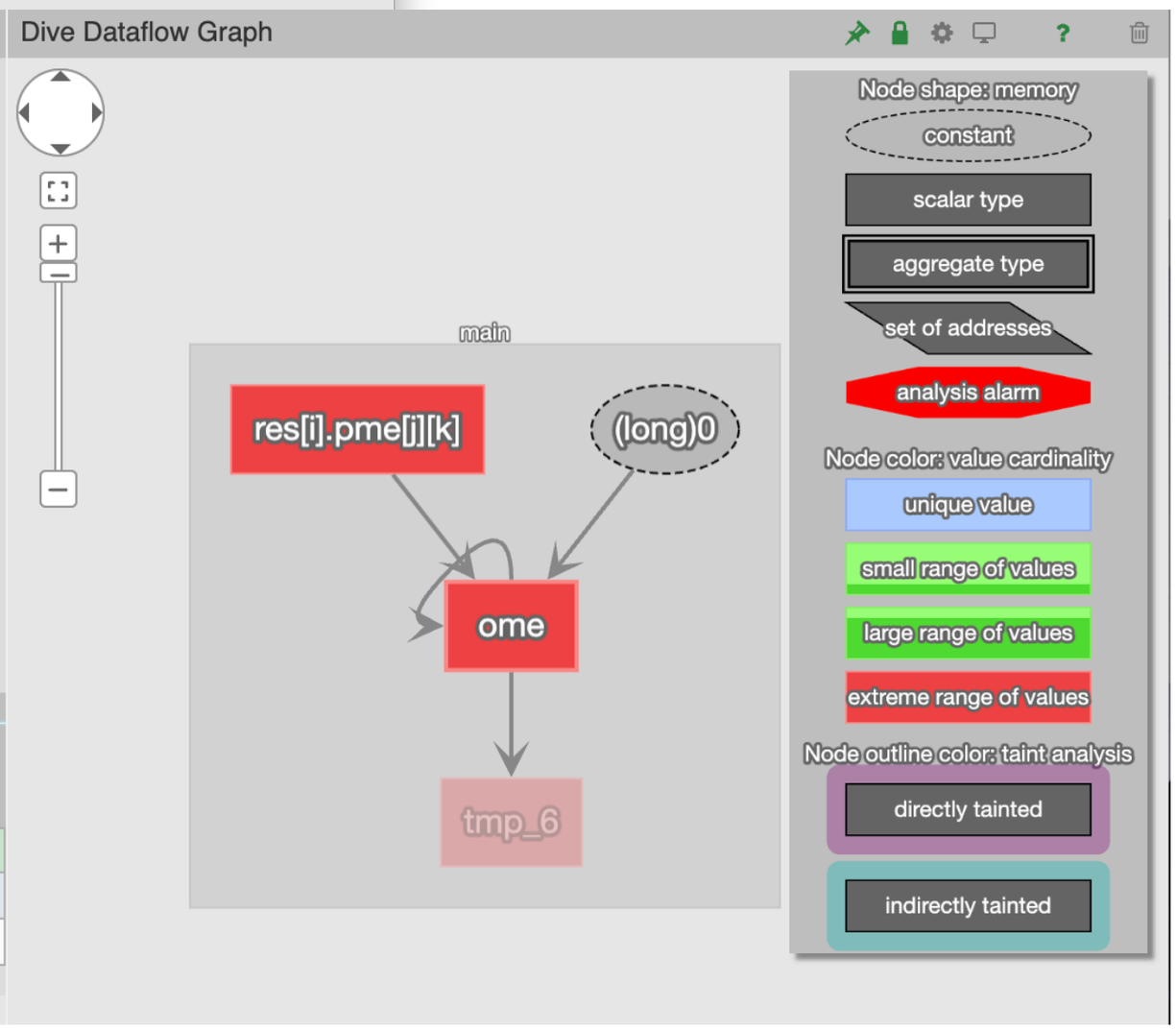
Reactive Navigation



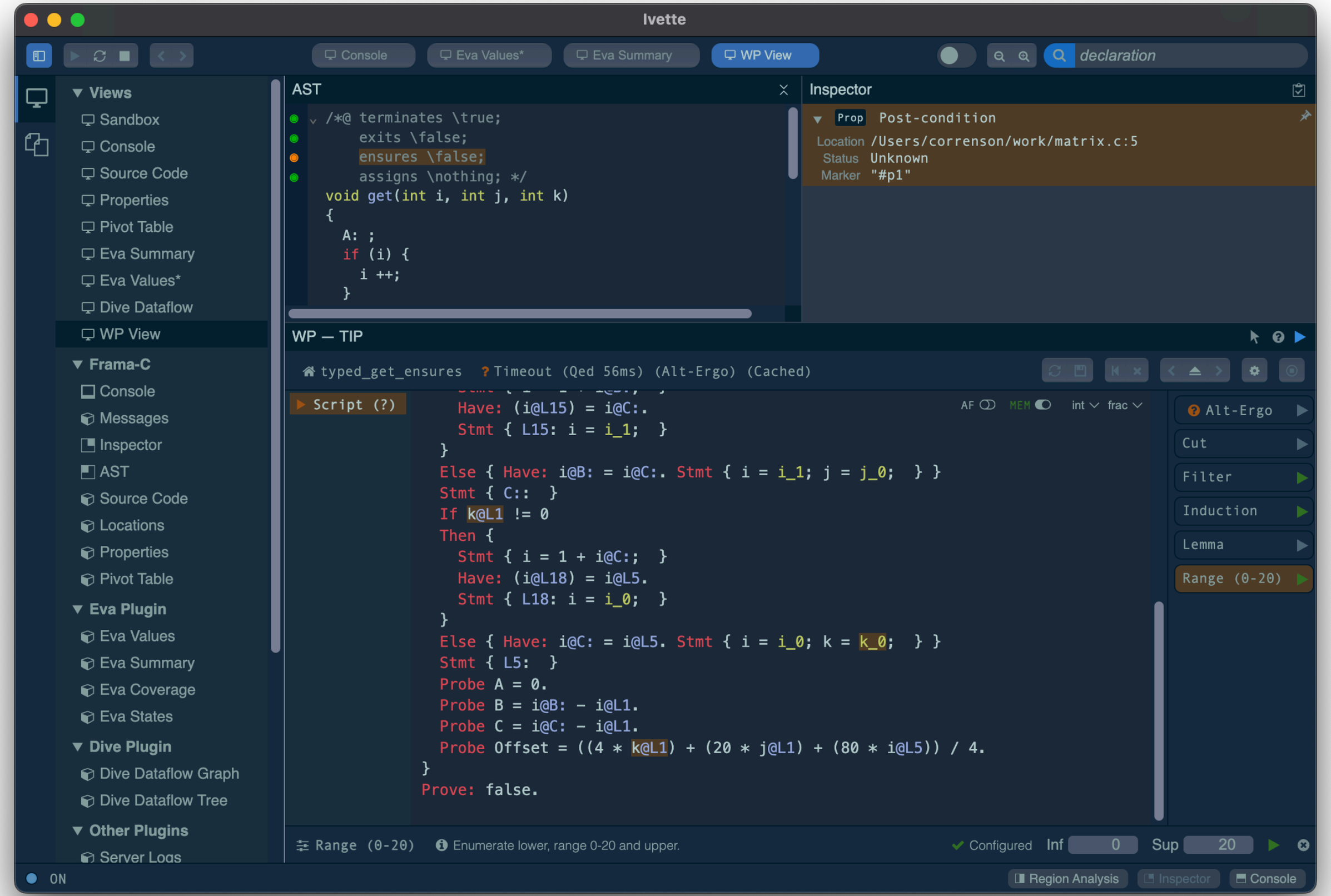
Software Analyzers

Eva — Studia — Dive



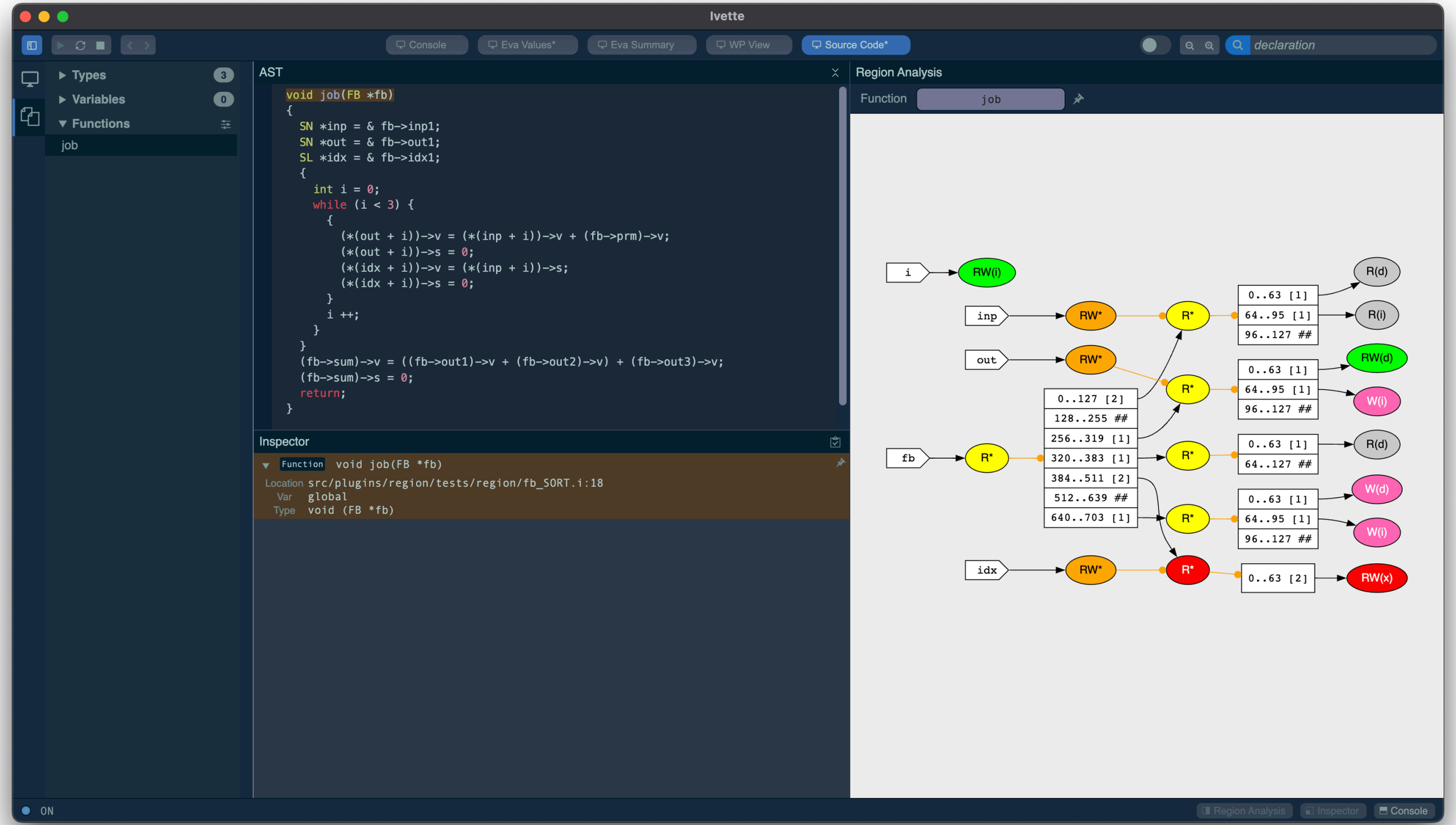


WP & Proof Transformer (TIP)



frama C
Software Analyzers

New Analysis & New Visualizations



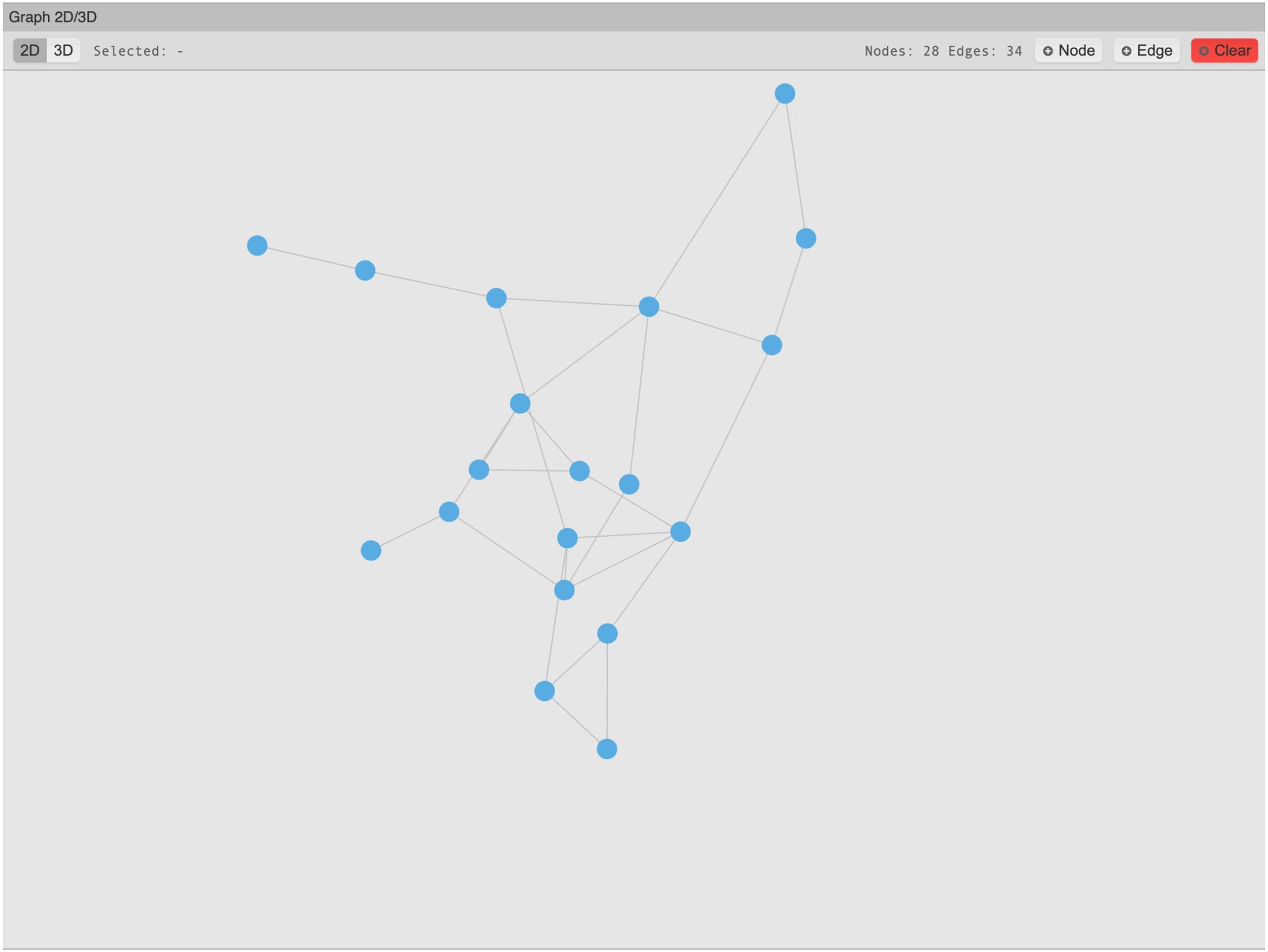
Software Analyzers

What’s Available ?

Rich Toolkit…

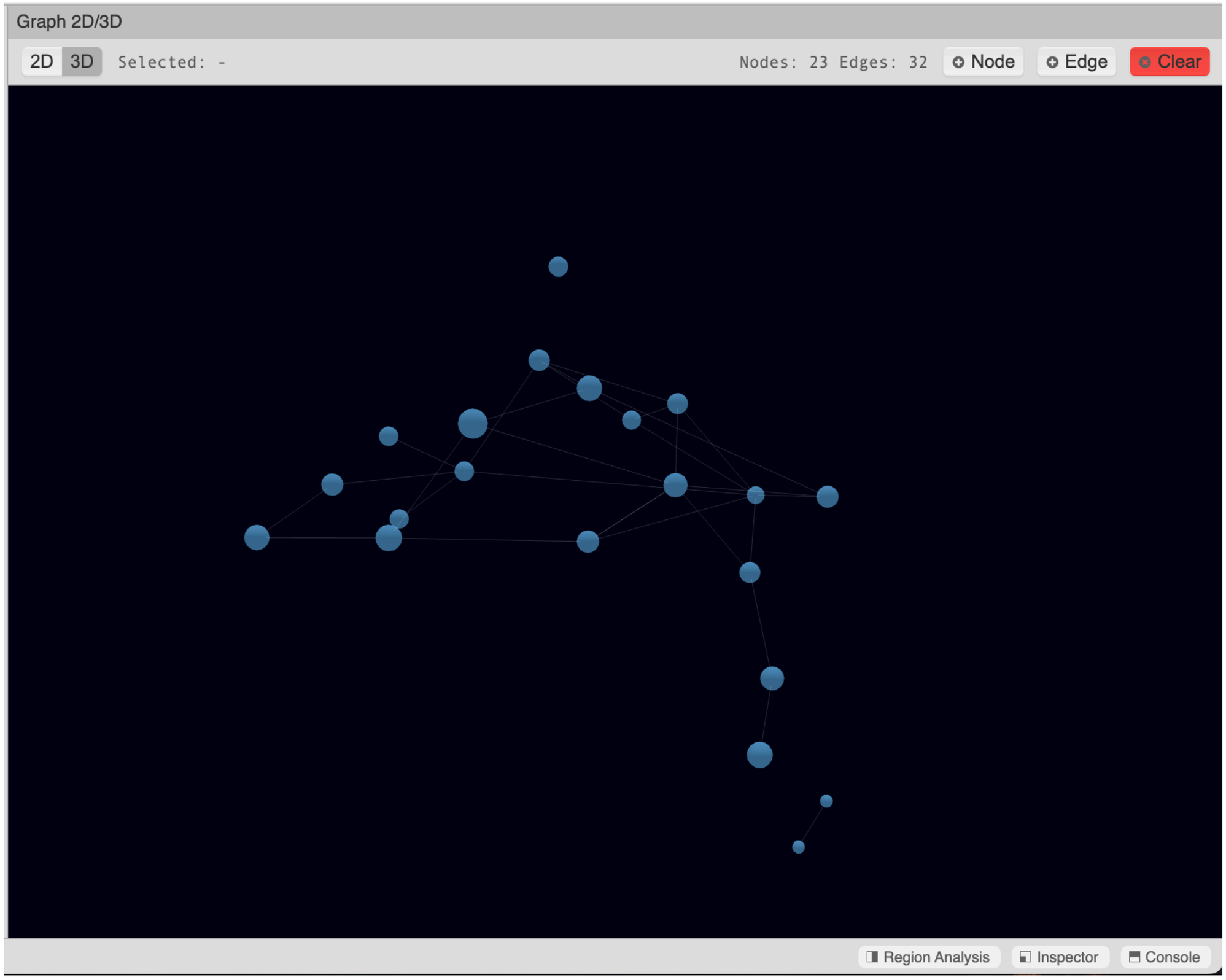
frama C
Software Analyzers

2D Graphs



frama C
Software Analyzers

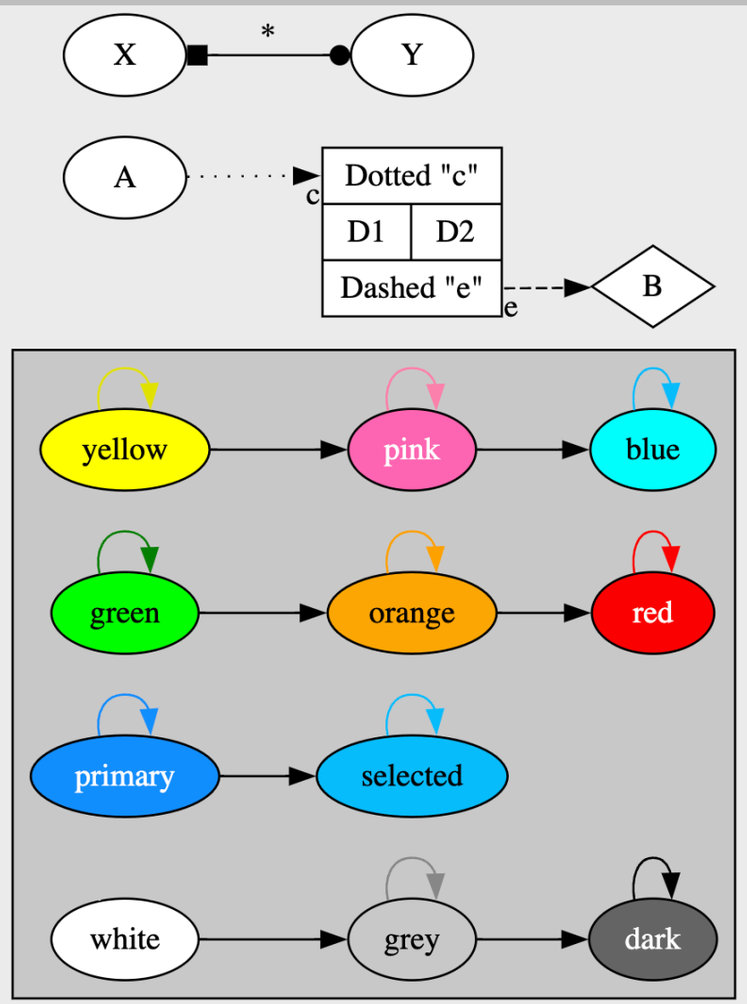
3D Graphs



DOT Diagrams

Diagram

```
Selected:
digraph {
  rankdir="LR"; bgcolor="none"; width=0.5;node [ style="filled"
  subgraph cluster_n0 {
    style="filled"; tooltip="Background Cluster"; fontcolor="bla
    n1; n2; n3; n4; n5; n6; n7; n8; n9; n10; n11;
  }
  n12 [ id="A"; label="A"; tooltip="A"; fontcolor="black"; fillc
  n13 [ id="B"; label="B"; shape="diamond"; tooltip="B"; fontcol
  n14 [ id="R"; shape="record"; label="<n15> Dotted \"c\"|{D1|D2
  n1 [ id="white"; label="white"; tooltip="white"; fontcolor="bl
  n2 [ id="grey"; label="grey"; tooltip="grey"; fontcolor="black
  n3 [ id="dark"; label="dark"; tooltip="dark"; fontcolor="white
  n4 [ id="primary"; label="primary"; tooltip="primary"; fontcol
  n5 [ id="selected"; label="selected"; tooltip="selected"; font
  n6 [ id="green"; label="green"; tooltip="green"; fontcolor="bl
  n7 [ id="orange"; label="orange"; tooltip="orange"; fontcolor=
  n8 [ id="red"; label="red"; tooltip="red"; fontcolor="white";
  n9 [ id="yellow"; label="yellow"; tooltip="yellow"; fontcolor=
  n10 [ id="blue"; label="blue"; tooltip="blue"; fontcolor="blac
  n11 [ id="pink"; label="pink"; tooltip="pink"; fontcolor="whit
  n17 [ id="X"; label="X"; tooltip="X"; fontcolor="black"; fillc
  n18 [ id="Y"; label="Y"; tooltip="Y"; fontcolor="black"; fillc
  n12 -> n14:n15 [ headlabel="c"; headtooltip="A -> R"; tooltip=
  n14:n16 -> n13 [ taillabel="e"; tailtooltip="R -> B"; tooltip=
  n4 -> n5 [ tooltip="primary -> selected"; arrowtail="none";];
  n1 -> n2 [ tooltip="white -> grey"; arrowtail="none";];
  n2 -> n3 [ tooltip="grey -> dark"; arrowtail="none";];
  n6 -> n7 [ tooltip="green -> orange"; arrowtail="none";];
  n7 -> n8 [ tooltip="orange -> red"; arrowtail="none";];
  n9 -> n11 [ tooltip="yellow -> pink"; arrowtail="none";];
  n11 -> n10 [ tooltip="pink -> blue"; arrowtail="none";];
  n1 -> n1 [ tooltip="white -> white"; color="#ccc"; arrowtail="
  n2 -> n2 [ tooltip="grey -> grey"; color="#888"; arrowtail="no
  n3 -> n3 [ tooltip="dark -> dark"; color="black"; arrowtail="r
  n4 -> n4 [ tooltip="primary -> primary"; color="dodgerblue"; a
  n5 -> n5 [ tooltip="selected -> selected"; color="deepskyblue"
  n6 -> n6 [ tooltip="green -> green"; color="green"; arrowtail=
  n7 -> n7 [ tooltip="orange -> orange"; color="orange"; arrowta
  n8 -> n8 [ tooltip="red -> red"; color="red"; arrowtail="none"
  n9 -> n9 [ tooltip="yellow -> yellow"; color="#e5e100"; arrowt
  n10 -> n10 [ tooltip="blue -> blue"; color="deepskyblue"; arro
  n11 -> n11 [ tooltip="pink -> pink"; color="palevioletred1"; a
  n17 -> n18 [ label="*"; labeltooltip="box to dot"; tooltip="bo
}
```

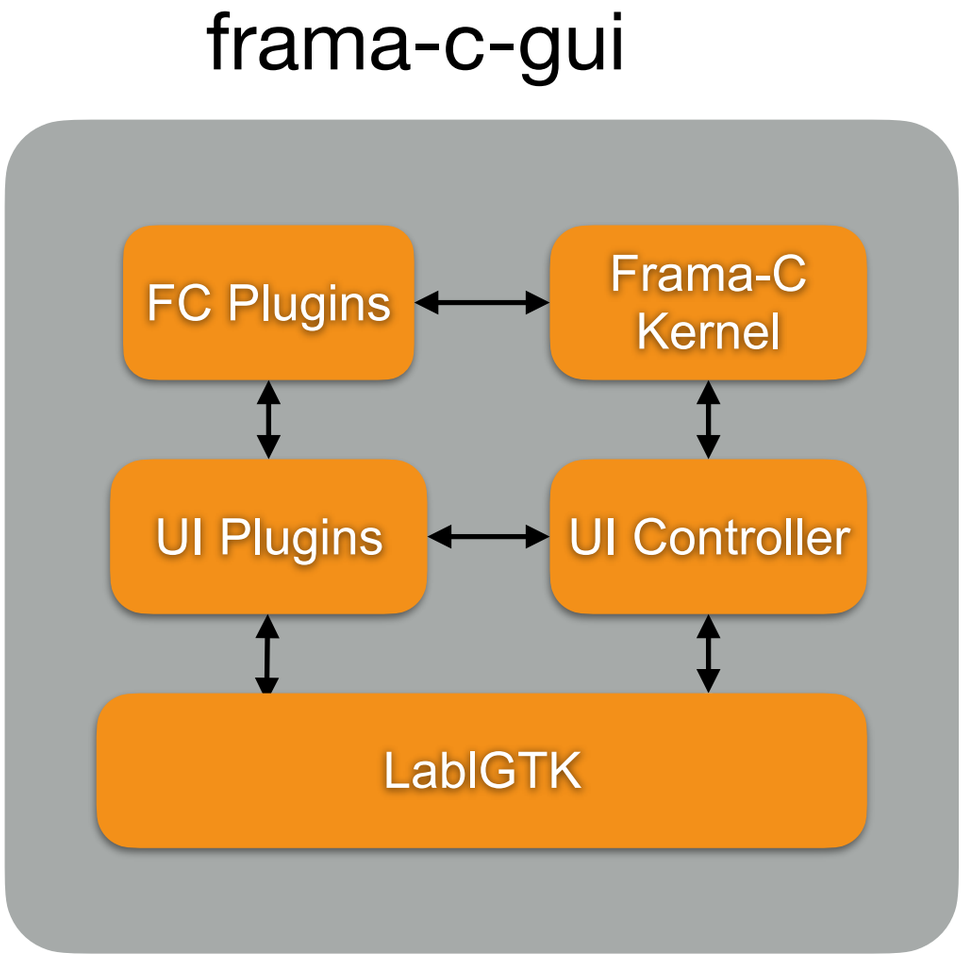


Software Analyzers

What’s Inside ?

Not just a relooking…

Frama-C Old GUI's Design



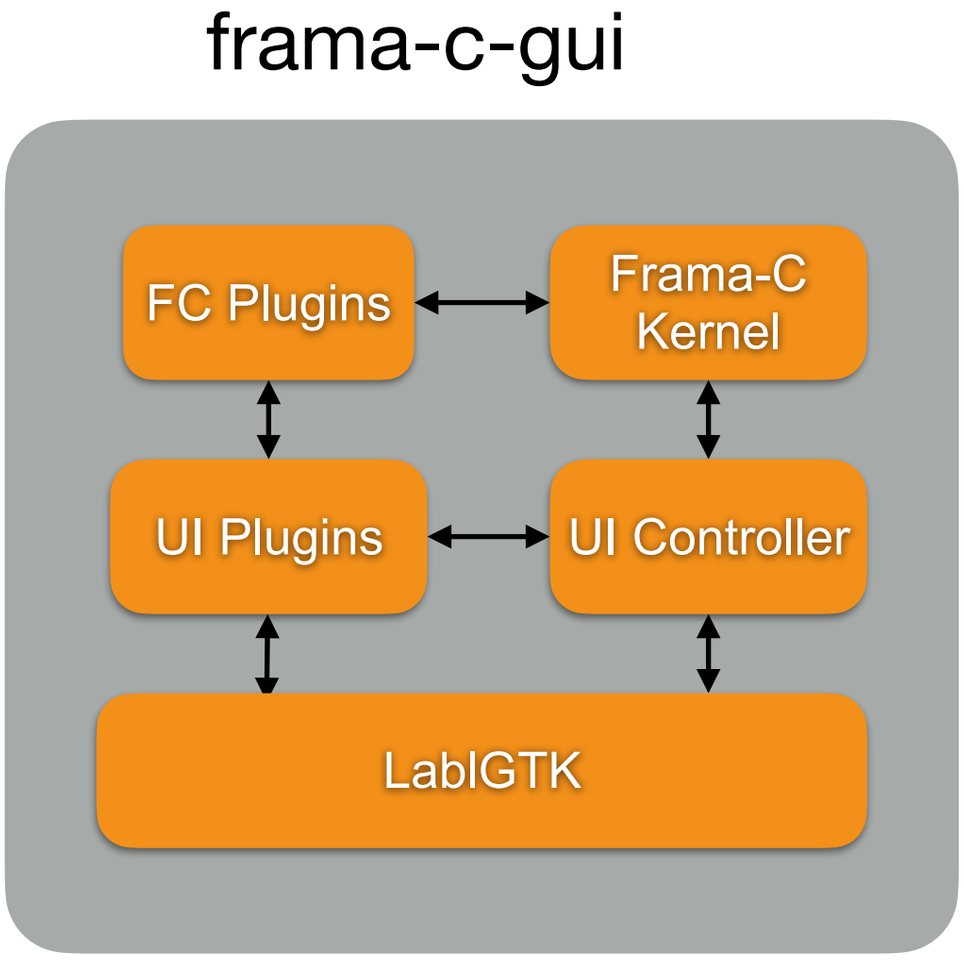
OCaml

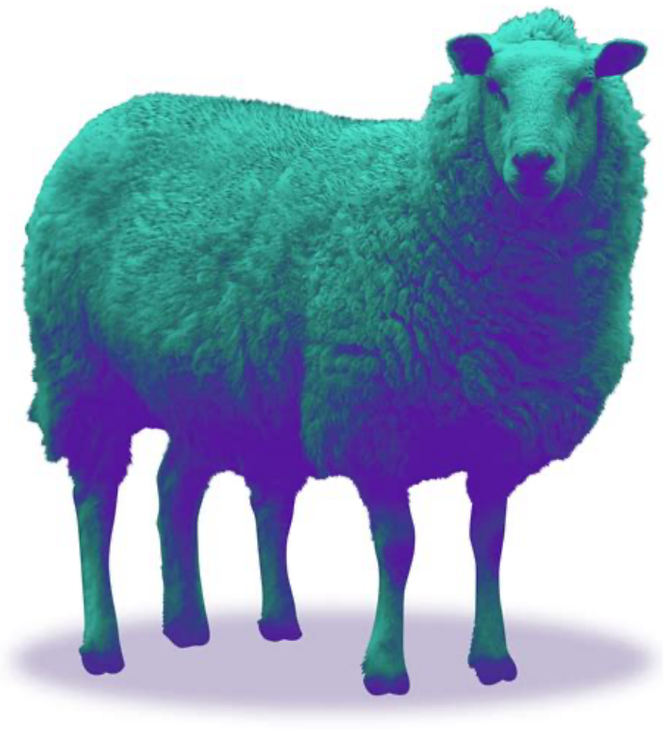
frama C Software Analyzers

GTK

UX UI

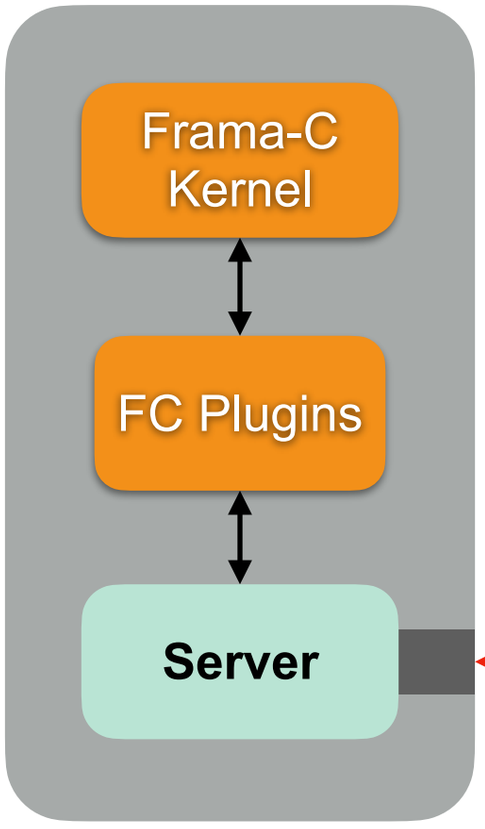
Looking for the Rare Birds …



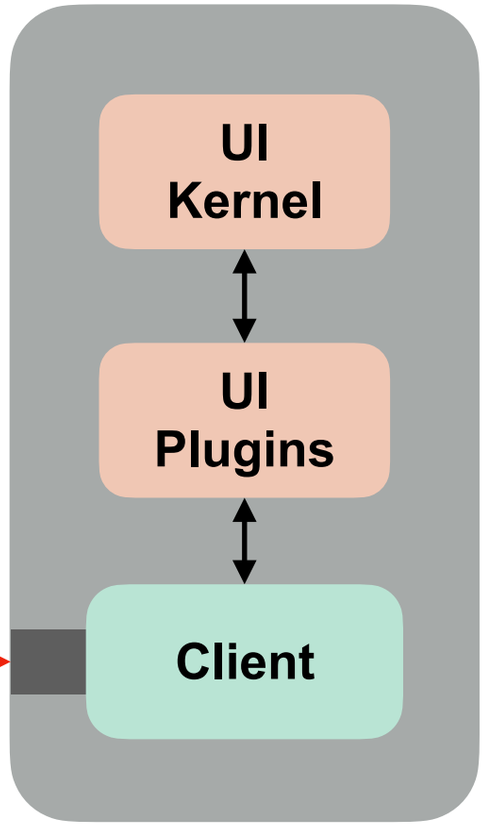


Redesign — Splitting

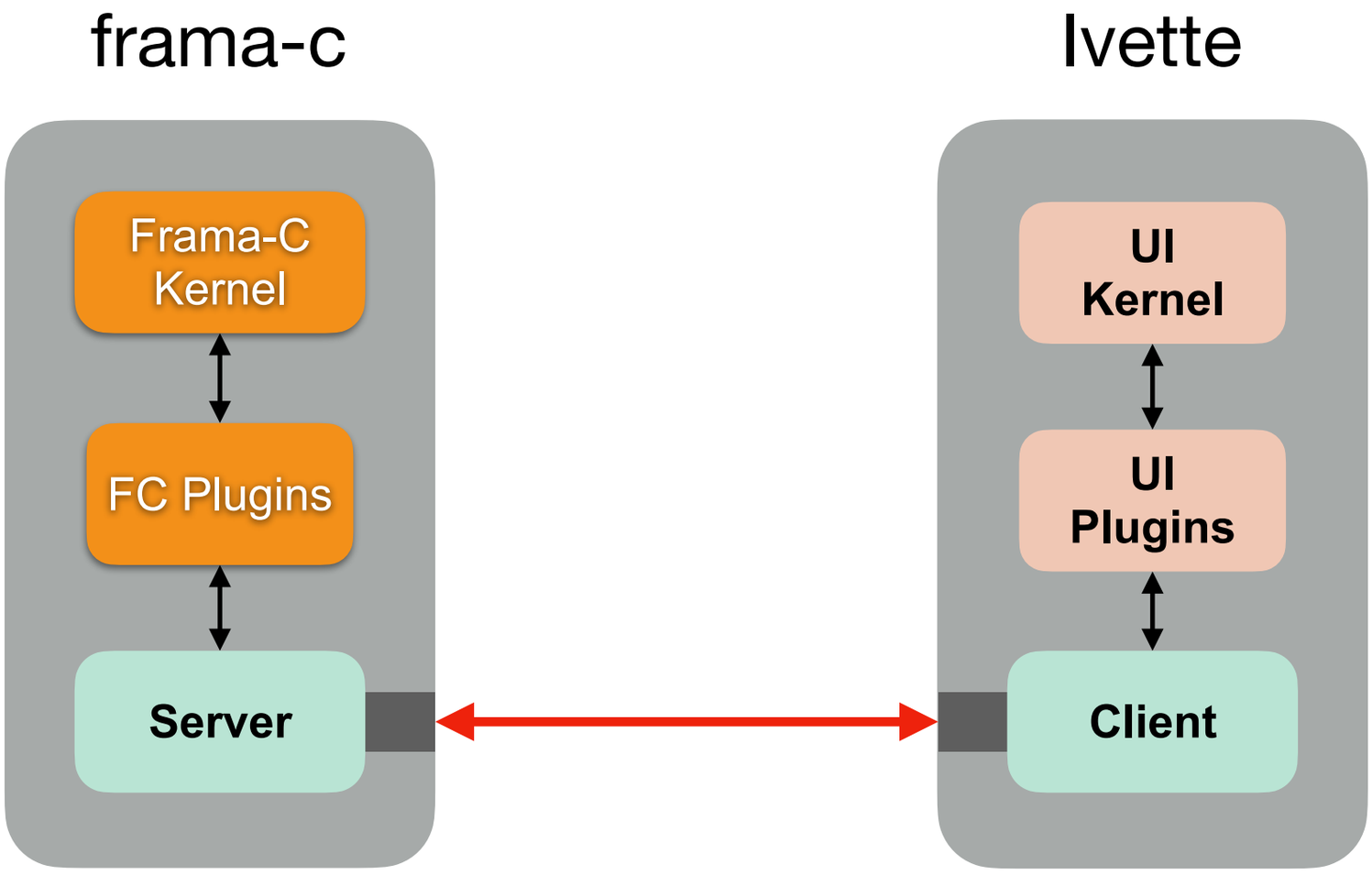
frama-c



Ivette



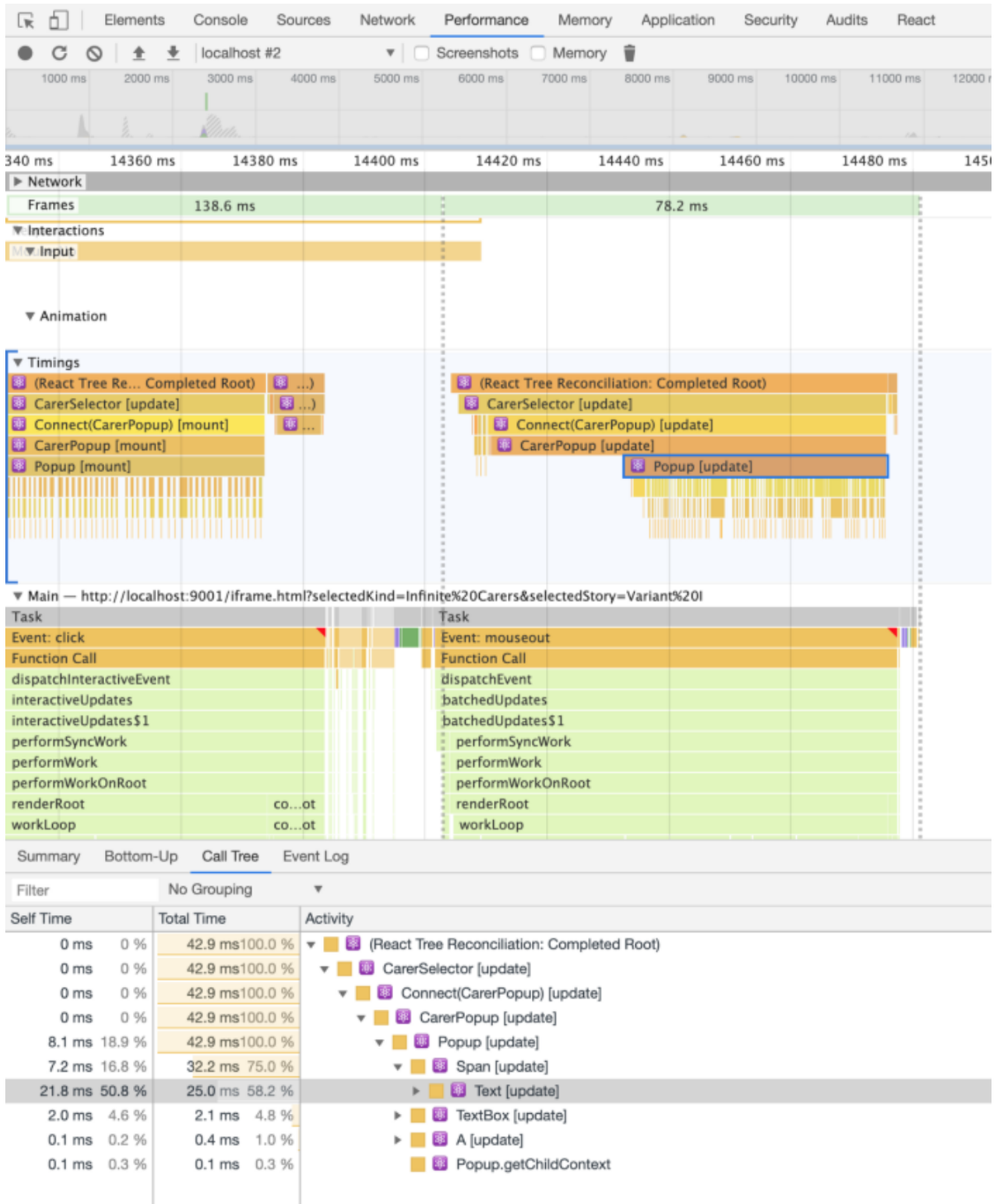
Different Paradigms



- ★ Command Driven (batch)
- ★ Synchronous
- ★ Intensive

- ★ Interactive
- ★ Asynchronous (threads)
- ★ Slow & Sparse

Time Lines



< 1 ms

few DOM Sync

~10 ms

sorting an array of
100,000 elements

~100 ms

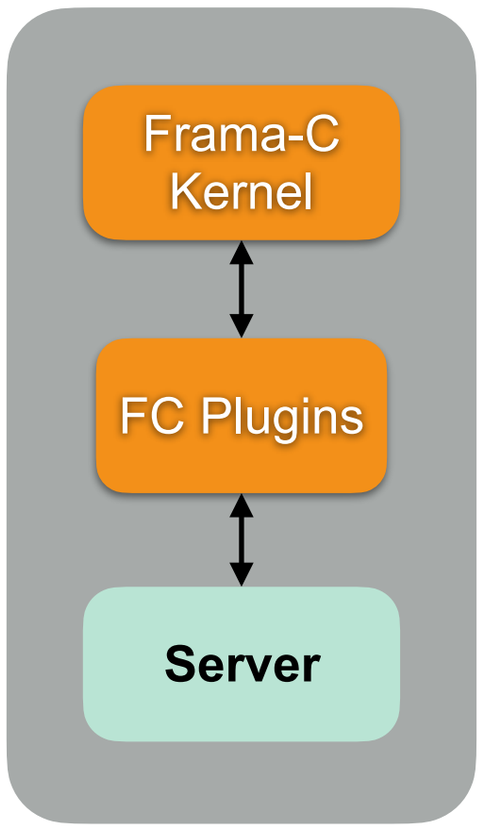
running SMT solver
on a Frama-C/WP
proof obligation

~300 ms

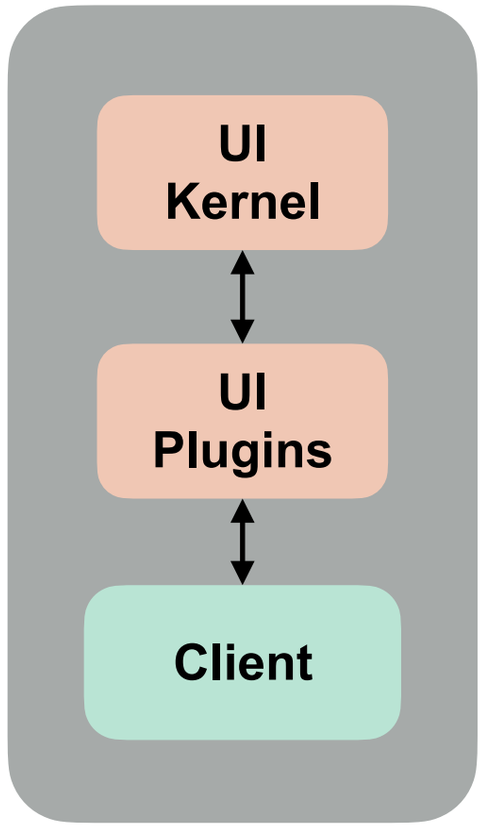
human expects
UI feedback
after a mouse click

Different Paradigms

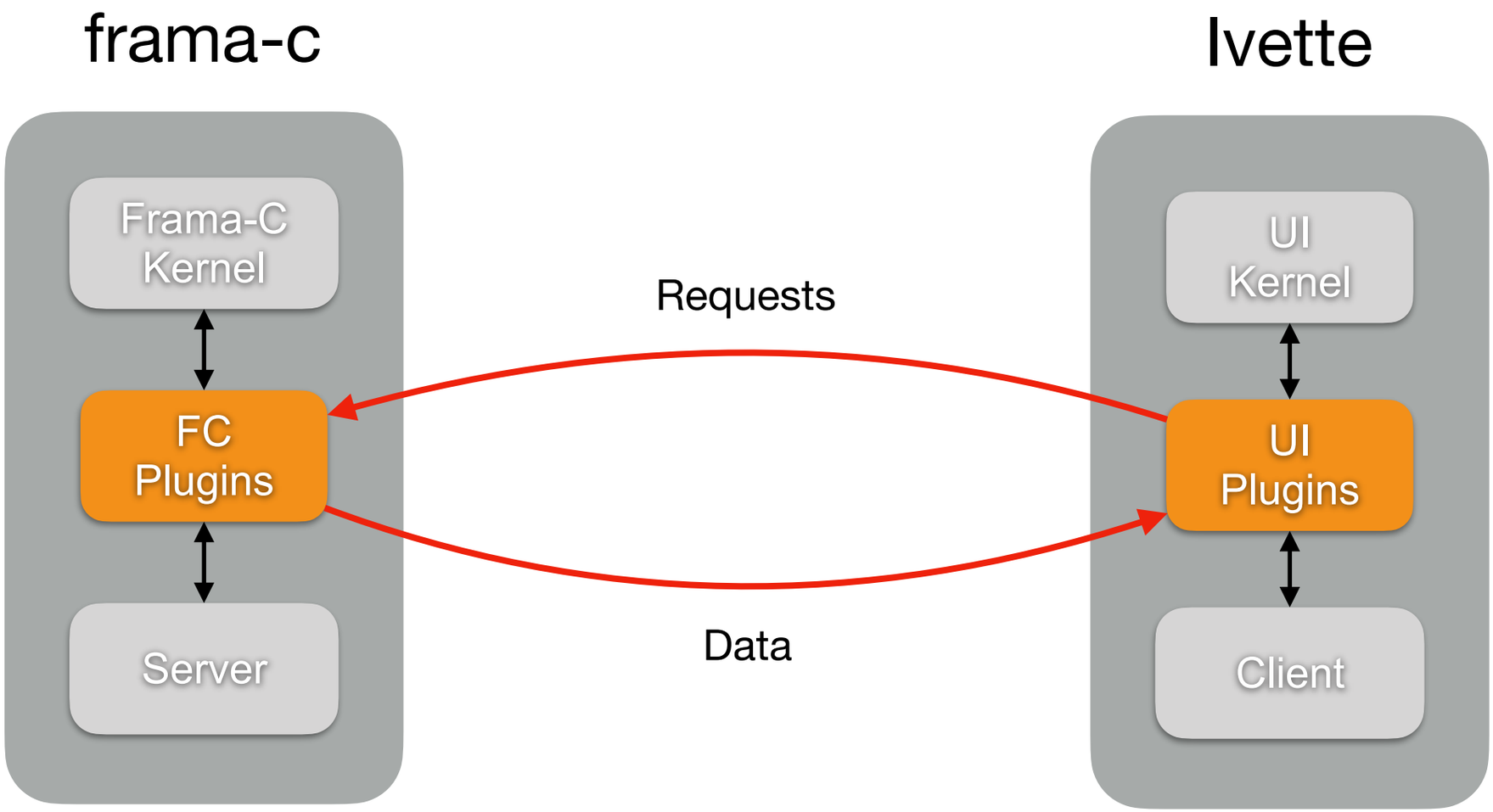
frama-c



Ivette

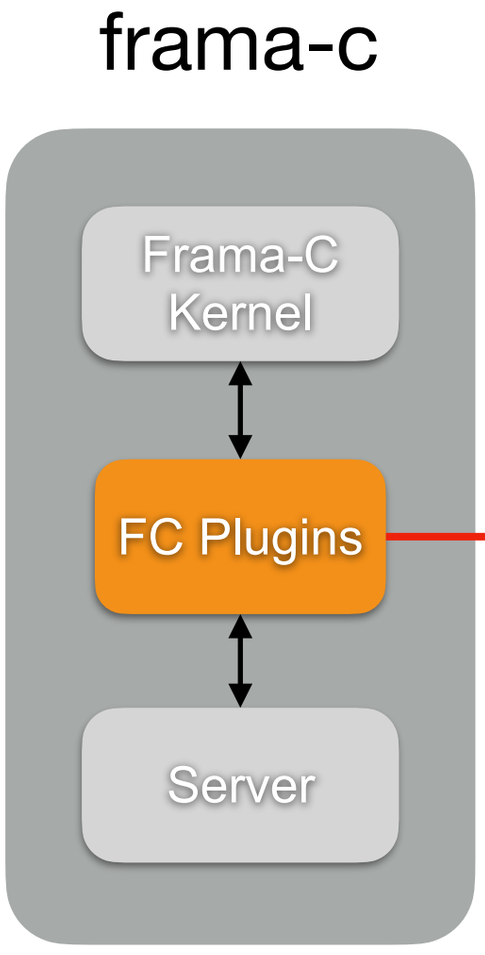


Mirroring



- ★ Declarative
- ★ Incremental
- ★ Cached
- ★ Sync / Async

Writing Plugins — Server Side



262 loc

```
let () =
  Request.register
    ~package ~kind:`GET ~name:"regions"
    ~descr:(Md.plain "Returns computed regions for the given declaration")
    ~input:(module Kernel_ast.Decl)
    ~output:(module Regions)
    ~signals:[signal]
    begin fun decl ->
      try Memory.regions @@ map_of_declaration decl
      with Not_found -> []
    end
```

★ Command Driven (batch)
★ Synchronous
★ Intensive

★ JSON Data Serialization
★ Registered Data
★ Registered Requests
★ Idiomatic OCaml Code
★ Auto-documented

(Auto) Documented API

Frama-C Server

Plugin Wp

WP Main Services

WP Interactive Prover
WP Tactics

plugins.wp.goalsData (DATA)

Data for array rows `goals`

`goalsData ::= { fields... }`

Field	Format	Description
`"wpo"`	`goal`	Entry identifier.
`"marker"`	`marker`	Associated Marker
`"scope"` (opt.)	`decl`	Associated declaration, if any
`"property"`	`marker`	Property Marker
`"fct"` (opt.)	*string*	Associated function name, if any
`"bhv"` (opt.)	*string*	Associated behavior name, if any
`"thy"` (opt.)	*string*	Associated axiomatic name, if any
`"name"`	*string*	Informal Property Name
`"smoke"`	*boolean*	Smoking (or not) goal
`"passed"`	*boolean*	Valid or Passed goal
`"status"`	`status`	Verdict, Status
`"stats"`	`stats`	Prover Stats Summary
`"proof"`	*boolean*	Proof Tree
`"script"` (opt.)	*string*	Script File
`"saved"`	*boolean*	Saved Script

plugins.wp.fetchGoals (GET)

Data fetcher for array `goals`

input `::=` *number*

output `::= { output... }`

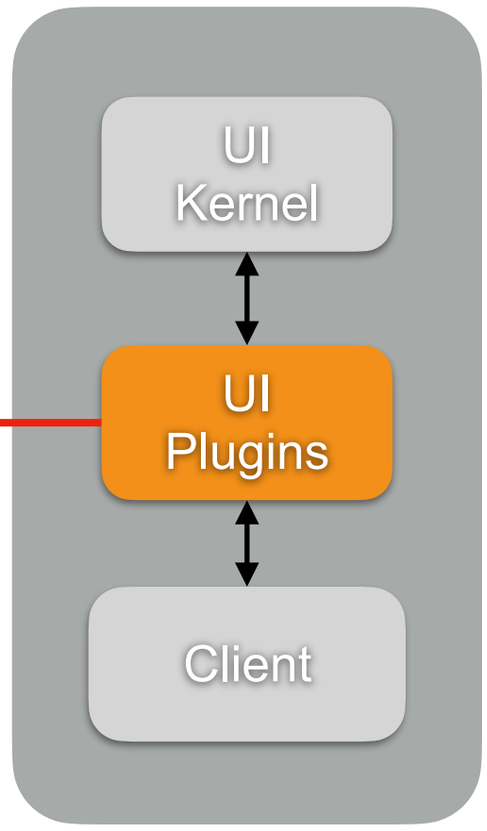
Output	Format	Description
`"reload"`	*boolean*	array fully reloaded
`"removed"`	`goal []`	removed entries
`"updated"`	`goalsData []`	updated entries
`"pending"`	*number*	remaining entries to be fetched

Writing Plugins — Client Side

223 loc

```
function RegionAnalys(): JSX.Element {
  // ....
  const [kf, setKf] = React.useState<States.Scope>();
  const scope = States.useCurrentScope();
  const regions = States.useRequest(Region.regions, kf) ?? [];
  // ...
  return (
    <>
      <Tools.ToolBar>
        <Label label='Function' />
        <LCD className='region-lcd' label={kfName ?? '---'} />
        <Tools.Button ... />
        // ...
      </Tools.ToolBar>
      <MemoryView regions={regions} />
    </>
  );
}
```

Ivette

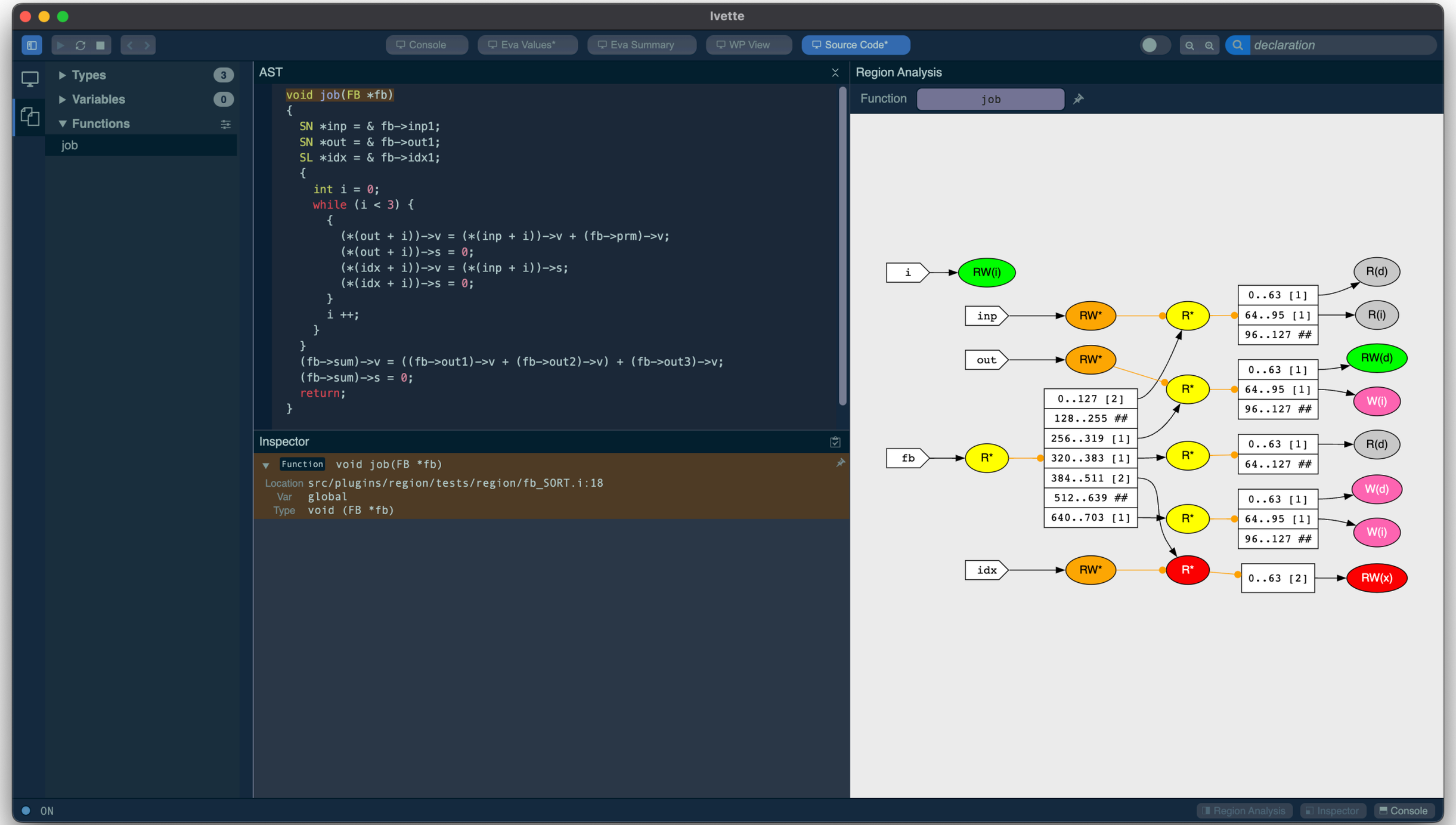


★ Generated Request API
★ Idiomatic X-TypeScript Code
★ Rich ToolKit

★ Interactive
★ Asynchronous (threads)
★ Slow & Sparse

frama C
Software Analyzers

Resulting UI

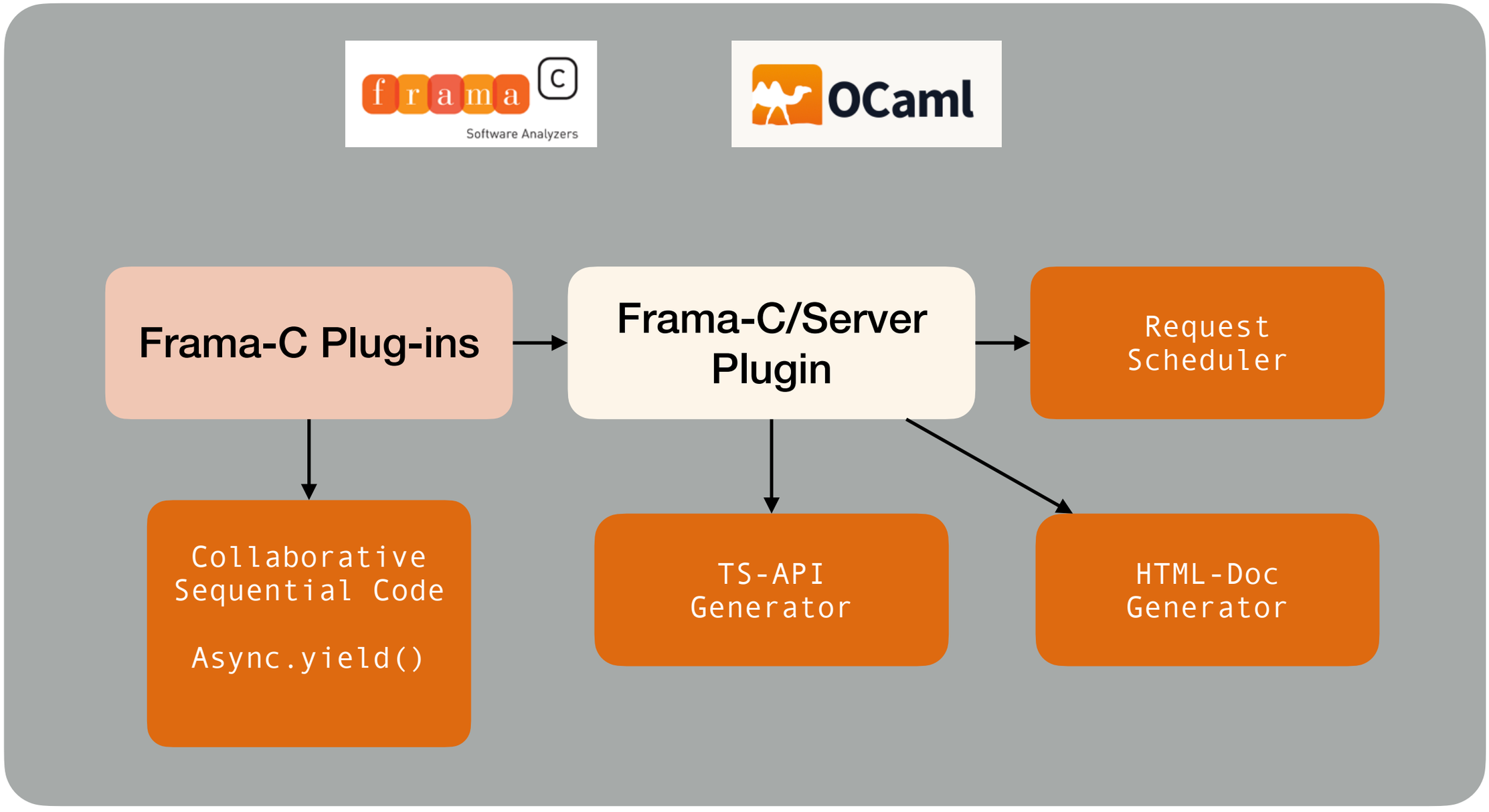


frama C
Software Analyzers

What’s that kind of Magic ?

— Middleware —

MiddleWare — Server Side



MiddleWare — UI Side

ELECTRON

React

TypeScript

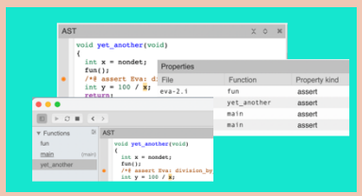
Ivette (WorkSpace)

Ivette (Plug-ins)

Frama-C Client

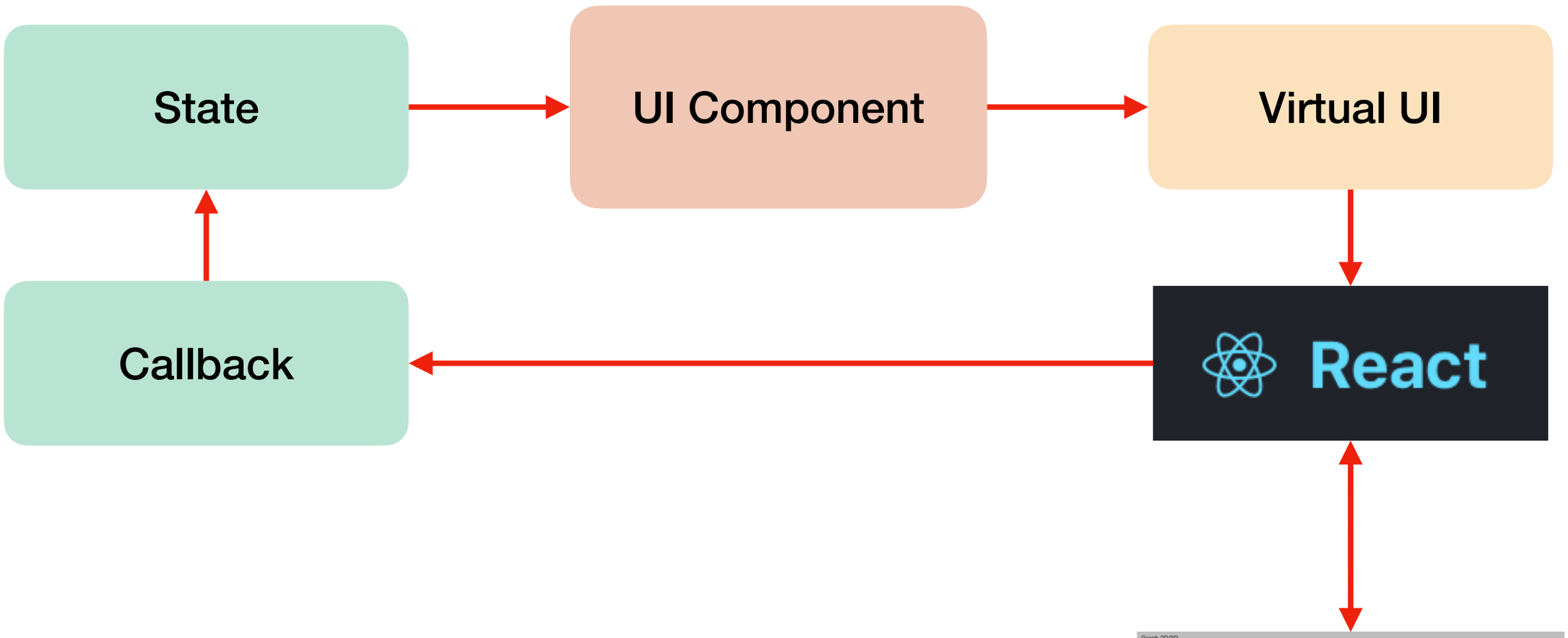
Frama-C Requests API (generated)

« Dome » Component Library

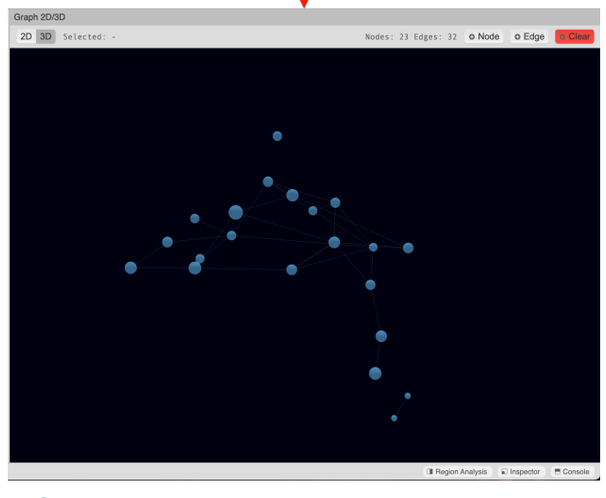


frama C
Software Analyzers

Reactive Programming

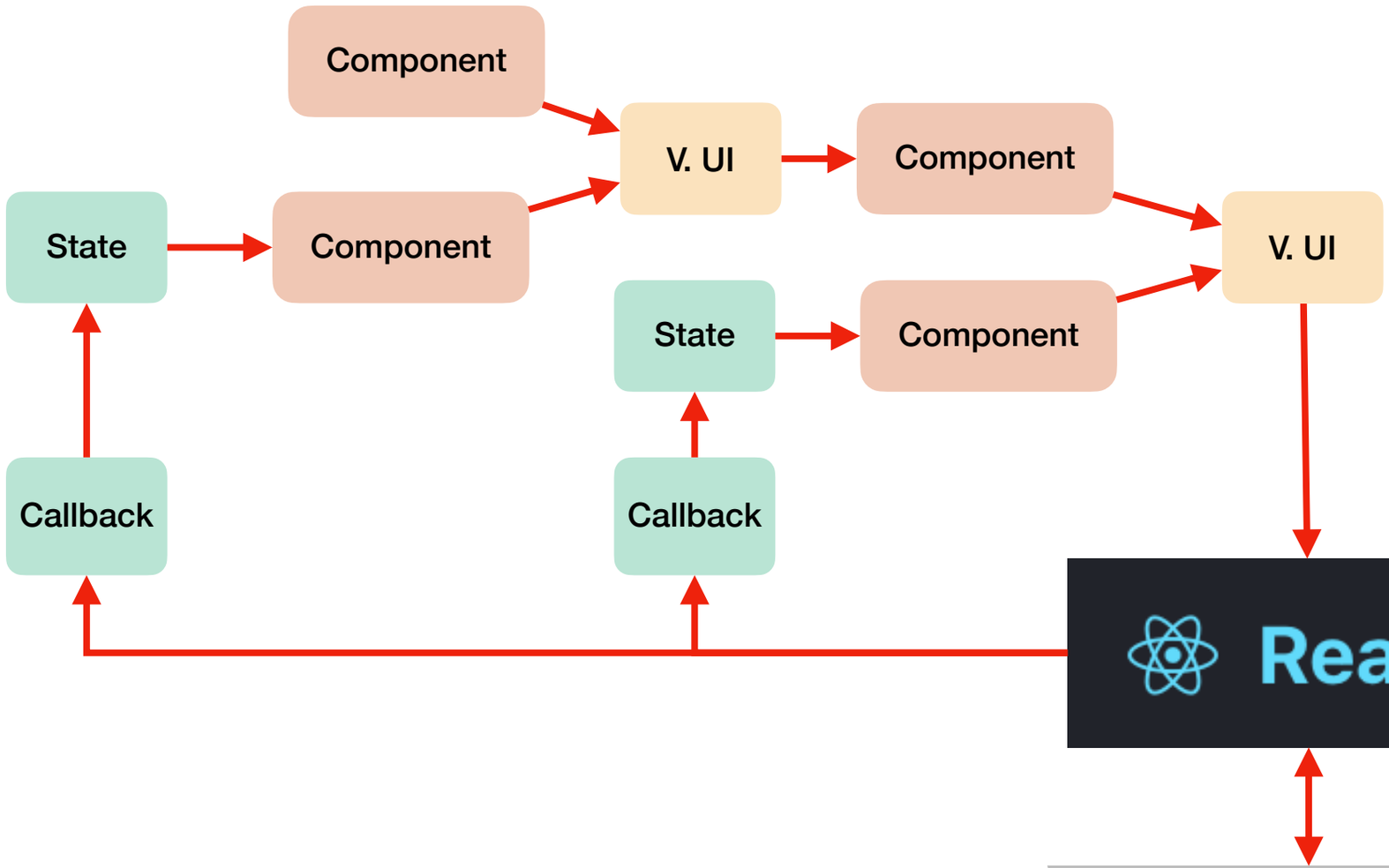


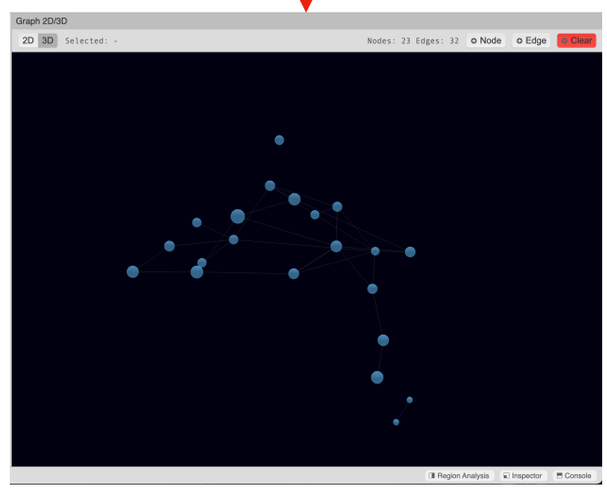
★ Simple Model
★ Functional Essence
★ Chromium Engine



ELECTRON

Composable Reactive Programming



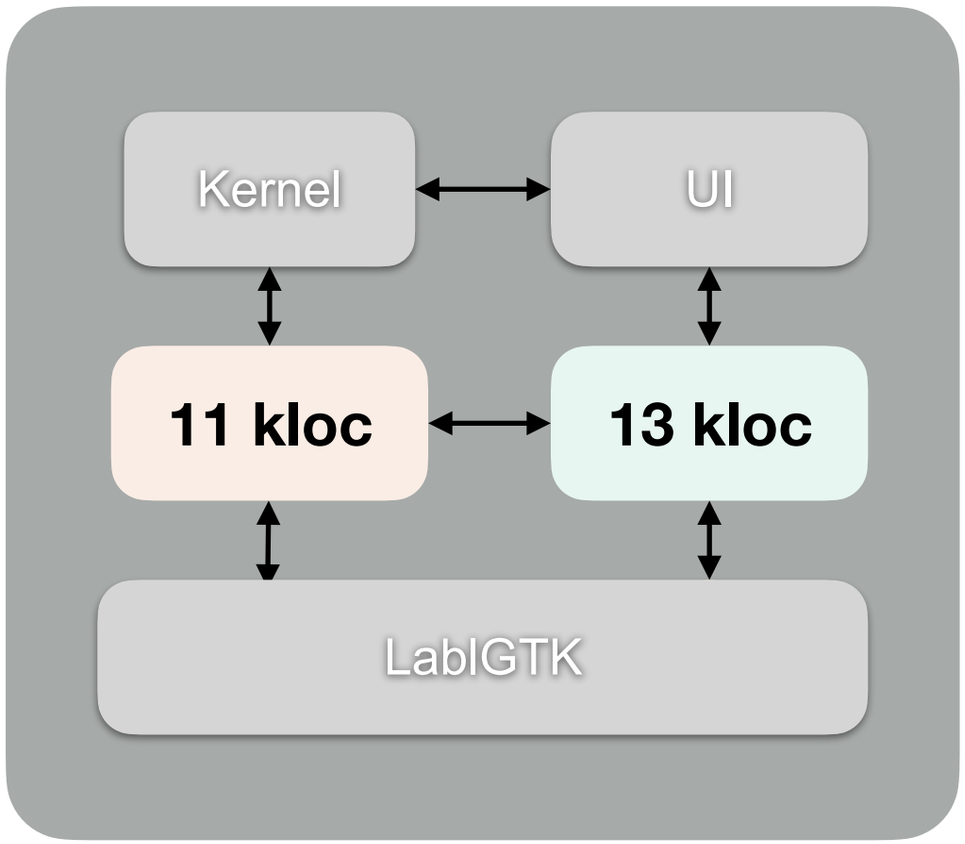


- ★ Functional
- ★ Compositional
- ★ Scales
- ★ Rich Ecosystem

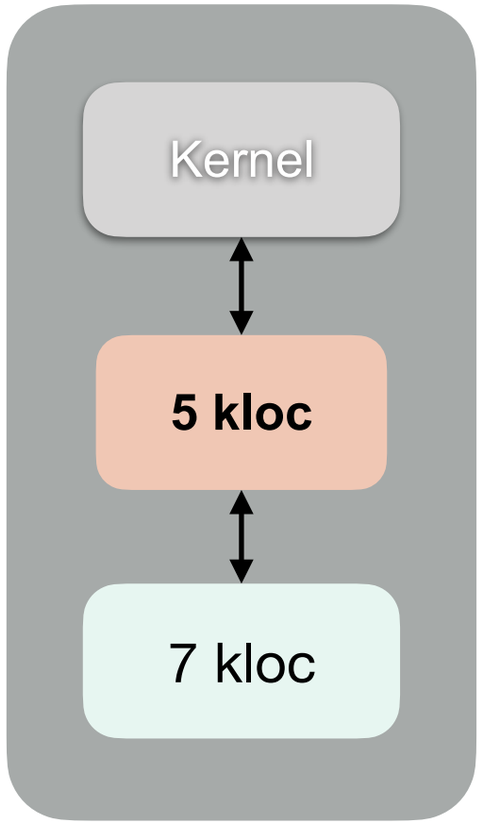
ELECTRON

Dev. Weight Comparison

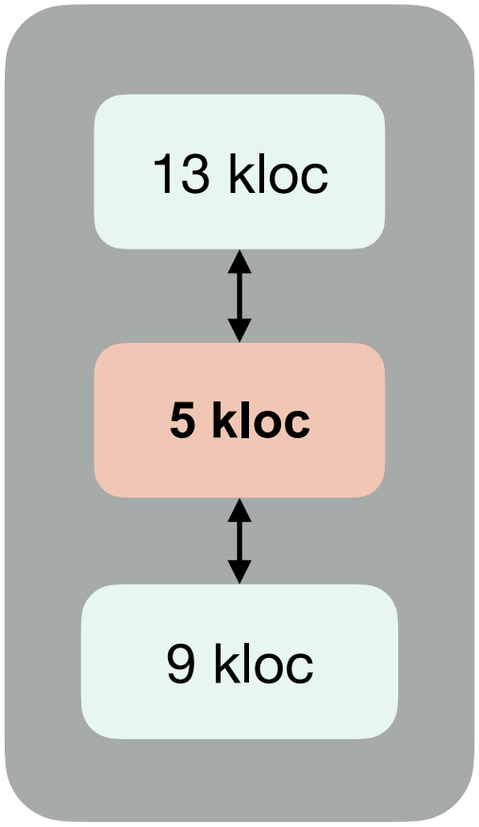
frama-c-gui



frama-c



Ivette



Old vs. New Design — Dev. Comparison

Clear Benefits

- ★ Skills Separation
- ★ Lightweight Plugins
- ★ Smooth Learning Curve

Attention Points

- ★ Outsourcing Still Difficult
- ★ Heavyweight Infrastructure
- ★ Web Ecosystem is Volatile

frama C
Software Analyzers

What’s Next ?

— Roadmap —

Timeline — Roadmap

frama C
Software Analyzers

The Ivette Team

D. Buhler

L. Correnson

V. Perrelle

M. Jacquemin

A. Maroneze

M. Alberti

R. Lazarini

And many others…

Thanks !

https://frama-c.com