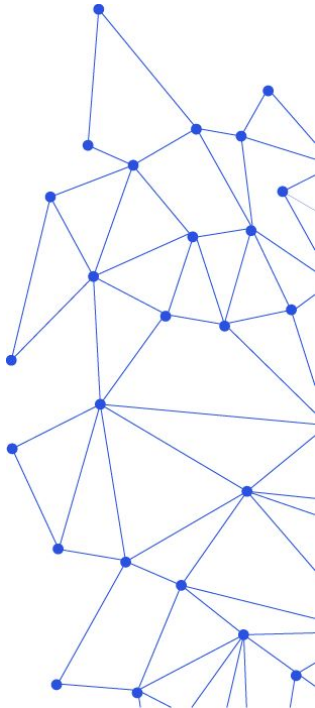
fortiss

Experimental Comparison between AutoFOCUS3 and Papyrus-RT

Tatiana Chuprina, Florian Hölzl, Vincent Aravantinos



Context – AutoFOCUS 3

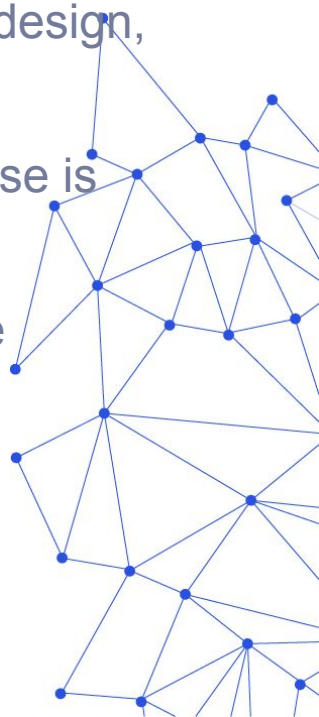
fortiss

AutoFOCUS (in short “AF3”):

- Model-based development tool
- Main application domain: embedded systems (aerospace, automotive, automation, railways…)
- Integration of the complete development process: requirements, design, simulation, test, code generation
- Seamless integration: everything in one tool, artifact of every phase is connected to other phases

  → traceability comes for free, semantics is the same everywhere
- Models have independently defined execution semantics

  → enabler for simulation, code generation, formal verification



Comparison AF3 / Papyrus: Motivation

fortiss

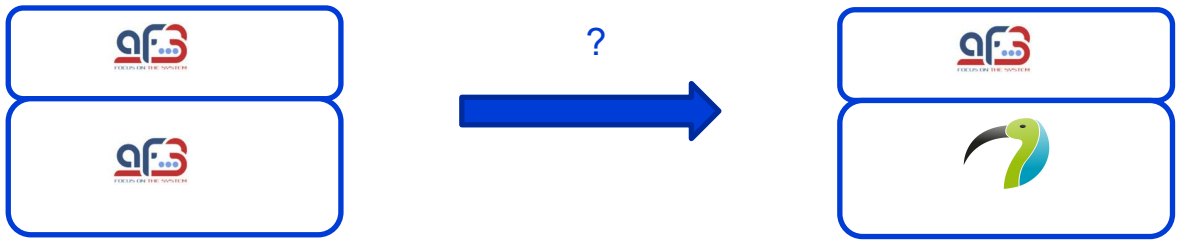
Observations:

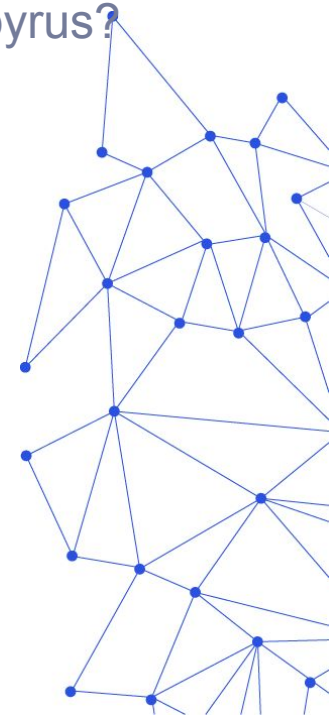
- AF3 development takes a tremendous amount of time

  → Especially GUI effort is huge
- Papyrus is getting more and more mature

- why reinvent the wheel when we can reuse the effort put into Papyrus?

Long-term idea: AF3 as a Papyrus/UML profile?





AF3 / Papyrus: differences

fortiss

Syntax:

- Papyrus: UML → rich, standardized, but hard to use
- AF3: own syntax →restricted, but simple to use

But in the context of AF3, **syntax is not enough.**

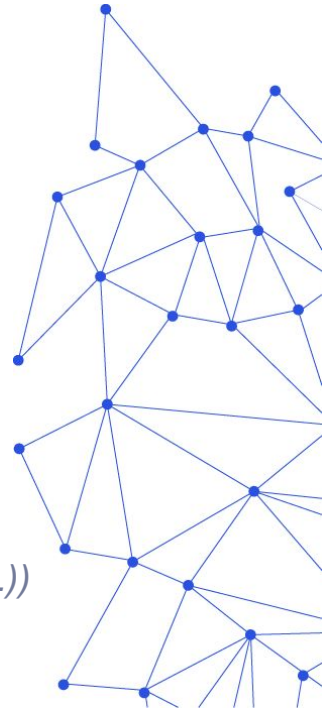
Execution semantics:

- Papyrus: UML → per se, none

  But:

  - fUML (tool: Moka)
  - UML RT (tool: Papyrus-RT)
  - Question to the community: others?
- AF3: FOCUS

  (M.Broy, K.Stolen, 2001 - Specification and Development of Interactive Systems(…))



Plan of the experiment

fortiss

Following AF3 artifacts, find the best Papyrus corresponding match:

1. First for state automata (easy)
2. Then for components (not easy for standard UML, easy for UML-RT)

For each of these matches, compare:

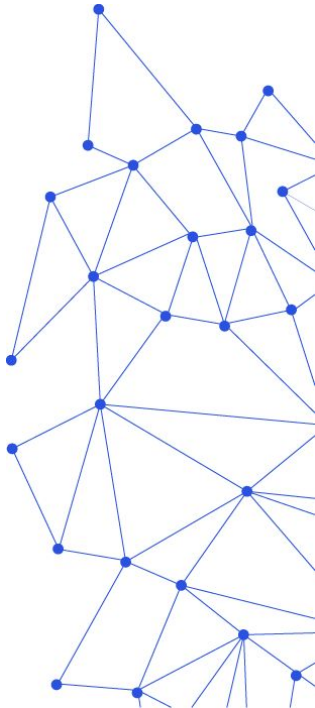
1. Syntax
2. Semantics

→ All through the tooling, and an example model

Comparing the semantics is done by examining

1. Simulation
2. Code Generation

capabilities.



Setup with Moka and Papyrus-RT

fortiss

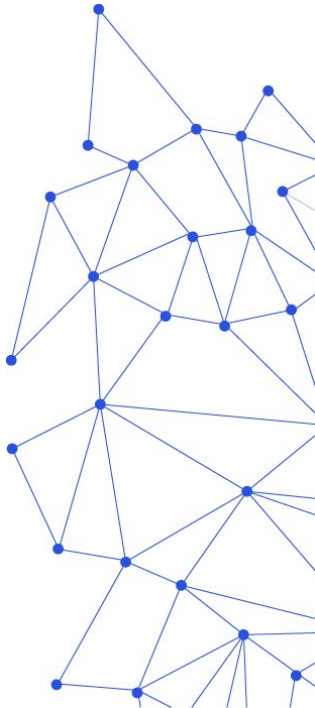
Moka:

- No code generation (yet?)
- AutoFOCUS3 is heavily based on state machines
- Moka does not yet support state machines (officially)
- Despite several attempts with the development version
  (thanks to Jérémy Tatibouet!)
  it was not mature enough for the needs of the experiment

Papyrus-RT:

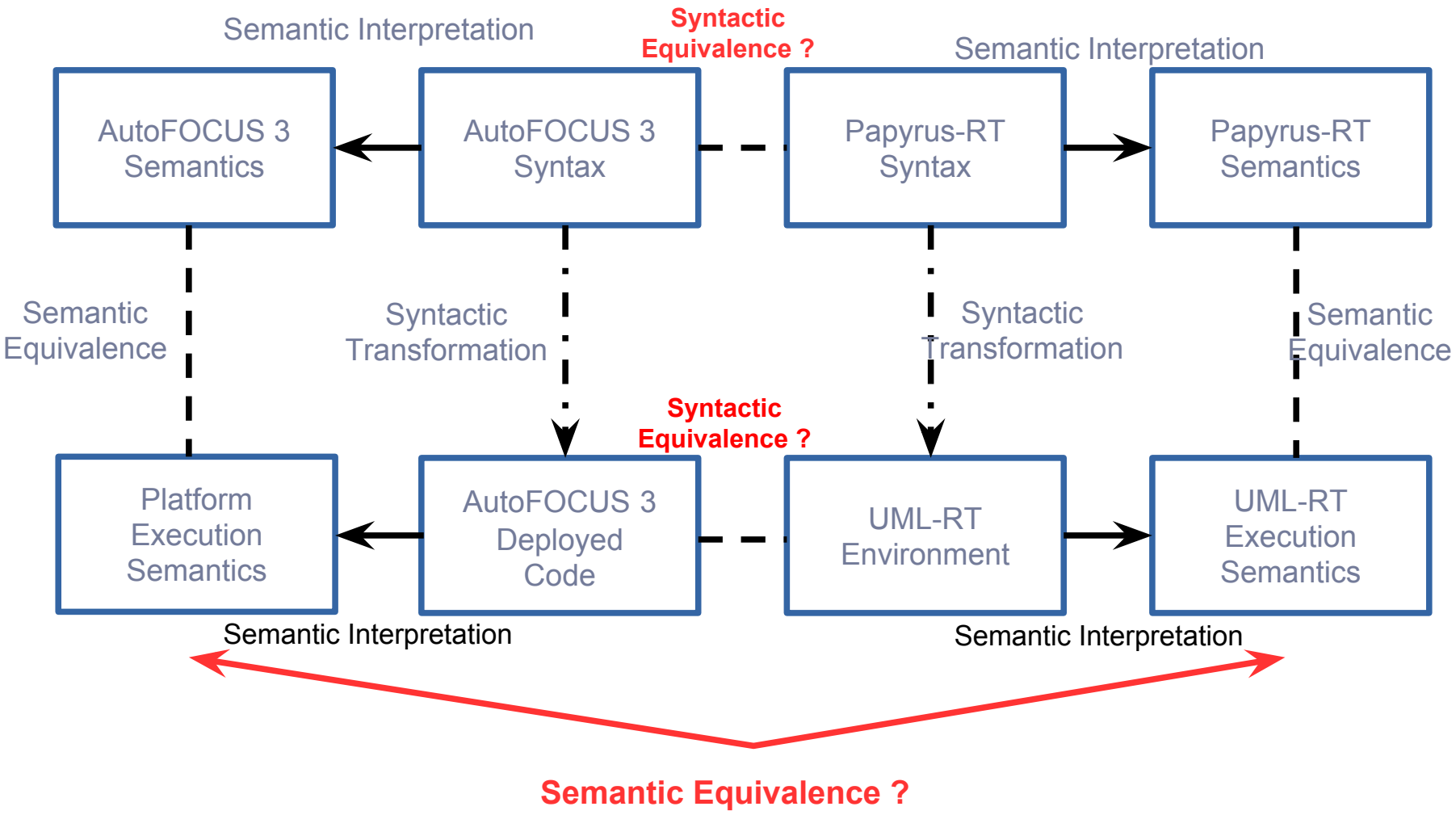
- Capsules looked very much like AF3 components
- No simulation (yet?)
- Lots of feedback from Zeligsoft
  (thanks to Ernesto Posse and Charles Rivet!)

⇒ Focus on Papyrus-RT in the rest of the experiment



Syntactic and Semantic Equivalences, Interpretations, and Transformations

fortiss

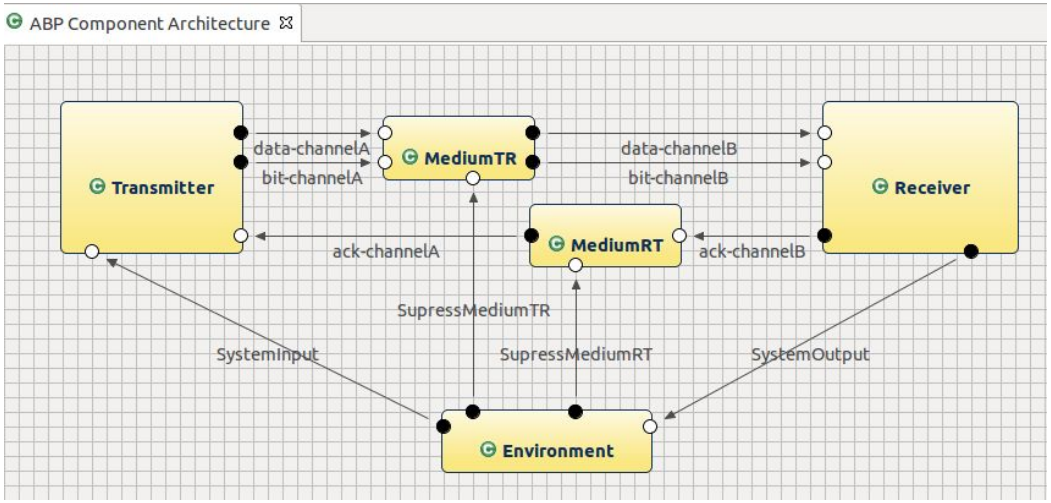


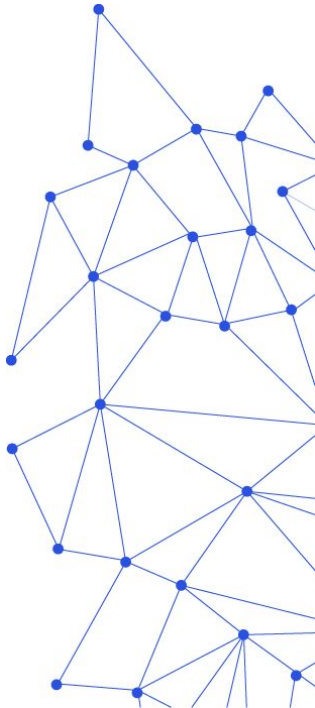
Example: Alternating Bit Protocol

fortiss

Choice of this example:

- Simple: transmitter, receiver, 2 media components
- Relevant: real distributed application
- Adaptable: lossy vs. non-lossy medium implementation
- Well-Known: used countless times





Results of the Experiment

fortiss

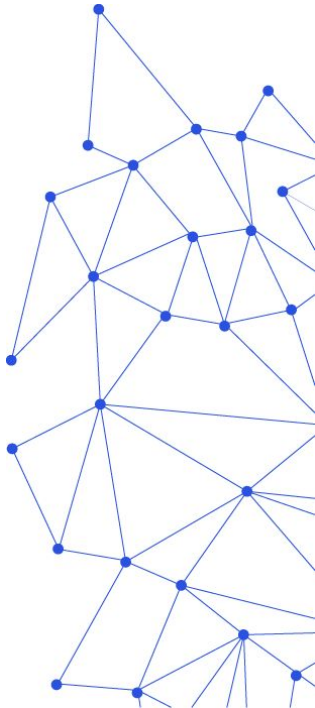
Semantic aspects

Semantic differences:

Category	Papyrus-RT	AutoFOCUS 3
System Model	*Multi-threaded System*	*Distributed System*
Synchronization Semantics	*Completely left to the modeler*	*Common, synchronous notion of time*
Relation to target language	*Target language code embedded in the model*	*Models are transformed into target code*

→ different philosophy for the user:

- AutoFOCUS3 is “model first”
- Papyrus-RT proposes a mix between “model” and “code”

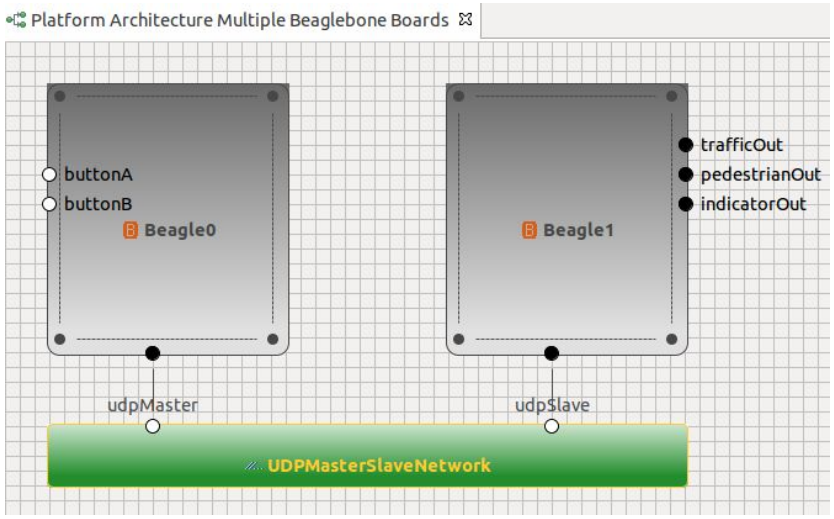


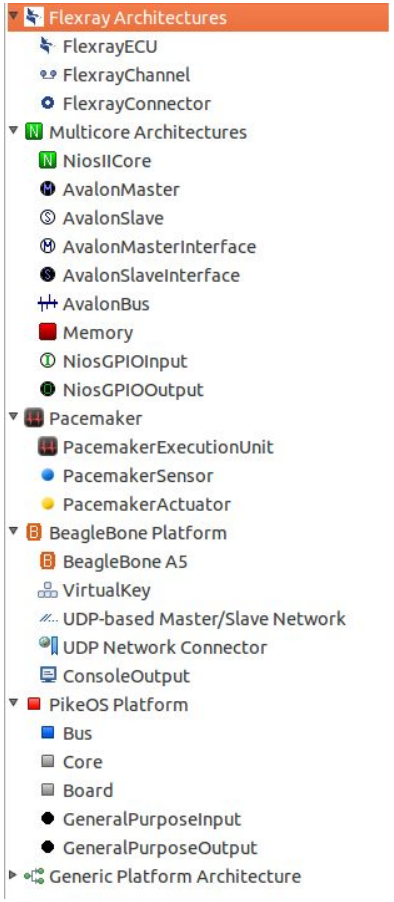
AutoFOCUS 3 Platform Model

fortiss

Platform Model describes

- Execution Units (e.g. Electronic Control Units in a car)
- Transmission Units (e.g. Flexray-Bus in a car)
- Sensors and Actuators





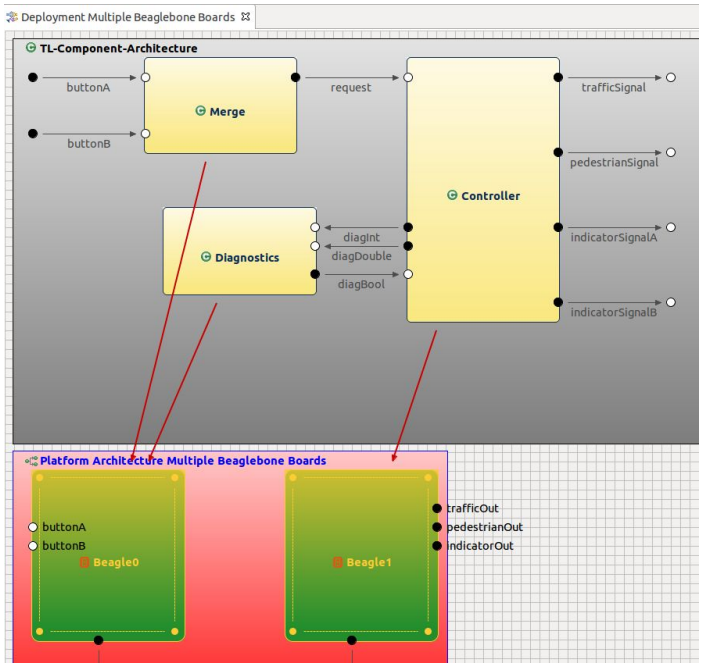
AutoFOCUS 3 Deployment Model

fortiss

Deployment Model describes

- Allocation of Components to Execution Units
- Allocation of Input Ports to Sensors and/or Transmitted Signals *[not shown]*
- Allocation of Output Ports to Actuators and/or Transmitted Signals *[not shown]*

- Deployment Code Generator synthesizes the complete system code including configuration files and transmission catalogs

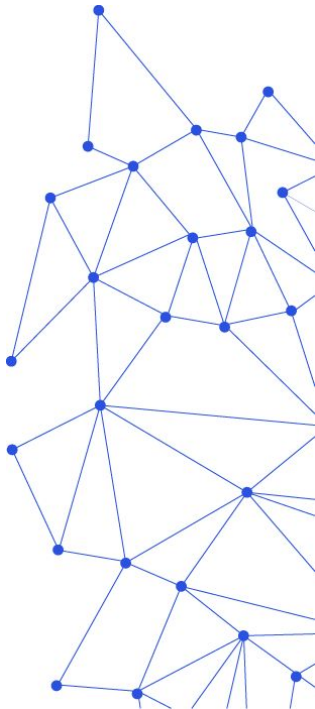


Results of the Experiment (continued)

fortiss

User interface aspects

- The Papyrus-RT profile does not constrain, but adds elements
- Makes the user interface of Papyrus-RT counter-intuitive for beginners:
  - Hard to assess the impact of UI elements on the semantics
    (we did a trial-and-error reverse engineering of model attributes effects on code generation)
  - Redundant pieces of information
    (port cardinality must match number of outgoing channels)
  - Missing constraint checkers (same example)
  - Relevant model elements are sometimes practically hidden
    (very hard to access, e.g., through nested dialogs)
  - Profile-specific modelling subset still presented in a general way
    (e.g., information scattered in nested dialogs could be "compressed")
- Documentation hard to find and easily outdated (same in AF3 :-)



Result of the experiment - Step back

fortiss

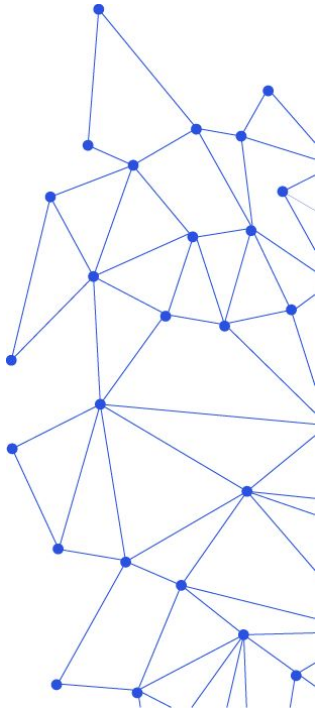
Is profiling (as of today) the good solution for such a use case?

I.e. for a use case where:

- Execution semantics is involved
- The gap between UML and the domain is big

  → hinders usability a lot

⇒ In this case, domain-specific is not just a graphic matter



Questions to the community

fortiss

Customization:

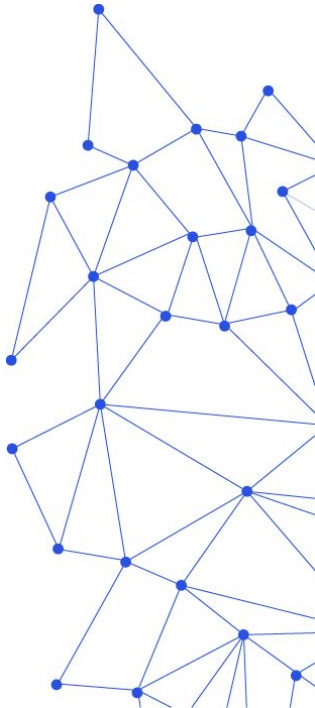
- What is the latest state/plan of customization possibilities?
- Where can we find pointers to it?

Simulation/Code generation:

- Plans in Papyrus-RT for simulation?
- Code generation to C instead of C++?
- Code generation with/connected to Moka?

Deployment:

- Plans to do something similar to AF3 deployments?



Future plans

fortiss

New experiment with another student:

- Make use of the latest Papyrus improvements w.r.t. customization
- Make use of the latest Moka improvements w.r.t. state automaton simulation
- Modelling a bigger system with Papyrus-RT

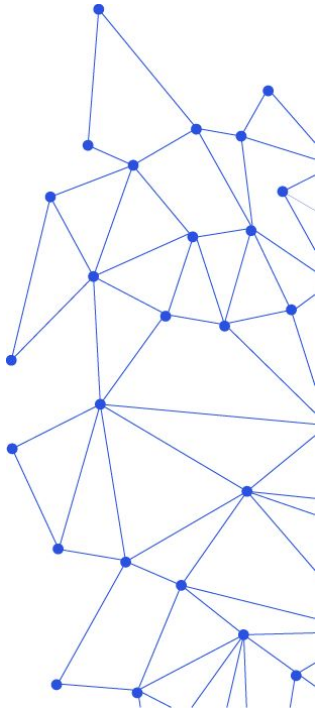
Long-term:

- Try to export some AF3 functionality to Papyrus-RT/Papyrus-Moka?
- Considering making an Eclipse Project out of AF3

  → Reason: increasing collaborations with industrial partners

  (Continental, BMW, Liebherr Aerospace, Diehl Aerospace, ...)

  → Could facilitate collaborations with the Papyrus ecosystem?



fortiss

Thanks!

Florian Hölzl, Vincent Aravantinos

{hoelzl, aravantinos}@fortiss.org

http://af3.fortiss.org/