Verification of software artifacts in the nuclear domain (an experience report)

Carlos Lopez Pombo

⊸Joint work with German Regis and the instrumentation and control department of the nuclear division of INVAP S.E.

IFIP WG1.3 Foundations of System Specification
Salzburg, 5-9 / February / 2024

THIS->UNRN.EDU.AR

Nuclear reactors are potentially dangerous yet safe in practice

The context

Why nuclear reactors are potentially dangerous?

- The fuel is radioactive material that can be disseminated:
  - Radiation in high dose is harmful and can even be lethal, and
  - Contamination with radioactive material can make entire regions uninhabitable for thousands of years.
- **Fuel and waste management:**
  - The nuclear fuel chain is complicated and it’s transport is in itself vulnerable for accidents, incidents and theft.
  - The use of nuclear power leads to the production of large quantities of dangerous radioactive waste.
- **Facilities safety:**
  - Facilities might be vulnerable to accidents, incidents and attacks (**Fukushima**).
  - Runaway chain reaction (**Chernobyl**)

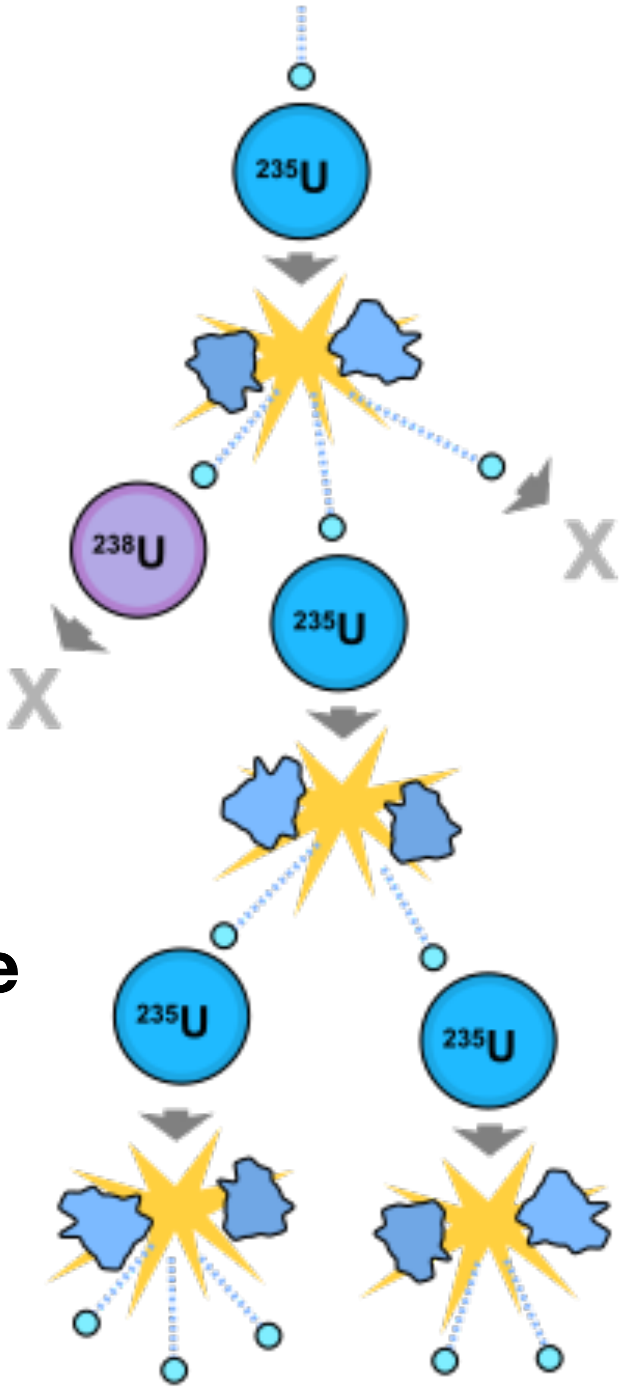
The context

What is a runaway chain reaction?

Nuclear fission chain reaction:

A uranium-235 atom absorbs a neutron and fissions into two fission fragments, releasing **two or three new neutrons** and a large amount of binding energy. Neutrons can be:

1. absorbed by an atom of uranium-238, not continuing the reaction,
2. leave the system without being absorbed, or
3. they might **collide with another atom of uranium-235**, leading to another fission, and the release of **two or three neutrons** and more binding energy.



A **runaway chain reaction** is an uncontrolled chain reaction in the presence of abundant fissionable material (**nuclear fuel rods**), resulting in an amount of energy that cannot be contained by the nuclear reactor's core (NRC).

The context

THIS->UNRN.EDU.AR

Why nuclear reactors are safe in practice?

- **How nuclear chain reactions can be controlled?**
  The NRC has **control rods** made of materials capable of absorbing neutrons, like boron, installed in a way that they can be mechanically:
  - lowered inside the NRC for slowing the chain reaction (and even stoping it) or
  - lifted for speeding it up.
- **Why safety is not an issue?**
  There are dozens of **analog signals** coming out of sensors placed inside the NRC; if any of them is **out of range**, the **control rods are automatically lowered and the chain reaction is fully stoped**.
- **How can we trust those analog signals?**
  They are measured and processed by a **voting process** of three independent **double-checked hardware modules** built using **nuclear grade certified components**.

THIS->UNRN.EDU.AR

If it is not safety, what is our role in this?

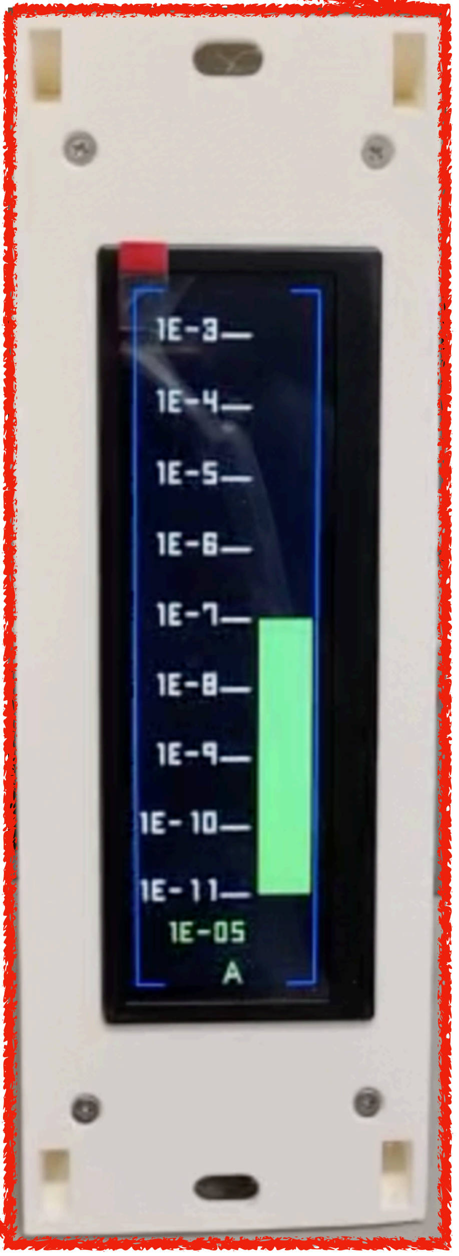
The problem

Then

- The probability of failure of nuclear grade certified electronics can be accurately estimated
- Analog processing and display of signals provides predictability according to the spec sheet of nuclear grade hardware components

The problem

Now



- Digital displays require software for processing the analog signals to be shown over digital hardware components
- Software failure probability cannot be accurately estimated, thus there is no predictability

The problem

What is the big deal with the displays?

- The cost of a nuclear reactor for producing radioisotopes (the ones built by INVAP S.E.) is between **10^8** and **10^9 euros**
- **More activity** in the NRC —>
  more energy freed by the chain reaction —>
  more production yielded by the nuclear reactor
- Displays show the **state of the NRC**, suggesting whether control bars could be **lifted** for speeding production up, or should be **lowered** for slowing it down
- If the displays **wrongly** show:
  —**less** activity than there is, control bars might be lifted forcing an emergency shut down, fully stoping the production (fuel is lost, NRC has to be cooled down and refueled, leading to **huge losses**)
  —**more** activity than there is, control bars might be lowered down slowing down production

The problem

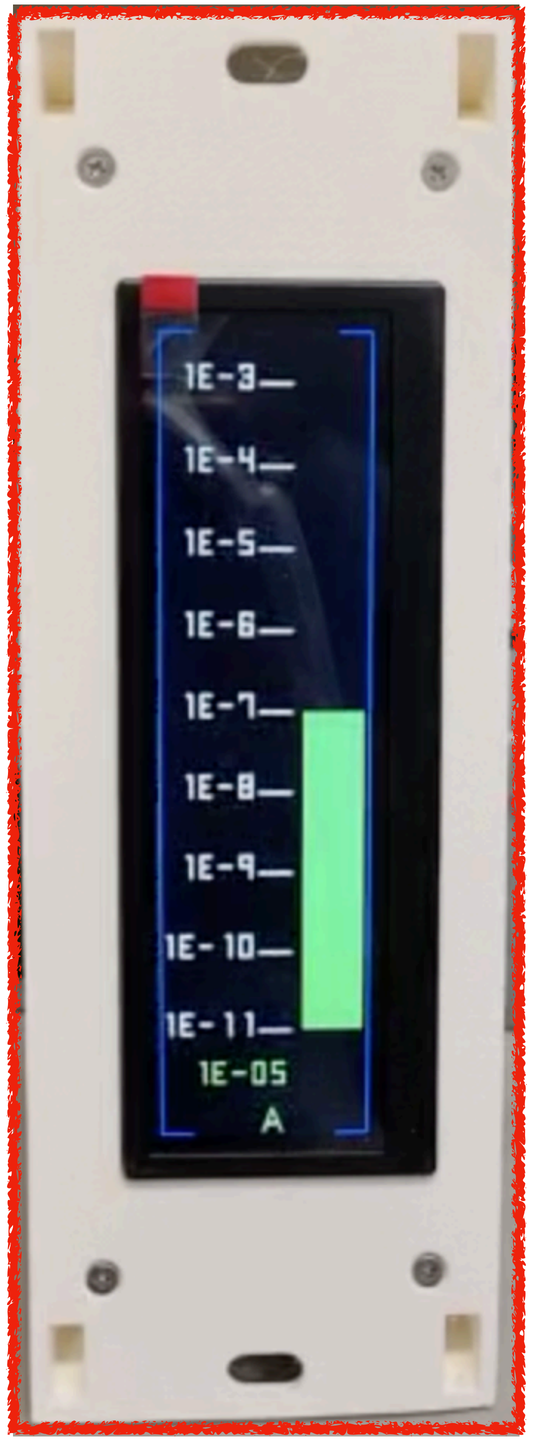
What can we offer?

- The usual practice of the software quality assurance department is to use **testing information** to clear software to be used in production.
- Formal Methods (FMs) have to be applied to the object code that will "fly"**(*)** excluding abstractions and imposing severe limitations on the manipulation of the source code
- Full certification of the correct behavior, even under prefixed desirable properties, is out of reach **[Reactive system + First order properties]**
- Application of FMs only **pays off** if the verdicts increase the confidence in the product according to the industry practices, **industry practices, standards and metrics**
- **Apply FMs** to increase confidence in the correct behavior of the system: **Case-study**: "The information shown is correct with respect to the analog-digital conversion of the signals coming out from the NRC"

(*) Industry motto: Test as you fly, fly as you test.

The problem

[<—] "The information shown is correct with respect to the analog-digital conversion of the signals coming out from the NRC"



- —>The background is invariant according to the scale in which the signal value is shown
- —>Coherence of the bar:
  - —>The first pixel of the line whose number correspond to the sample data received from the ADC is on and the one immediately on top is off
  - —>If a pixel in a line l within the boundaries of the bar is black (rest. green), then all the pixels in l within the boundaries of the bar are black (rest. green)
  - —>If a pixel in a line l and column c within the boundaries of the bar is black (rest. green), then all the pixels in line l' > l (resp. l' < l) and column c within the boundaries of the bar are black (resp. green)
- —>The pixel state showing the number representation at the bottom of the display correspond to the conversion to float of the sample data received from the ADC

The (proposed) solution

The (proposed) solution

THIS->UNRN.EDU.AR

Runtime Verification
(a compromise between testing and verification)

- **Pros:**
  - **Extends testing quite naturally:** the software executes and a tool monitors that the desired properties (i.e., the specification of the correct behavior) hold
  - **Appropriate instrumentation** allows the separation of event reporting from properties monitoring
  - **Event reporting** constitutes a relatively small time overhead that can be accurately estimated
- **Grey area:** has to be performed near-line or off-line because the correct behavior of the system depends on time related issues (e.g., the ADC has time bounds for the readings to be correct) and, in general, property checking is slow
- **Cons:** We still cannot be sure…

A runtime verification framework

Specification language and events

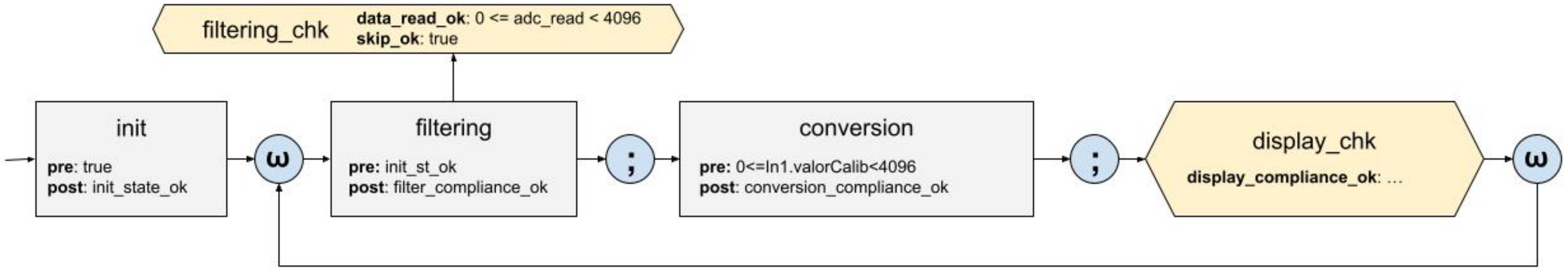
- **Specification language:**
  - **properties** to be monitored are first-order formulae with free variables expressed in SMT-LIB 2 **[1]**. Free variables are mapped to the program variables, whose values are to monitored,
  - an abstract view of the software artifact as a **workflow** consisting of **tasks** (abstracting some form of computation) and **checkpoints** (indicating points of interest in the code) as atoms, and combined with **regular operators** (including an infinite iteration operator to express reactive behavior, like control loops of embedded systems)
  - **tasks** have pre and postconditions and optional internal **checkpoints**
  - **checkpoints**, both top level and those within tasks, have a set of properties of interest to be checked

[1] Clark Barrett Pascal Fontaine Cesare Tinelli, "The SMT-LIB Standard Version 2.6”. Available at: https://smtlib.cs.uiowa.edu/language.shtml. Release: 2021-05-12

A runtime verification framework

Specification language (example)

The workflow specifying the code for displaying neutron flow in logarithmic scale**(*)**:


filtering_chk
data_read_ok: 0 <= adc_read < 4096
skip_ok: true
init
pre: true
post: init_state_ok
ω
filtering
pre: init_st_ok
post: filter_compliance_ok
;
conversion
pre: 0<=In1.valorCalib<4096
post: conversion_compliance_ok
;
display_chk
display_compliance_ok: ...
ω

(*) The pre and postconditions of tasks, and properties associated to checkpoints were abstracted away to lighten the visual representation of the model.

A runtime verification framework

Specification language and events

- **Events (and triggered actions):**
  - **variable declaration:** declares a variable name for the monitor to track values of interest (i.e., the free variables in the SMT formulae)
  - **value assignment:** reports the newly assigned value to a monitored variable for the monitor to register a change in the program state
  - **checkpoint reached:** indicates that a checkpoint has been reached (whether it is top level or within tasks), triggering the verifications: **a)** the checkpoint can be reached, according to the traversing of the workflow and, provided that this holds, **b)** the properties associated to the checkpoint hold in the current program state

A runtime verification framework

Specification language and events

- **Events (and triggered actions):**
  - **task started:** indicates that the execution of the program entered the fragment of code implementing a specific task, triggering the verifications: **a)** the task can start, according to the traversing of the workflow and, provided that this holds, **b)** the precondition holds in the current program state
  - **task finished:** indicates that the execution of the program reached the end of a fragment of code implementing a specific task, triggering the verifications: **a)** the task can finish, according to the traversing of the workflow and, provided that this holds, **b)** the postcondition holds in the current program state

A runtime verification framework

Instrumentation / reporting

- Instrumentation is done via an API written in the same language of the code to be monitored (in this case C) containing a single function (`report`)
- The function `report` takes a single parameter of type string formatted as comma separated values:

```
10068,workflow_event,task_finished,init
10069,workflow_event,task_started,filtering
10071,workflow_event,variable_value_assigned,main_contador,0
10076,hardware_event,adc,lectura,0,1661
10109,workflow_event,checkpoint_reached,filtering_chk
```

- Fields correspond to: **1)** time of the event in microseconds, **2)** type of the event (i.e., `workflow_event` / `hardware_event`), **3)** workflow event type or hardware component receiving an API function call, depending on the value of 2), **4)** task id / checkpoint id / variable id (in the monitor name space) , value reported, in the case of 2) being `workflow_event` and input parameters and output parameter separated by comma, in the case of 2) being `hardware_event`

A runtime verification framework

Hardware events

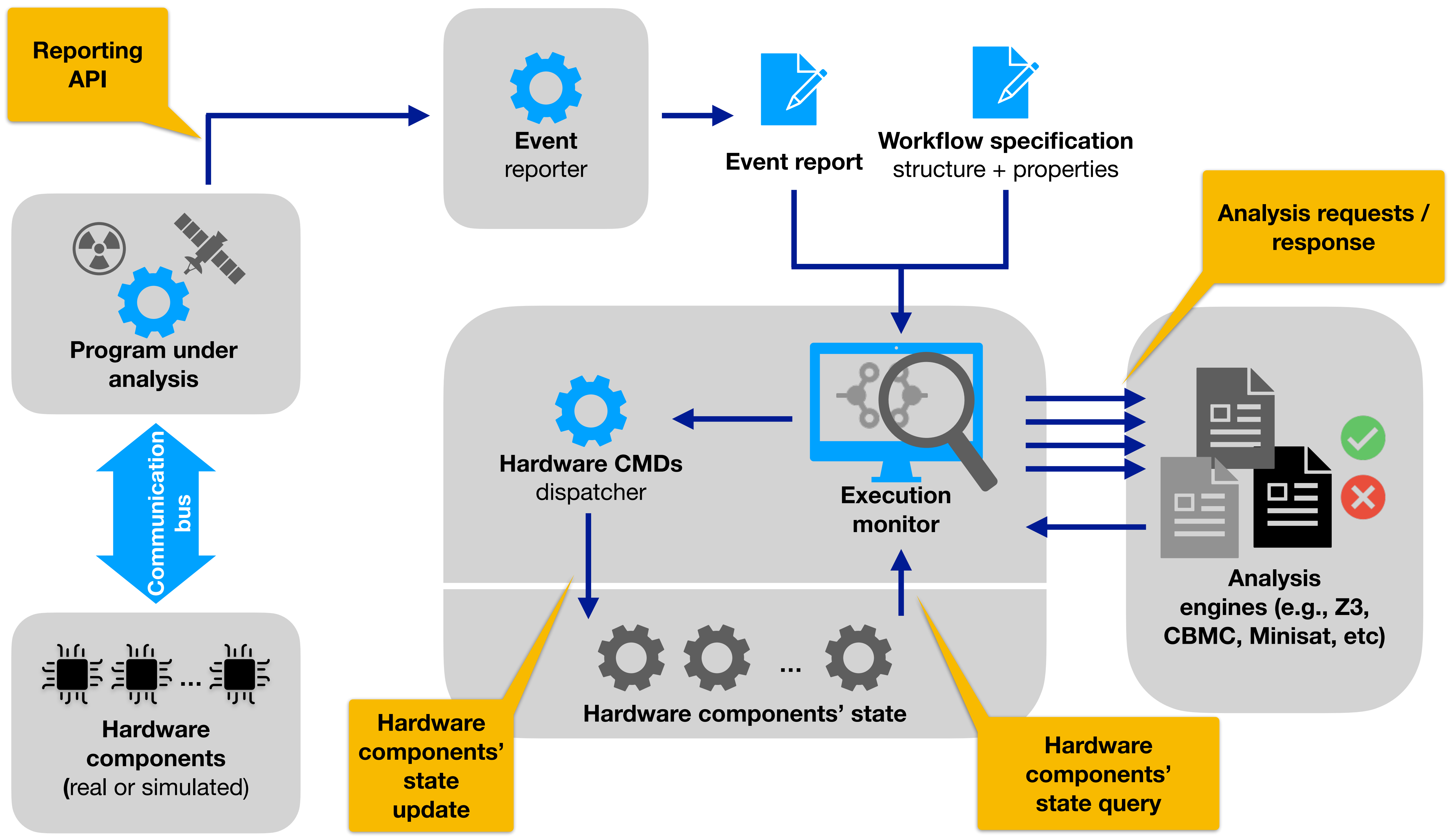
- An important aspect of the monitoring framework is capturing events related with hardware components…
- … software artifacts run within isolated microcontrollers, which are connected to specific hardware components (in our case-study, an ADC and an SSD1963 digital display)
- The **state of the system** does not only depend on the **values of the variables being monitored**, but also on the **internal state of the hardware components**
  - **Example:** Without knowing which pixels are set to which color, there is no way in which the monitor can check the integrity of the information shown in the display, and how it relates to the data measured by the sensor in the NRC

THIS->UNRN.EDU.AR

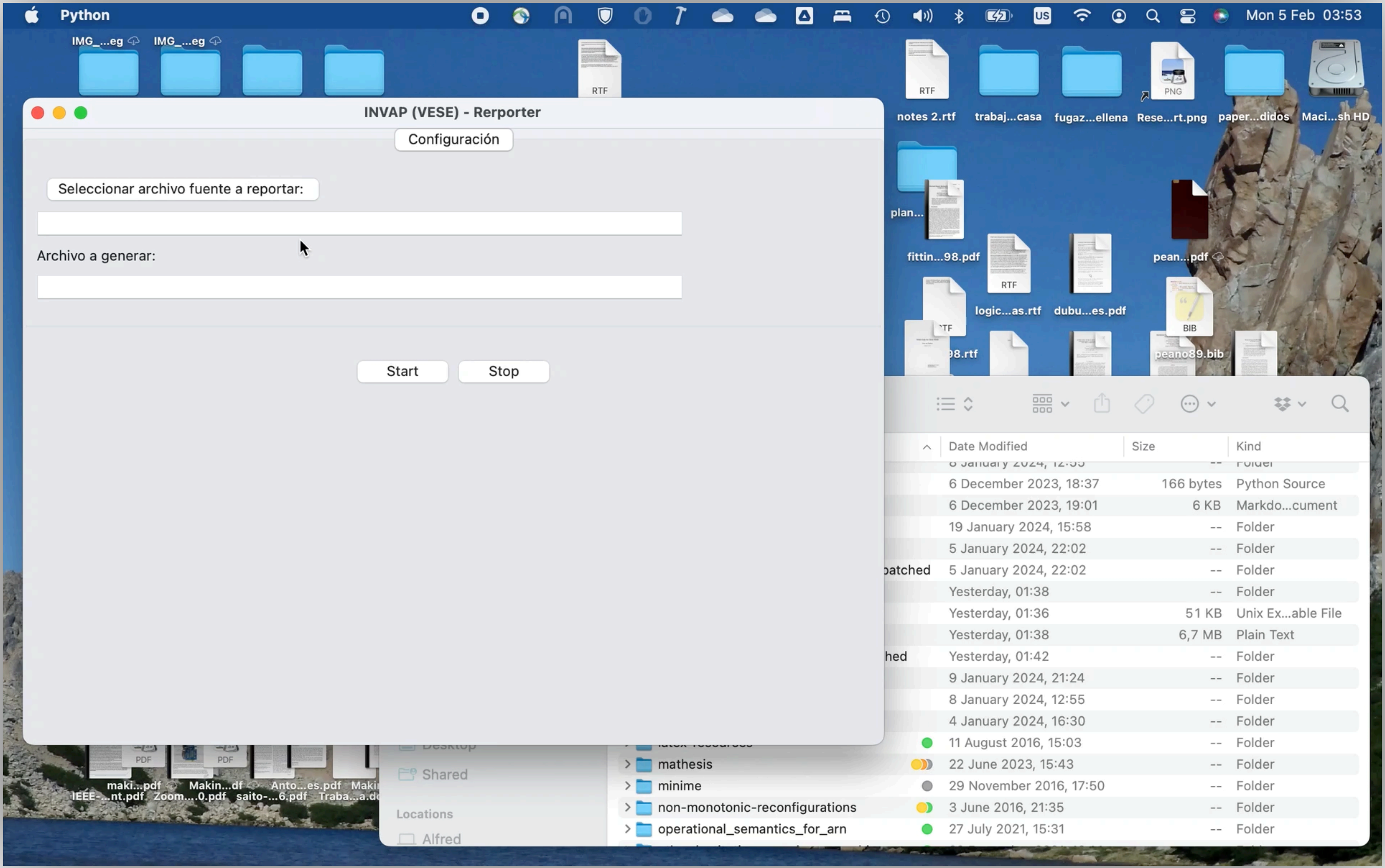
The tool

Tool Architecture

THIS->UNRN.EDU.AR

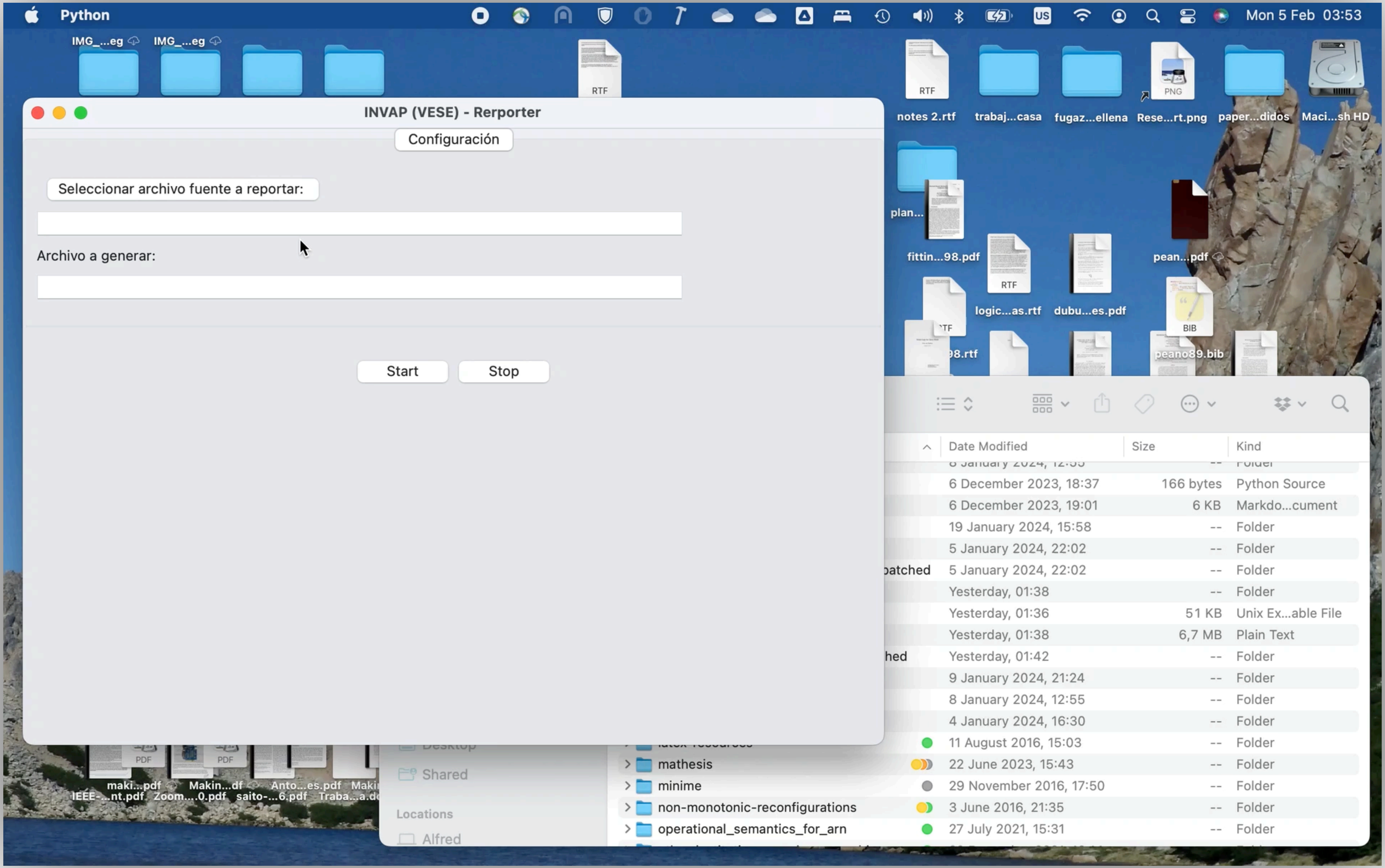

Reporting API
Event reporter
Event report
Workflow specification structure + properties
Program under analysis
Communication bus
Analysis requests / response
Hardware CMDs dispatcher
Execution monitor
Analysis engines (e.g., Z3, CBMC, Minisat, etc)
Hardware components (real or simulated)
Hardware components' state
Hardware components' state update
Hardware components' state query

Tool Demo (Reporting)



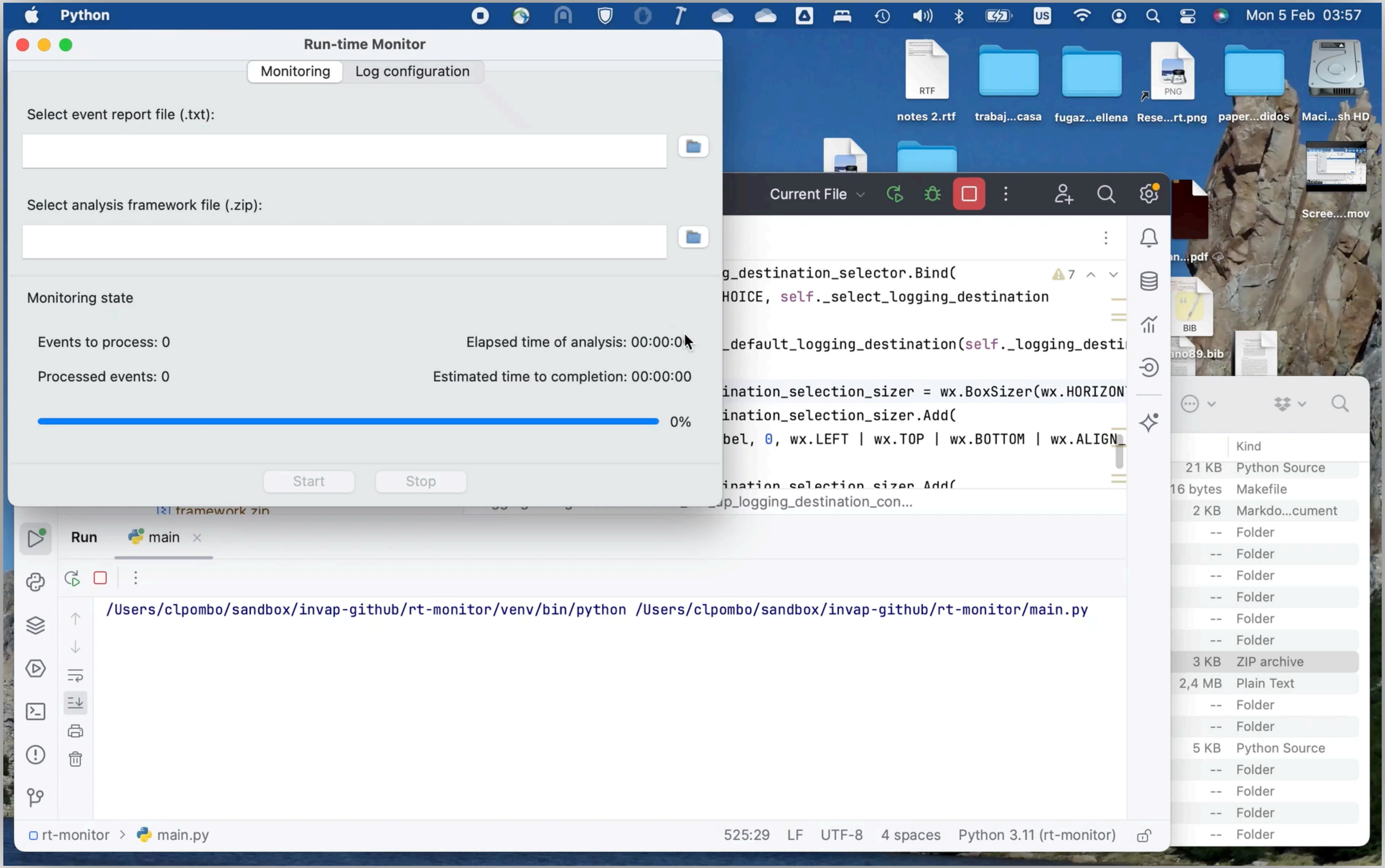
	Date Modified	Size	Kind
	6 December 2023, 18:37	166 bytes	Python Source
	6 December 2023, 19:01	6 KB	Markdo...cument
	19 January 2024, 15:58	--	Folder
	5 January 2024, 22:02	--	Folder
patched	5 January 2024, 22:02	--	Folder
	Yesterday, 01:38	--	Folder
	Yesterday, 01:36	51 KB	Unix Ex...able File
	Yesterday, 01:38	6,7 MB	Plain Text
hed	Yesterday, 01:42	--	Folder
	9 January 2024, 21:24	--	Folder
	8 January 2024, 12:55	--	Folder
	4 January 2024, 16:30	--	Folder
	11 August 2016, 15:03	--	Folder
mathesis	22 June 2023, 15:43	--	Folder
minime	29 November 2016, 17:50	--	Folder
non-monotonic-reconfigurations	3 June 2016, 21:35	--	Folder
operational_semantics_for_arn	27 July 2021, 15:31	--	Folder

Tool Demo (Reporting)



	Date Modified	Size	Kind
	6 December 2023, 18:37	166 bytes	Python Source
	6 December 2023, 19:01	6 KB	Markdo...cument
	19 January 2024, 15:58	--	Folder
	5 January 2024, 22:02	--	Folder
atched	5 January 2024, 22:02	--	Folder
	Yesterday, 01:38	--	Folder
	Yesterday, 01:36	51 KB	Unix Ex...able File
	Yesterday, 01:38	6,7 MB	Plain Text
hed	Yesterday, 01:42	--	Folder
	9 January 2024, 21:24	--	Folder
	8 January 2024, 12:55	--	Folder
	4 January 2024, 16:30	--	Folder
	11 August 2016, 15:03	--	Folder
mathesis	22 June 2023, 15:43	--	Folder
minime	29 November 2016, 17:50	--	Folder
non-monotonic-reconfigurations	3 June 2016, 21:35	--	Folder
operational_semantics_for_arn	27 July 2021, 15:31	--	Folder

Tool Demo (Monitoring)



Select event report file (.txt):

Select analysis framework file (.zip):

Monitoring state

Events to process: 0 Elapsed time of analysis: 00:00:0

Processed events: 0 Estimated time to completion: 00:00:00

0%

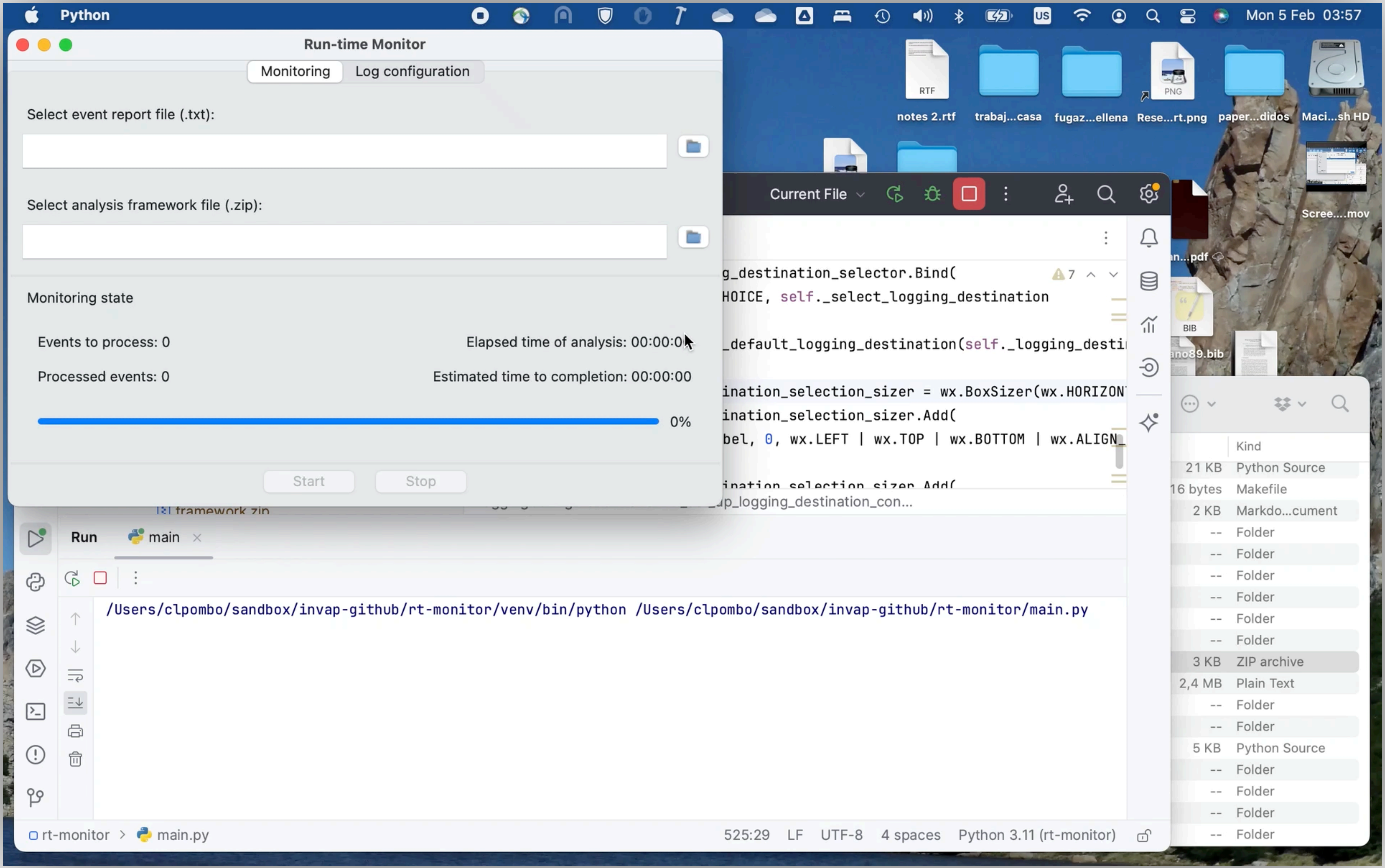
```
g_destination_selector.Bind(
HOICE, self._select_logging_destination

_default_logging_destination(self._logging_desti

ination_selection_sizer = wx.BoxSizer(wx.HORIZON
ination_selection_sizer.Add(
bel, 0, wx.LEFT | wx.TOP | wx.BOTTOM | wx.ALIGN_
```

```
/Users/clpombo/sandbox/invap-github/rt-monitor/venv/bin/python /Users/clpombo/sandbox/invap-github/rt-monitor/main.py
```

rt-monitor > main.py 525:29 LF UTF-8 4 spaces Python 3.11 (rt-monitor)

21 KB	Python Source
16 bytes	Makefile
2 KB	Markdo...cument
--	Folder
--	Folder
--	Folder
--	Folder
--	Folder
--	Folder
3 KB	ZIP archive
2,4 MB	Plain Text
--	Folder
--	Folder
5 KB	Python Source
--	Folder
--	Folder
--	Folder
--	Folder

Tool Demo (Monitoring)



Run-time Monitor

Monitoring | Log configuration

Select event report file (.txt):

Select analysis framework file (.zip):

Monitoring state

Events to process: 0 Elapsed time of analysis: 00:00:00

Processed events: 0 Estimated time to completion: 00:00:00

0%

Start Stop

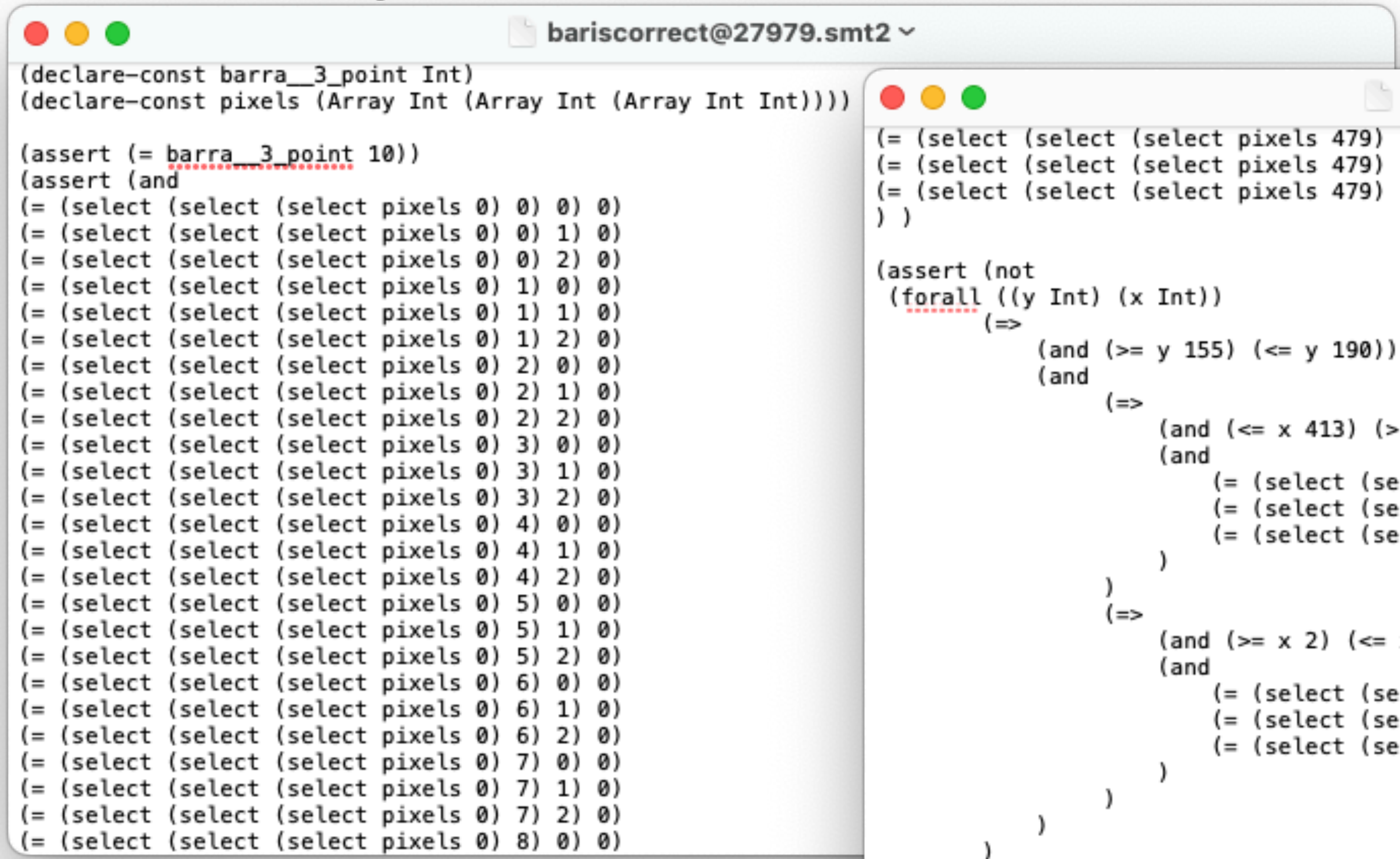
```
g_destination_selector.Bind(
HOICE, self._select_logging_destination

_default_logging_destination(self._logging_desti

ination_selection_sizer = wx.BoxSizer(wx.HORIZON
ination_selection_sizer.Add(
bel, 0, wx.LEFT | wx.TOP | wx.BOTTOM | wx.ALIGN_

ination_selection_sizer.Add(
```

```
/Users/clpombo/sandbox/invap-github/rt-monitor/venv/bin/python /Users/clpombo/sandbox/invap-github/rt-monitor/main.py
```

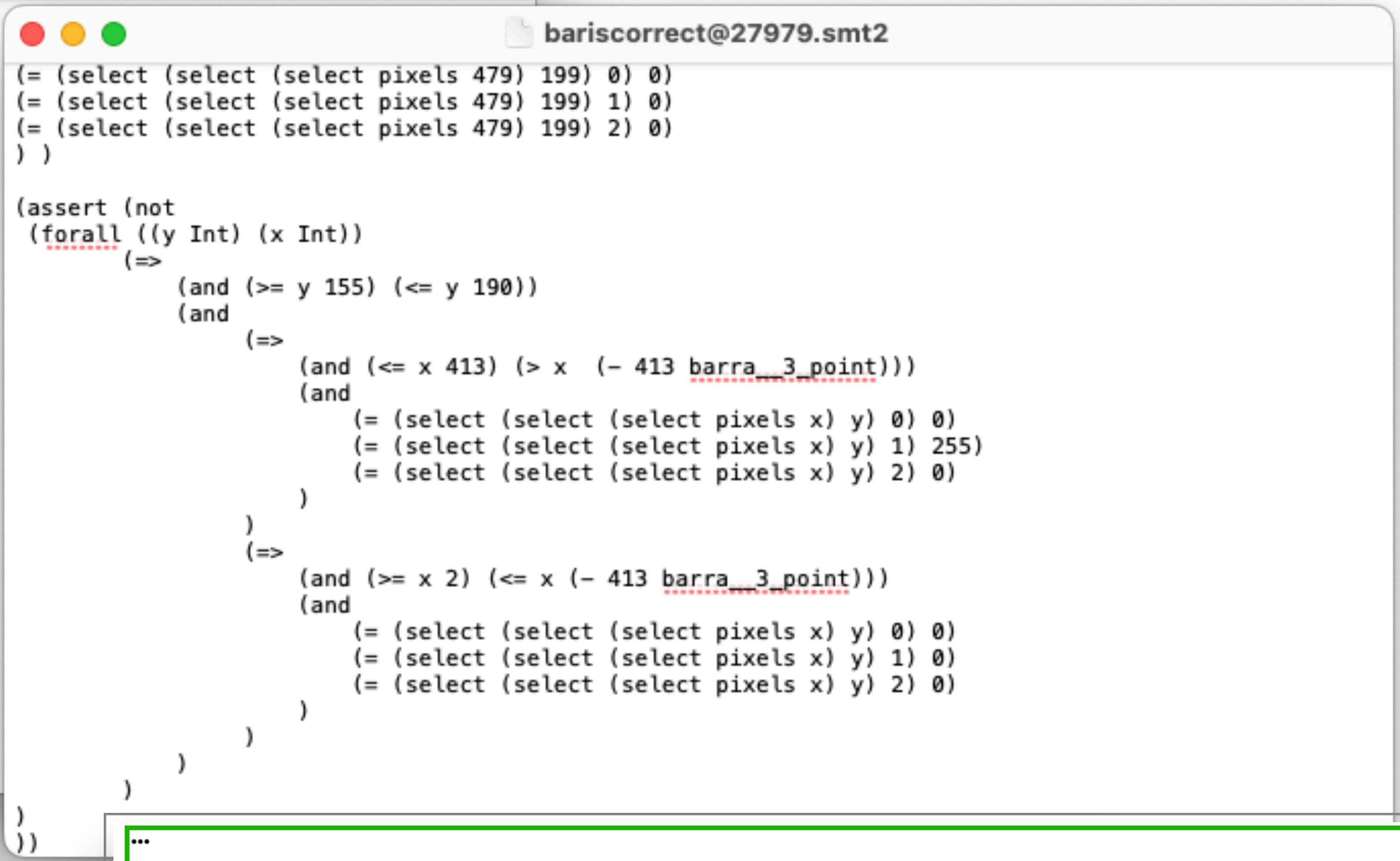
525:29 LF UTF-8 4 spaces Python 3.11 (rt-monitor)

	Kind
21 KB	Python Source
16 bytes	Makefile
2 KB	Markdo...cument
--	Folder
--	Folder
--	Folder
--	Folder
--	Folder
--	Folder
3 KB	ZIP archive
2,4 MB	Plain Text
--	Folder
--	Folder
5 KB	Python Source
--	Folder
--	Folder
--	Folder
--	Folder

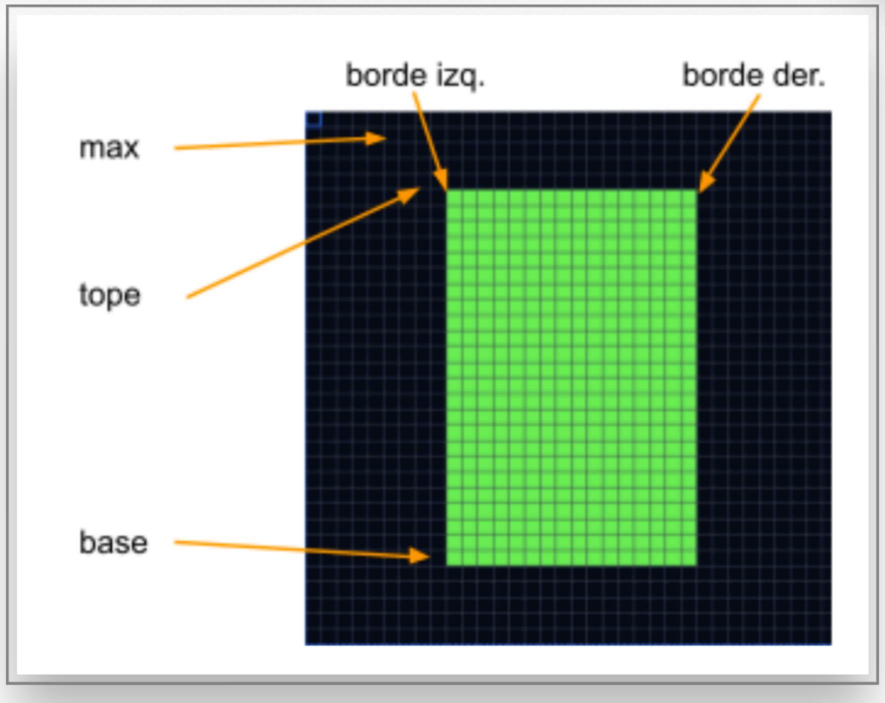
The “bug” found

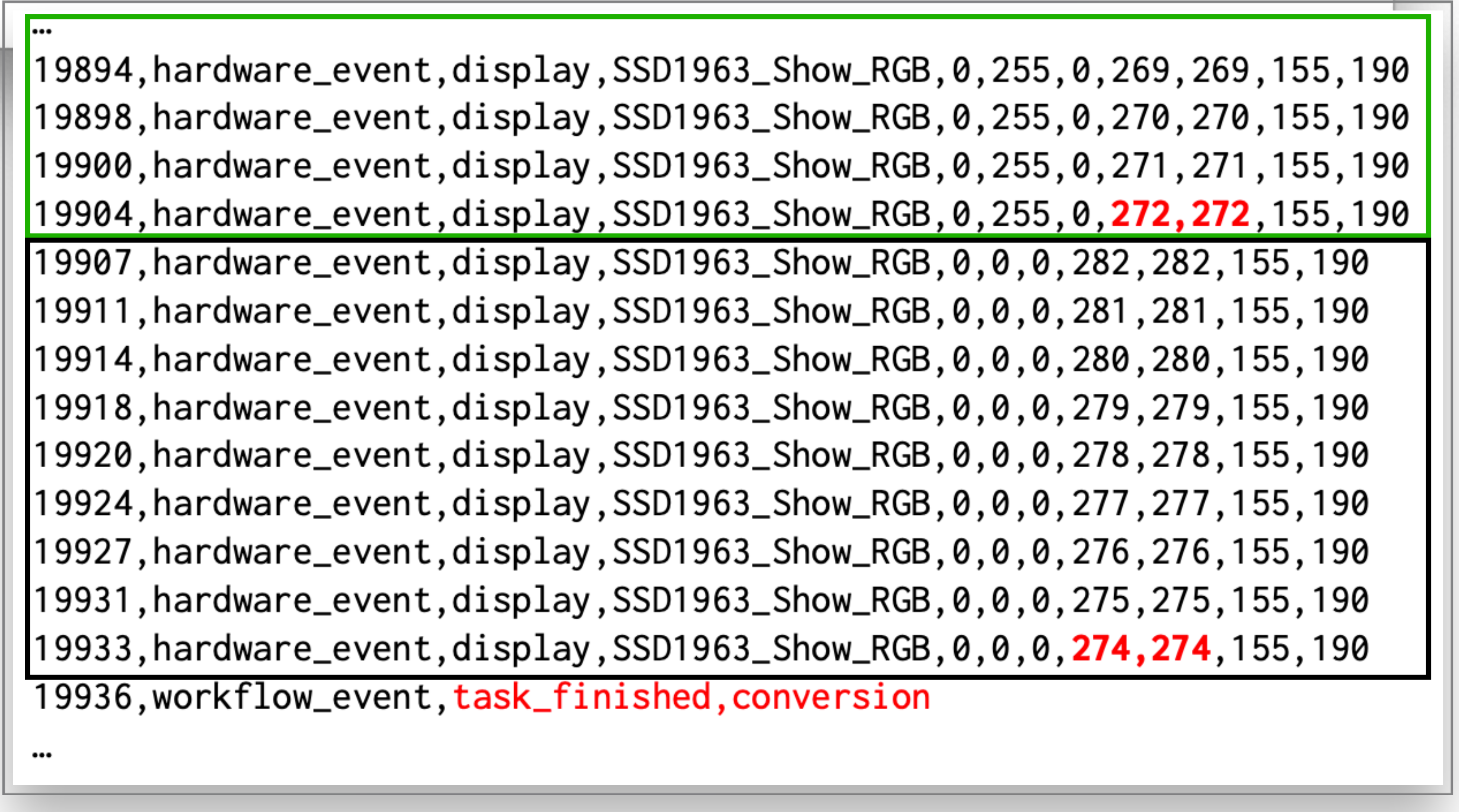

bariscorrect@27979.smt2

```
(declare-const barra__3_point Int)
(declare-const pixels (Array Int (Array Int (Array Int Int))))

(assert (= barra__3_point 10))
(assert (and
(= (select (select (select pixels 0) 0) 0) 0)
(= (select (select (select pixels 0) 0) 1) 0)
(= (select (select (select pixels 0) 0) 2) 0)
(= (select (select (select pixels 0) 1) 0) 0)
(= (select (select (select pixels 0) 1) 1) 0)
(= (select (select (select pixels 0) 1) 2) 0)
(= (select (select (select pixels 0) 2) 0) 0)
(= (select (select (select pixels 0) 2) 1) 0)
(= (select (select (select pixels 0) 2) 2) 0)
(= (select (select (select pixels 0) 3) 0) 0)
(= (select (select (select pixels 0) 3) 1) 0)
(= (select (select (select pixels 0) 3) 2) 0)
(= (select (select (select pixels 0) 4) 0) 0)
(= (select (select (select pixels 0) 4) 1) 0)
(= (select (select (select pixels 0) 4) 2) 0)
(= (select (select (select pixels 0) 5) 0) 0)
(= (select (select (select pixels 0) 5) 1) 0)
(= (select (select (select pixels 0) 5) 2) 0)
(= (select (select (select pixels 0) 6) 0) 0)
(= (select (select (select pixels 0) 6) 1) 0)
(= (select (select (select pixels 0) 6) 2) 0)
(= (select (select (select pixels 0) 7) 0) 0)
(= (select (select (select pixels 0) 7) 1) 0)
(= (select (select (select pixels 0) 7) 2) 0)
(= (select (select (select pixels 0) 8) 0) 0)
```


bariscorrect@27979.smt2

```
(= (select (select (select pixels 479) 199) 0) 0)
(= (select (select (select pixels 479) 199) 1) 0)
(= (select (select (select pixels 479) 199) 2) 0)
) )

(assert (not
 (forall ((y Int) (x Int))
        (=>
            (and (>= y 155) (<= y 190))
            (and
                (=>
                    (and (<= x 413) (> x  (- 413 barra__3_point)))
                    (and
                        (= (select (select (select pixels x) y) 0) 0)
                        (= (select (select (select pixels x) y) 1) 255)
                        (= (select (select (select pixels x) y) 2) 0)
                    )
                )
                (=>
                    (and (>= x 2) (<= x (- 413 barra__3_point)))
                    (and
                        (= (select (select (select pixels x) y) 0) 0)
                        (= (select (select (select pixels x) y) 1) 0)
                        (= (select (select (select pixels x) y) 2) 0)
                    )
                )
            )
        )
)
))
```





```
…
19894,hardware_event,display,SSD1963_Show_RGB,0,255,0,269,269,155,190
19898,hardware_event,display,SSD1963_Show_RGB,0,255,0,270,270,155,190
19900,hardware_event,display,SSD1963_Show_RGB,0,255,0,271,271,155,190
19904,hardware_event,display,SSD1963_Show_RGB,0,255,0,272,272,155,190
19907,hardware_event,display,SSD1963_Show_RGB,0,0,0,282,282,155,190
19911,hardware_event,display,SSD1963_Show_RGB,0,0,0,281,281,155,190
19914,hardware_event,display,SSD1963_Show_RGB,0,0,0,280,280,155,190
19918,hardware_event,display,SSD1963_Show_RGB,0,0,0,279,279,155,190
19920,hardware_event,display,SSD1963_Show_RGB,0,0,0,278,278,155,190
19924,hardware_event,display,SSD1963_Show_RGB,0,0,0,277,277,155,190
19927,hardware_event,display,SSD1963_Show_RGB,0,0,0,276,276,155,190
19931,hardware_event,display,SSD1963_Show_RGB,0,0,0,275,275,155,190
19933,hardware_event,display,SSD1963_Show_RGB,0,0,0,274,274,155,190
19936,workflow_event,task_finished,conversion
…
```

The missing pieces

THIS->UNRN.EDU.AR

- **Distributed analysis:** we are implementing a parallel and distributed, multi language verification engine.
- **Transparent reporting of hardware events:** we are reporting hardware events at software level but we will move to a more transparent approach by sniffing ports.
- **Timing:** we are working on adding time invariants to tasks enabling the additional verification of time constraints when tasks start or finish, and when checkpoints are reached.
- **Monitoring temporal properties:** we plan to implement a monitoring algorithm for temporal properties, based on the work on Stream Runtime Verification of Manna, Sanches, et.al.**[2]**
- **Interface and usability:** rough and minimal, we plan to improve on this as soon as we get some experience in production

[2] Online and Offline Stream Runtime Verification of Synchronous Systems. In Christian Colombo, Martin Leucker, eds: Proceedings of the 18th International Conference Runtime Verification - RV 2018, Limassol, Cyprus, November 10-13, 2018. Lecture Notes in Computer Science 11237, pp. 138-163. Springer-Verlag, 2018.

? & !

THIS->UNRN.EDU.AR