Queen's School of Computing
Queen's University
Kingston, Ontario, Canada

Queen's University

Sum Ergo Computo
I am therefore I compute

Supporting the Model-Driven Development of Real-time Embedded Systems with Simulation and Animation via Highly Customizable Code Generation

Nondini Das, Suchita Ganesan, Leo Jweda,
Mojtaba Bagherzadeh, Reza Ahmadi, Nicolas Hili, Juergen Dingel
{ndas, ganesan, juwaidah, mojtaba, ahmadi, hili, dingel}@cs.queensu.ca
School of Computing, Queen's University,
Kingston, Ontario, Canada

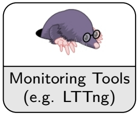
Progress Report Meeting, École Polytechnique de Montréal, May 2016

Outline

Introduction and Motivation

Monitoring Tools
(e.g. LTTng)

Introduction and Motivation



From Runtime Model Monitoring...

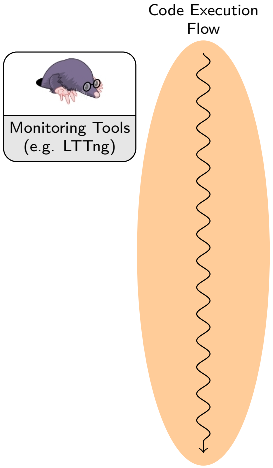
Introduction and Motivation

Monitoring Tools (e.g. LTTng)

From Runtime Model Monitoring...

✓ Timing / Resource Constraint Violation

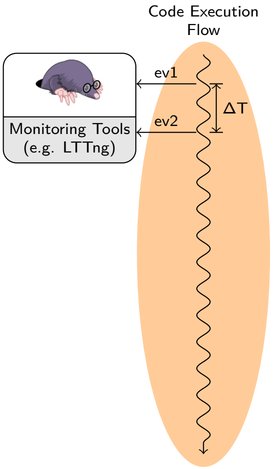
Introduction and Motivation



From Runtime Model Monitoring. . .

✓ Timing / Resource Constraint Violation
✓ Code-driven

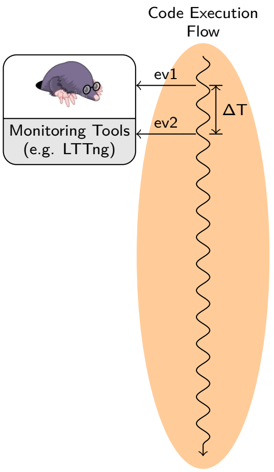
Introduction and Motivation



From Runtime Model Monitoring...

- ✓ Timing / Resource Constraint Violation
- ✓ Code-driven
- ✓ LTTng acts as an *observer*:
  - Listens for specific events
  - Does not disrupt the execution flow

Introduction and Motivation



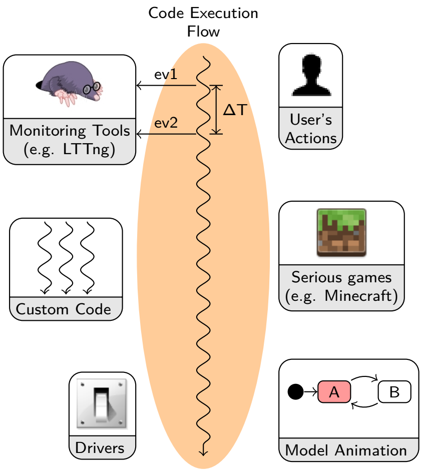
From Runtime Model Monitoring...

- ✓ Timing / Resource Constraint Violation
- ✓ Code-driven
- ✓ LTTng acts as an *observer*:
  - ▶ Listens for specific events
  - ▶ Does not disrupt the execution flow

▼

...to a more general vision

Introduction and Motivation



From Runtime Model Monitoring. . .

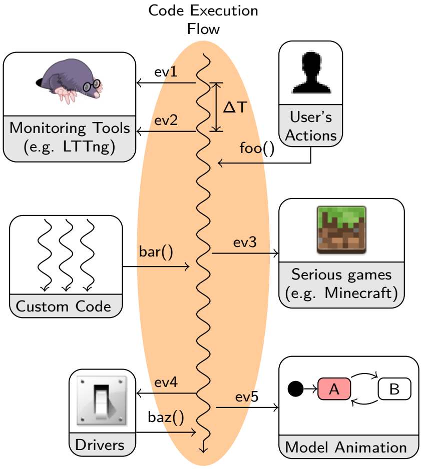
- ✓ Timing / Resource Constraint Violation
- ✓ Code-driven
- ✓ LTTng acts as an *observer*:
  - Listens for specific events
  - Does not disrupt the execution flow

▼

. . . to a more general vision

☞ Pluggable components:
- Animation and Interaction
- Debugging

Introduction and Motivation



From Runtime Model Monitoring...

✓ Timing / Resource Constraint Violation
✓ Code-driven
✓ LTTng acts as an *observer*:
- Listens for specific events
- Does not disrupt the execution flow

▼

...to a more general vision

☞ Pluggable components:
- Animation and Interaction
- Debugging

☞ Consumer / Producer of events

Infrastructure Overview

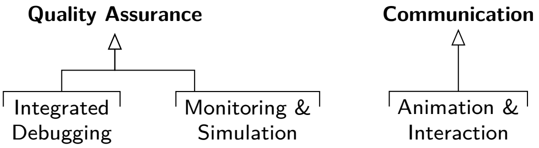
Integrated Debugging

Monitoring & Simulation

Animation & Interaction

Three activities...

Infrastructure Overview

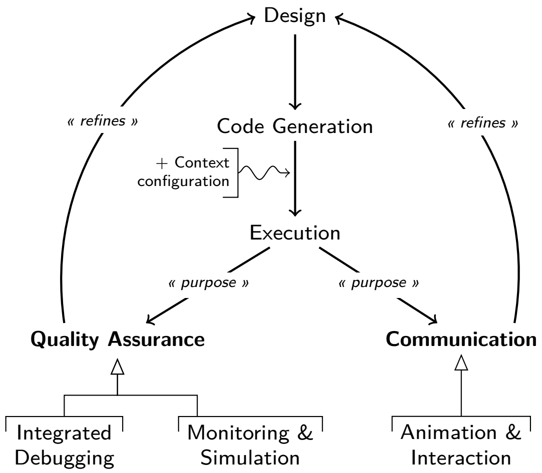


. . . For two purposes. . .

▲

Three activities. . .

Infrastructure Overview



. . . to support model-driven design

- Allows for continous development
- Driven by the code generation
- Highly Configurable

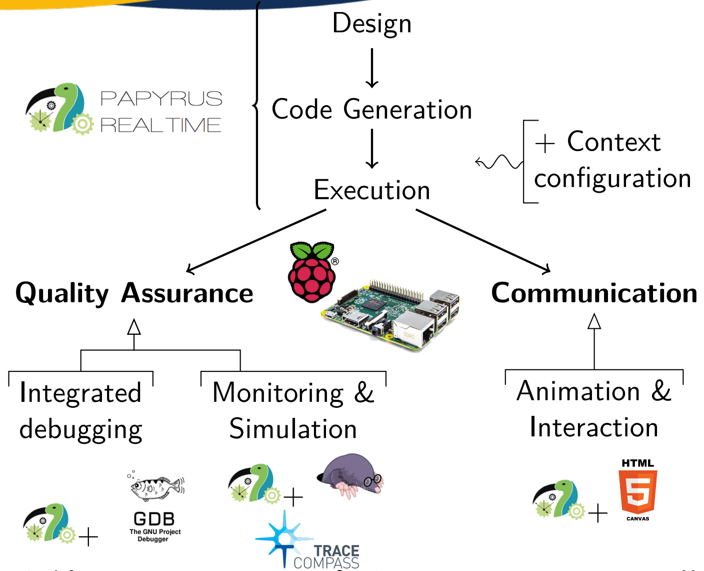
▲

. . . For two purposes. . .

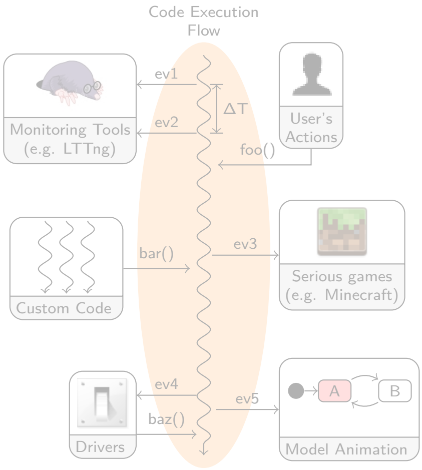
▲

Three activities. . .

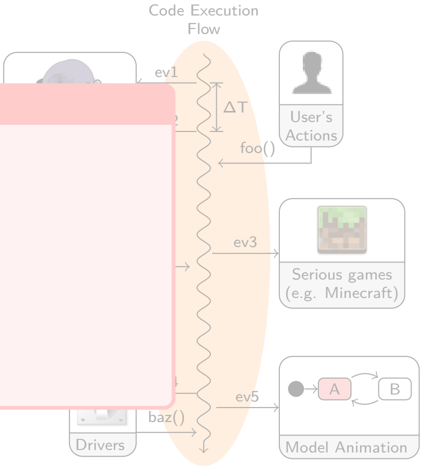
Open Source tool Support



Infrastructure's Challenges

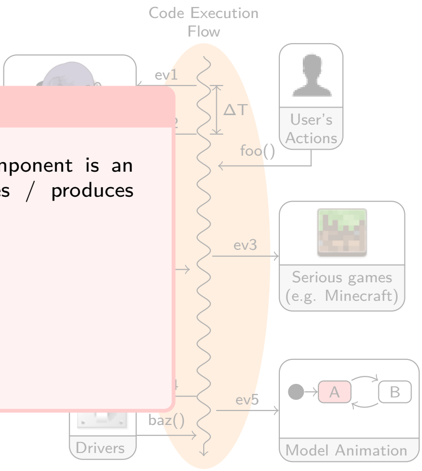


Infrastructure's Challenges



Challenges to address

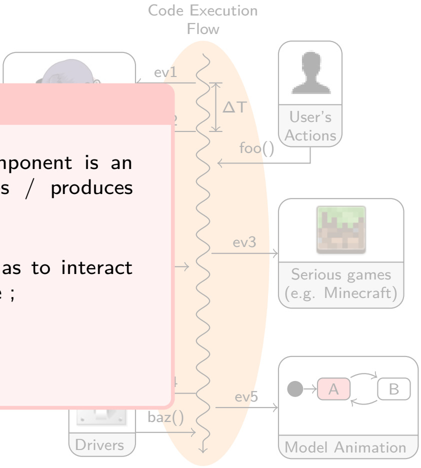
Infrastructure's Challenges



Challenges to address

☞ Each pluggable component is an observer that consumes / produces specific events ;

Infrastructure's Challenges

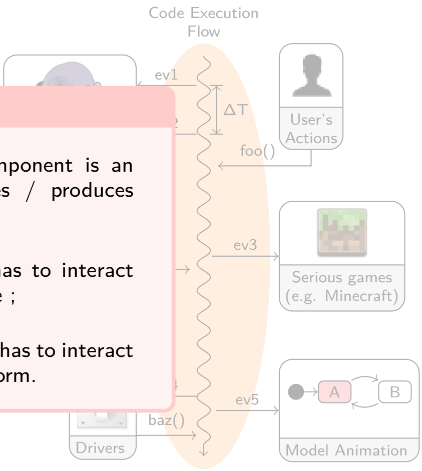


Challenges to address

☞ Each pluggable component is an observer that consumes / produces specific events ;

☞ Each component has to interact with the generated code ;

Infrastructure's Challenges



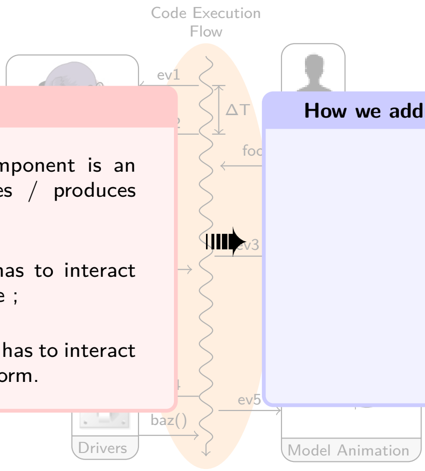
Challenges to address

☞ Each pluggable component is an observer that consumes / produces specific events ;

☞ Each component has to interact with the generated code ;

☞ The generated code has to interact with the hardware platform.

Infrastructure's Challenges



Challenges to address

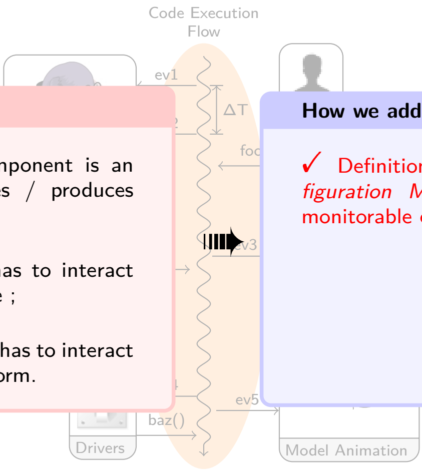
☞ Each pluggable component is an observer that consumes / produces specific events ;

☞ Each component has to interact with the generated code ;

☞ The generated code has to interact with the hardware platform.

How we addressed them

Infrastructure's Challenges



Challenges to address

☞ Each pluggable component is an observer that consumes / produces specific events ;

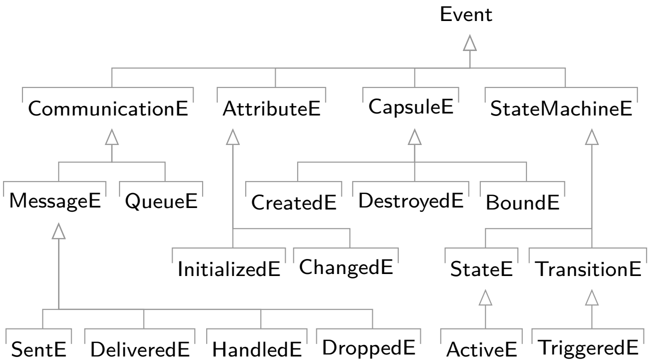
☞ Each component has to interact with the generated code ;

☞ The generated code has to interact with the hardware platform.

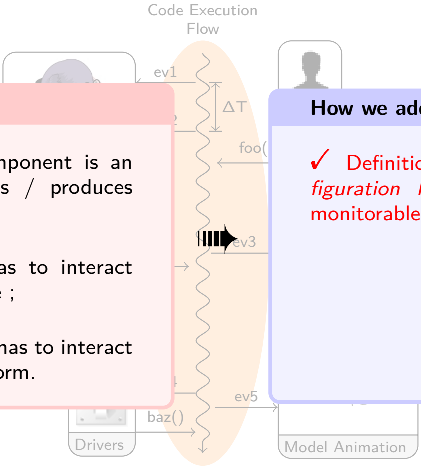
How we addressed them

✓ Definition of a *Context Configuration Model* that lists all monitorable events ;

Configuring the infrastructure



Infrastructure's Challenges



Challenges to address

☞ Each pluggable component is an observer that consumes / produces specific events ;

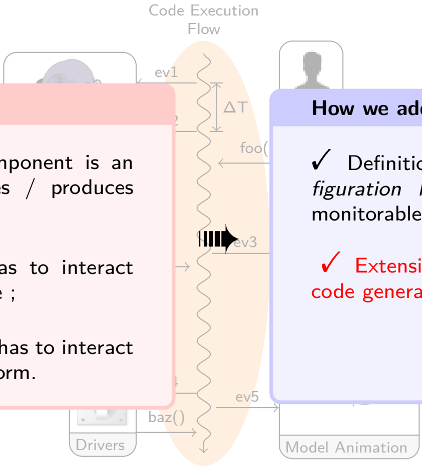
☞ Each component has to interact with the generated code ;

☞ The generated code has to interact with the hardware platform.

How we addressed them

✓ Definition of a *Context Configuration Model* that lists all monitorable events ;

Infrastructure's Challenges



Challenges to address

☞ Each pluggable component is an observer that consumes / produces specific events ;

☞ Each component has to interact with the generated code ;

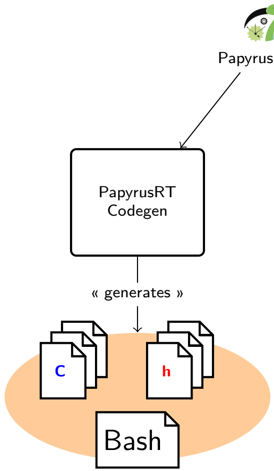
☞ The generated code has to interact with the hardware platform.

How we addressed them

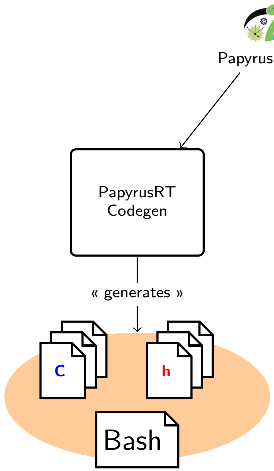
✓ Definition of a *Context Configuration Model* that lists all monitorable events ;

✓ Extension of the PapyrusRT code generator ;

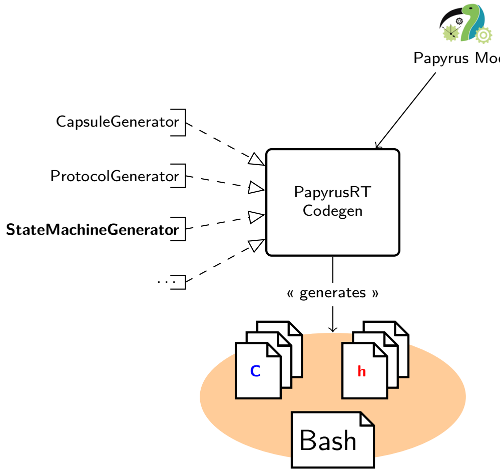
Extending the PapyrusRT Code Generator



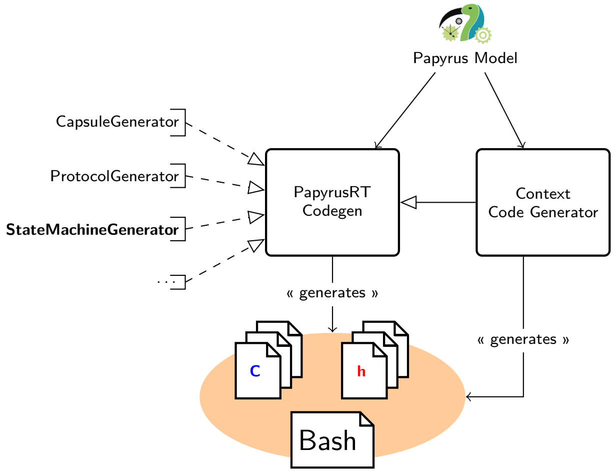
Extending the PapyrusRT Code Generator



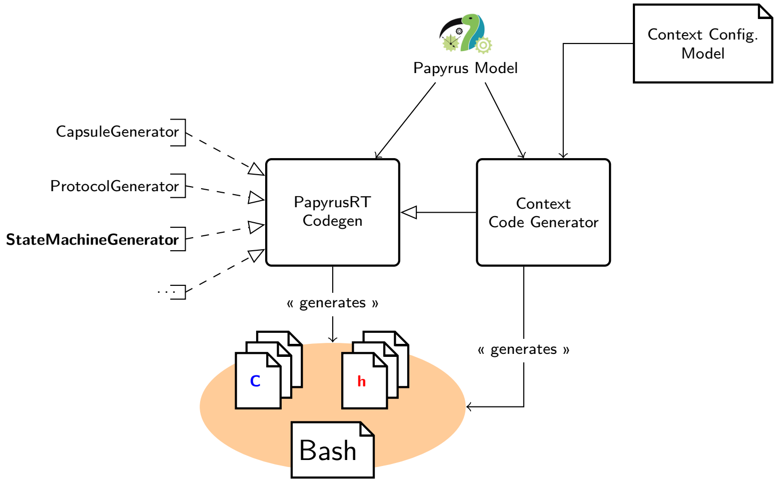
Extending the PapyrusRT Code Generator



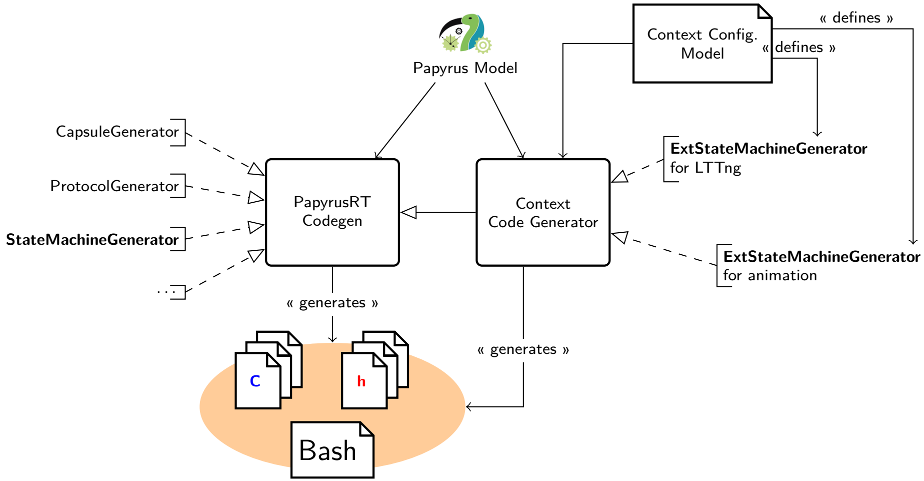
Extending the PapyrusRT Code Generator



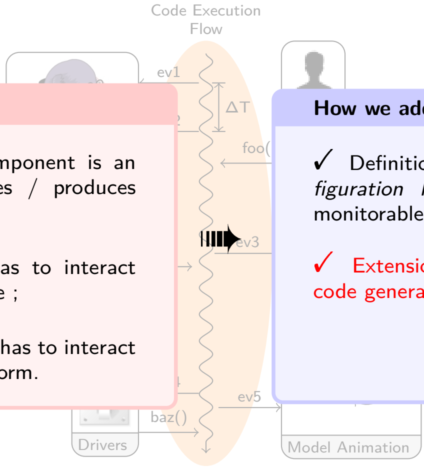
Extending the PapyrusRT Code Generator



Extending the PapyrusRT Code Generator



Infrastructure's Challenges



Challenges to address

☞ Each pluggable component is an observer that consumes / produces specific events ;

☞ Each component has to interact with the generated code ;

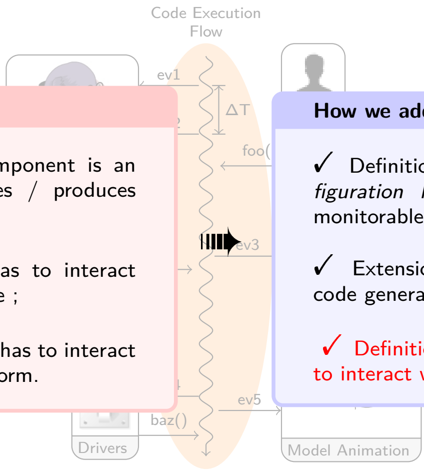
☞ The generated code has to interact with the hardware platform.

How we addressed them

✓ Definition of a *Context Configuration Model* that lists all monitorable events ;

✓ Extension of the PapyrusRT code generator ;

Infrastructure's Challenges



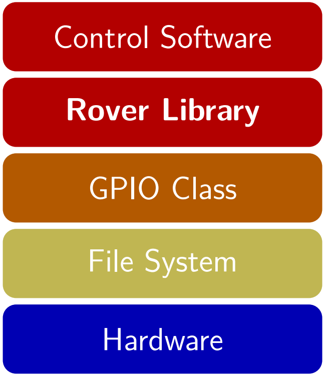
Challenges to address

- ☞ Each pluggable component is an observer that consumes / produces specific events ;
- ☞ Each component has to interact with the generated code ;
- ☞ The generated code has to interact with the hardware platform.

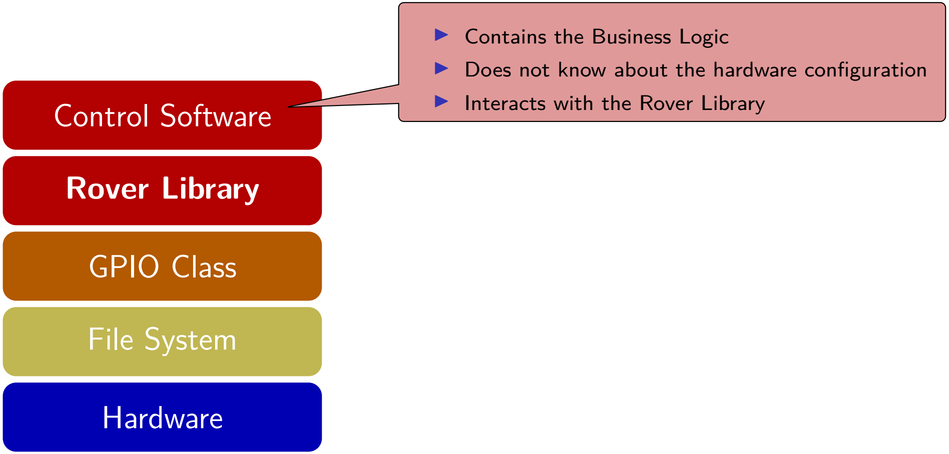
How we addressed them

- ✓ Definition of a *Context Configuration Model* that lists all monitorable events ;
- ✓ Extension of the PapyrusRT code generator ;
- ✓ Definition of a Rover Library to interact with the hardware.

Definition of the Rover Library

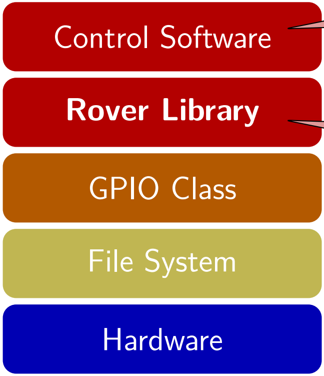


Definition of the Rover Library



- Contains the Business Logic
- Does not know about the hardware configuration
- Interacts with the Rover Library

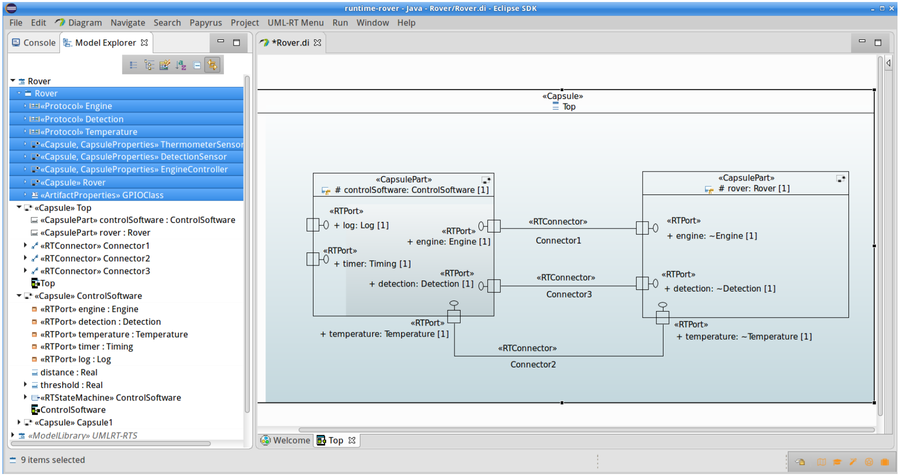
Definition of the Rover Library



- Contains the Business Logic
- Does not know about the hardware configuration
- Interacts with the Rover Library

- Makes the glue with the Hardware
- Defines the protocols the Business Logic will have to interact with
- Specific to a design configuration

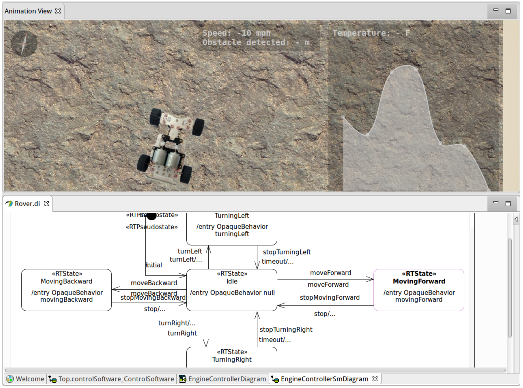
Definition of the Rover Library



- Rover
  - Rover
    - «Protocol» Engine
    - «Protocol» Detection
    - «Protocol» Temperature
    - «Capsule, CapsuleProperties» ThermometerSensor
    - «Capsule, CapsuleProperties» DetectionSensor
    - «Capsule, CapsuleProperties» EngineController
    - «Capsule» Rover
    - «ArtifactProperties» GPIOClass
  - «Capsule» Top
    - «CapsulePart» controlSoftware : ControlSoftware
    - «CapsulePart» rover : Rover
    - «RTConnector» Connector1
    - «RTConnector» Connector2
    - «RTConnector» Connector3
    - Top
  - «Capsule» ControlSoftware
    - «RTPort» engine : Engine
    - «RTPort» detection : Detection
    - «RTPort» temperature : Temperature
    - «RTPort» timer : Timing
    - «RTPort» log : Log
    - distance : Real
    - threshold : Real
    - «RTStateMachine» ControlSoftware
    - ControlSoftware
  - «Capsule» Capsule1
- *«ModelLibrary» UMLRT-RTS*

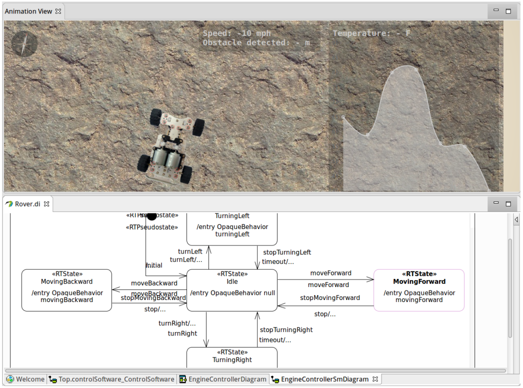
9 items selected

Animating the Model



Web based animation & State Machine Live Monitoring

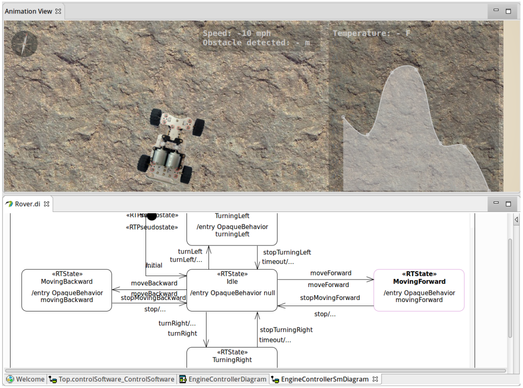
Animating the Model



Web based animation & State Machine Live Monitoring

✓ Animation of the Rover Model

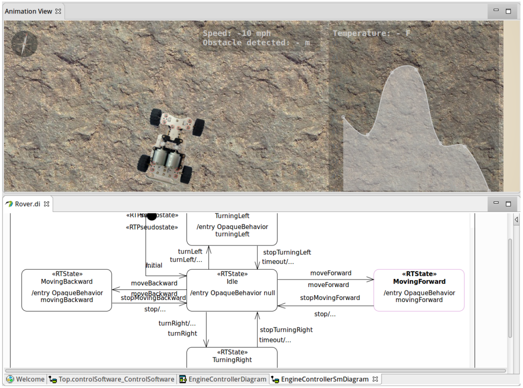
Animating the Model



Web based animation & State Machine Live Monitoring

✓ Animation of the Rover Model
✓ Code-driven (different from Moka)

Animating the Model



Web based animation & State Machine Live Monitoring

✓ Animation of the Rover Model
✓ Code-driven (different from Moka)
✓ Works as an observer:

- Bi-directional socket communication with the C++ code
- Listen all events (state changes, transitions fired)
- Would at last interact with the code execution flow (not supported yet)

What's Next ?

Vision

☞ Improve the different parts of the infrastructure, especially the code generator to allow for several configurations to be used at the same time ;

☞ Define different libraries for different models ;

☞ Some bugs have to be corrected in Papyrus / PapyrusRT (e.g. Internal transition with effects, graphical glitches since the new Eclipse version, etc.).

Animation & Interaction

☞ Allow the user to interact with the model using the animation view ;

☞ Implement other animation engines (2D/3D, Unity, etc.) ;

☞ Propose a creation tool to automatically create animation views to animate and interact with the model ;

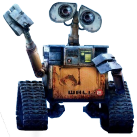
☞ Look at Moka to see if it can be used to simulate the state machine execution ;

Queen's SCHOOL OF Computing
Queen's University
Kingston, Ontario, Canada

Queen's UNIVERSITY

Sum Ergo Computo
I am therefore I compute

Thank You !



Questions ?