PAPYRUS TOOL SUPPORT FOR FMI TUTORIAL

MODPROD 2016
Linköping, Sweden, February 7-8, 2017

ITEA European project

Sahar GUERMAZI, Sébastien REVOL, Arnaud CUCCURU, Saadia DHOUIB, Jérémie TATIBOUET, Sébastien GERARD
CEA LIST / DILS / LISE
www.cea.fr
CEA is a major actor in research and innovation.

- **16,000 people**
- **16 centers in France**
- **Budget: 4.3 billion €**
- **1,600 patents**
- **4,000 publications/year**
- **150 startups created since 1984**
The LISE labs in a nutshell

Correct-by-construction design of safe CPS

~50 persons

- 30 permanent members + 17 non-permanent members including PhD students, post-docs ... (2015)

Main research concerns

- Modeling Language Engineering
- Model-based Formal Analysis (e.g., auto gen. of tests)
- Run-time Formal Verification and Monitoring
- Model-based Simulation
- Model-based Security & Safety Engineering
- Archi. Exploration, Configuration & Deployment
- Process, Requirement and Variant Engineering
Papyrus is the official open-source Eclipse UML2 modeling tool:  
www.eclipse.org/papyrus

- Papyrus provides a complete graphical editor for both UML and SysML standards based on the MDT::UML2 component for its repository.
- Papyrus addresses the two key features expected from a UML2 graphical editor: modeling and profiling.
- Papyrus is highly customizable and extensible enabling DSML definitions based on standard UML profiles!
- Papyrus provides a support to MARTE 1.1 (including a rich text editor for VSL).
• Synergy of two complementary standards for Complex system modeling and simulation

• FMI (Functional Mockup Interface)
  • Emerging standard for co-simulation
  • Enables multiple compliant modeling and simulation tools to interoperate
  • Particularly interesting for designing CPS (Cyber Physical Systems)

• UML in the FMI eco-system
  • UML (and its variants) can be used to design parts of CPS
    • E.g., the high-level control logic of an embedded software
  • Would be nice to have the possibility to assess the relevance of the UML-based parts with respect to their (simulated) environment
    • Scenario exploration, early error detections.

• Papyrus now provides FMI tool support
  • Based on Moka, the Papyrus module for model execution
  • Early results, still in incubation phase
OVERVIEW OF PAPYRUS TOOL SUPPORT FOR FMI

FMU (model + solver) exported from MATLAB/SIMULINK, Dymola, OpenModelica, ...

Moka FMU

import

export

Simulation outputs. Can be opened in new XY diagram kind provided by Papyrus

Composite structures assembly

Master algorithm
• Papyrus MOKA overview
• Short Reminder on FMI/FMU

• Papyrus as FMI co-simulation Master :
  • FMU modelling in Papyrus
  • Import of a simple FMU in Papyrus
  • Run a simple simulation
  • Visualize results

• Papyrus as FMU provider :
  • Reminder on OMG standards for Executable Modeling
  • Study and debug a simple UML-based FMU model
  • Export FMU
  • Analysis of generated FMU

• Integration :
  • Integration and co-simulation of the newly exported FMU
• **Papyrus is based on Eclipse**
  • Most common platforms are supported (Windows/Linux/Mac…)
  • Requires **JAVA 8**

• **Papyrus for FMI cosimulation**
  • JAVA imposes restrictions on 32bits/64bits DLL loading
  • DLL should have the same architecture as the running JVM
  • $\rightarrow$ 64 bits JVM can only load 64 bits FMUs (and 64 bits eclipse distributions)
  • Running mix of 32 bits/64 bits FMUs is not possible
  • But running 32 bits FMUs on a 64 bits machine is possible
    → Install 32 bits JVM and Eclipse/Papyrus distribution

• **Papyrus as FMU provider**:
  • Generated FMUs can run on Linux 32/64 bits and Windows 64 bits
  • Other architectures may be supported on-demand
  • Generated FMUs may requires a JVM on the running machine

• **For this tutorial**
  • We only provide a Windows 64 bits Papyrus distribution and FMU example
TUTORIAL SETUP

- **Papyrus.zip**: papyrus distribution to unzip on your machine
  - Includes pre-installed MOKA FMI plugins
  - Run Papyrus.exe after unzip
  - Select a workspace where your projects will be saved
- **TutorialProjects.zip**: zipped projects
  - No need to unzip
  - In eclipse: File → Import … → Existing project … → select archive
PART I

OVERVIEW OF MOKA, THE PAPYRUS MODULE FOR MODEL EXECUTION
MOKA: OVERVIEW

• **Papyrus module for model execution**
  • Help designers to understand/orient their design choices
  • Basis for a straightforward, simulation-driven design process:
    . (Model / Execute / Observe / Refine)+
  • Front-end for integration of simulation tools and techniques

• **Model Debugging capabilities**
  • Control (start/stop, suspend/resume, breakpoints)
  • Observation (diagram animation, variables, threads)

• **Complies with standard OMG semantics of UML**
  • Implements the fUML and PSCS execution models (PSSM coming)
  • Experimental tool support for Alf, the standard textual notation of fUML

• **Flexible/extendible**
  • New execution engines can be plugged (to support multiple semantics and UML profiles)
  • Extension points to inject control/execution model libraries (to trigger the execution of external functions and procedures directly from a UML model)
• Controlling and Observing executions

CONNECTION WITH THE ECLIPSE DEBUG FRAMEWORK

- Execution control panel
- Breakpoint Control Panel / Runtime (Variables) state panel
- Threads / Stack frames view
- Emphasis on the element associated with the stack frame
• Multiple execution engines can be registered
MOKA: OVERVIEW

- **Papyrus plug-in for model execution**
  - Help designers to understand/orient their design choices
  - Basis for a straightforward, simulation-driven design process:
    - (Model / Execute / Observe / Refine)+
  - Front-end for integration of simulation tools and techniques

- **Model Debugging capabilities**
  - Control (start/stop, suspend/resume, breakpoints)
  - Observation (diagram animation, variables, threads)

- **Complies with standard OMG semantics of UML**
  - Implements the fUML and PSCS execution models (PSSM coming)
  - (Tool support for Alf, the standard textual notation of fUML)

- **Flexible/extendible**
  - New execution engines can be plugged (to support multiple semantics and UML profiles)
  - Extension points to inject control/execution model libraries (to trigger the execution of external functions and procedures directly from a UML model)

- **NEW: Support for FMI Co-Simulation standard**
  - Export of FMUs from executable UML models
  - Ability to import and assemble FMUs, co-simulate them with the built-in Moka master, and visualize simulation traces on XY charts.
FUNCTIONAL MOCK-UP INTERFACE (FMI)

Allows to export each executable model as a standalone unit (FMU)

- An FMU has to implement a standard binary interface as a shared library (.dll/.so)
  - Set Inputs
  - Get outputs
  - Do Step (stepSize)

The simulation Master synchronizes and orchestrates the FMUs
PART II

PAPYRUS AS FMI CO-SIMULATION MASTER
FMU loading/saving integrated in Eclipse Modeling Framework

- FMUs are considered as « Models »
- Automatically unzip/zip FMU archive
- FMUs can be used as inputs or outputs of model transformation techniques
EXERCISE 1: OPEN AN FMU IN ECLIPSE

FMUs can be edited with default Ecore Reflective editor
EXERCISE 1: OPEN AN FMU IN ECLIPSE

1. Complete modelDescription.xml editor.
2. Additionnal files are listed.
EXERCISE 1: ADD AN ARCHIVE IN RESOURCE FOLDER OF FMU...
EXERCISE 1: ADD AN ARCHIVE IN RESOURCE FOLDER OF FMU...

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive File</td>
<td>C:\Prog\papyrus-neon-2.0.2-tutorial\Papyrus-FMI-Tutorial\TutorialProjects.zip</td>
</tr>
<tr>
<td>Name</td>
<td>test</td>
</tr>
</tbody>
</table>
Result after saving: open the FMU file as an Archive (out of eclipse)

- A new folder named « test » is created inside FMU resource folder
- It contains the contents of the archive
Papyrus first class citizen are UML model elements
- We must provide a mechanism to represent FMUs as UML model elements
- This is the purpose of the FMI profile

A profile allows to extend standard UML concepts with domain specific attributes

FMI profile :
- Adds to UML elements FMI specific concepts
- Not a full one to one translation : only useful concepts for UML display/handling
- Includes a direct link to in-memory original FMU model

FMU import in Papyrus
- model transformation from FMU metamodel to UML + FMI profile

FMU generation
- model transformation from UML + FMI profile to FMU metamodel
- generation according Moka computation mechanism
  - Ex : only discrete variables
FMI PROFILE OVERVIEW

- FMUs are represented as a special kind of Class
Scalar Variables are represented as a special kind of Class attributes
THE FMI MASTER FUNCTIONNALITY

• **Key features:**
  
  • Ability to import FMUs from FMI 2.0 compliant tools
  
  • Definition of the co-simulation graphs (i.e., assembly of FMUs + configuration of simulation runs)
  
  • Master algorithm specified by an executable UML model, along with a dedicated model library
    - Fixed step size, no usage of rollbacks, but we have some plans to go further…
  
  • Visualization of co-simulation results with XY charts
Create a new Papyrus project
Select UML -> next -> name the project and the model file

**EXERCISE 2 : PAPYRUS FMU IMPORT**
EXERCISE 2 : PAPYRUS FMU IMPORT

Select FMI simulator model template and finish

→ predefined « ready to run » Simulator model
Open Simulator architecture diagram

- Private editor page layout
- Remember last active page

**Languages:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>UML</td>
<td>2.5.0</td>
</tr>
</tbody>
</table>

**Related Views**

- Properties View
- Model Explorer
- Model Validation

**Notation Views**

- Simulator Architecture
- Simulator
- Start Simulation
- startSimulation
EXERCISE 2: PAPYRUS FMU IMPORT

Empty composite structure diagram

CS_Graph stereotype allows to configure simulator settings

Exercise: set simulation step to 0.01
EXERCISE 2 : PAPYRUS FMU IMPORT

From model explorer root : right click, Moka, FMI, Import FMU for co-simulation
Model Library is useful to group several FMUs

Select generator.fmu from workspace/PapyrusFMITutorial/input
We obtain a new Class named « generator » with an output port called « Out1 ».
EXERCISE 2 : PAPYRUS FMU IMPORT

Drag and drop the class into the diagram, and select FMU-specific Papyrus drop strategy

We get a new FMU instance
EXERCISE 2 : PAPYRUS FMU IMPORT

Create a new Moka Run configuration

UML Model:
platform:/resource/Cosimulation/cosimulation.uml

Element to be executed:
(Activity) RootElement::Simulator::startSimulation

Execution Engine (if no selection, the default engine is used):
org.eclipse.papyrus.moka.fuml.cosimulation
If everything is ok....

A new simulation trace appears in project explorer (after refresh, press F5)
Import CSV into Papyrus Model

A new «DataSource» appears in model explorer

- `DataSource` fmi_simulation_trace
  - `time` : `Real`
  - `generator1.Out1` : `Real`
EXERCISE 2 : PAPYRUS FMU IMPORT

Create a new graph from the data source

Select the traces to display
XY graphes are new kinds of Papyrus Diagrams
PART III

PAPYRUS AS FMU DESIGNER
**EXECUTABLE UML OMG SPECIFICATIONS**

Alf (Action Language for fUML):
- Textual surface notation for the fUML subset
1. Class diagram (~ BDD)

- Instantiation of an active class implies starting of its behavior

AcceptEventActions enable to specify reactive behaviors

2. Composite structure diagram (~ IBD)

- Instantiation of a composite structure implies instantiation of its constituents

SendSignalAction enable to specify asynchronous communications, which will flow through ports and connectors

Event dispatching occurs at Run To Completion (RTC) steps
An FMU is an fUML Active Object with a classifier behavior described with an Activity diagram (state-machine support ongoing).

```uml
classifier CS_FMU
  «Port» {direction=in} x: Integer
  «Port» {direction=out} y: Boolean
  compare( in threshold: Integer): Boolean

change on x
  After 0.5

self
  result

2
  result

2
  threshold

setY
  object
  value
  result

callCompare
  target
  result

Change event on x (x = 3)
Time event
```

Values observed by the master
Open PapyrusFMITutorial/input/SimpleFMU UML model

FMU structure can be described in a Composite Structure diagram
**FMU Class key aspects:**

- Should be an active class
- Should be stereotyped with `FMIPreofile::CS_FMU` stereotype
  - No need to feel stereotypes attributes, they will be filled by Moka at export time
- Can own several behaviors
  - Only one should be referenced as Classifier behavior
  - Other behaviors can be called from the Classifier behavior
• **FMU Port key aspects**

  • Should be stereotyped with `FMIProfile::Port stereotype`
    - `direction` (in/out) and `valueReference` (unique ID) should be specified
    - Other attributes will be generated at FMU export

  • **Ports should have a type**
    - Only UML standard primitive types (Integer, Boolean, String, Real)

  • **Ports should have a default value**
    - Only UML primitive types values (LiteralInteger, LiteralBoolean, LiteralString, LiteralReal)

  • **Multiplicity must be set to 1**
**Simple FMU behavior**

- Infinite loop
- Waiting on input changes
- And comparing the input versus 0
- Write true on output if input greater or equals to 0
Switch to Moka Debug Perspective
• Create new Moka run configuration

Select the FMU Class (not the activity!)

Select the FMU debug engine
EXERCISE 3 : PAPYRUS FMU DESIGN

- **FMU controller allows to:**
  - Change inputs
  - Configure size and run FMI steps
PART IV

FMU GENERATION
• **Architecture of exported FMUs**

  - **API FMI**
    - FMU Wrapper (impl. C++)
    - Provided interface
    - Synchronous communication
    - Execution

  - **Moka FMU**
    - FMU Wrapper (impl. Java)
    - FMU metadata
    - .uml model of the FMU

  - **Moka RCP**
    - Moka Engine

• **No code generation**
  - Only the modelDescription.xml is generated
  - The generated FMU includes the UML model a minimal Moka interpreter
  - And a generic DLL implementing the FMI interface and interacting with Moka
• From the FMU Class: right click, Moka, FMI, Export FMU for co-simulation
• Provide an FMU name (FMI model identifier)
• Select the target directory
• Select the target platform
  • Currently only win64, Linux32 and Linux64 are supported
  • Other platforms can be supported on demand
• Optionally: a JRE can be embedded in the FMU
  • Can be a minimal JRE (example Linux Embedded)
  • Useful if target platform doesn’t have a JRE installed
• FMU structure and modelDescription.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<fmiModelDescription fmiVersion="2.0" generationDateTime="2017-02-05T22:29:56.920+01:00" generationTool="Moka FMU exporter" guid="">
  <CoSimulation canBeInstantiatedOnlyOncePerProcess="true" canGetAndSetFMUState="false" canHandleVariableCommunicationStepSize="true">
    <ModelVariables>
      <ScalarVariable causality="input" initial="approx" name="in" valueReference="0" variability="discrete">
        <Real start="-1.0"/>
      </ScalarVariable>
      <ScalarVariable causality="output" initial="exact" name="out" valueReference="1" variability="discrete">
        <Boolean start="false"/>
      </ScalarVariable>
    </ModelVariables>
    <ModelStructure>
      <Outputs>
        <Unknown index="2"/>
      </Outputs>
    </ModelStructure>
  </fmiModelDescription>
```
EXERCISE 4 : PAPYRUS FMU GENERATION

- Import generated FMU in first co-simulation model (cf exercise 2)
  - Connect generator output to SimpleFMU input
EXERCISE 4 : PAPYRUS FMU GENERATION

- Re-run simulation

- Re-import new CSV
• **On Master Side:**
  - FMU parameters configuration (almost there!)
  - Simulation debug (breakpoints at time, at port value, step by step simulation, runtime values visualization…)
  - Delegation to external master (Cosim or Model exchange)
  - Improve logging interface (select values to be logged, direct graph generation without CSV import)
  - .mat file simulation trace support

• **On Slave Side:**
  - State machine support (almost there!)
  - Rollback support
  - Performance improvement
  - New target platform support

• **Part of these features will be developed in OpenCPS ITEA project**
GETTING STARTED WITH MOKA:
HTTPS://WIKI.ECLIPSE.ORG/PAPYRUS/USERGUIDE/MODELEXECUTION

VIDEO TUTORIALS :
HTTPS://WWW.YOUTUBE.COM/CHANNEL/UCXYPOBLZC_RKLS7_K2DTWYA

Acknowledgments to the LISE team for their direct and indirect contributions to this presentation.