UNIVERSITY OF TWENTE.

Simulation and Visualization Tool Design for Robot Software

Zhou Lu, Tjalling Ran and Jan Broenink

Robotics and Mechatronics,
University of Twente

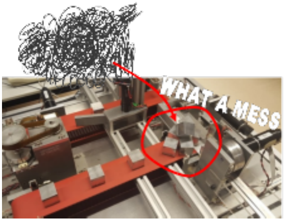
August 22, 2016

Outline

- Introduction
- Design of Simulation
- Visualizing Simulation Results
- Results
- Conclusions and Recommendations

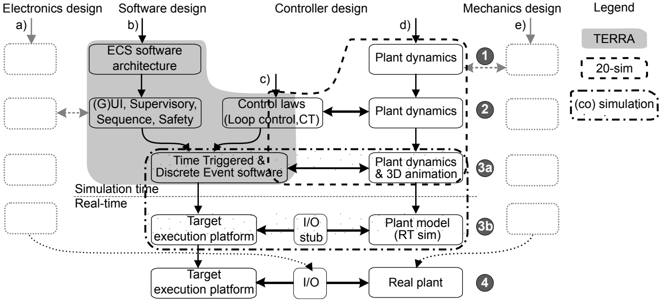
Introduction - Context

- Cyber-Physical Systems (CPS) co-design: why challenging?
  - Combine multiple different engineering disciplines/domains
  - Seamless interaction with physical environments
  - Concurrency is intrinsically presented in CPS
  - Most CPS are safety-critical



Introduction - Related Work

- Design CPS using Model-Driven Design (MDD)
  - Models can be formalized and checked
  - Modelled systems can be tested and simulated off-line
  - Ease tedious and error-prone concurrent software development



Introduction - Problem Statements and Motivation

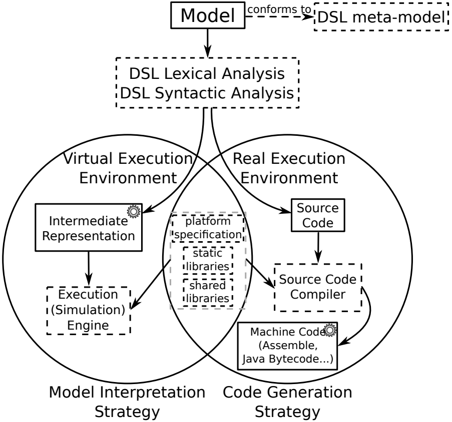
- Iterative and incremental design and development in MDD
  - Sufficient verification and/or validation of models are required
- MDD in CPS: different domain models are involved
  - Discrete-Event and Continuous-Time domains
- Co-simulation is needed to support co-design
- Current infrastructure: TERRA
  - Does not provide sufficient simulation nor visualization facilities

Design of Simulation - Obtain Executable Models

- Executable models
  - Executability: depends more on execution tools
  - Execution tools: depend on assessment requirements
- Required assessments in CPS
  - Process execution order
  - Results of algorithms

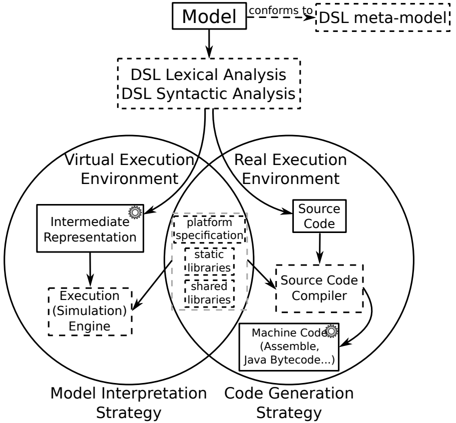
Design of Simulation - Obtain Executable Models

- Executable models
  - Executability: depends more on execution tools
  - Execution tools: depend on assessment requirements
- Required assessments in CPS
  - Process execution order
  - Results of algorithms
- Two strategies to obtain executable models
  - Model interpretation
  - Code generation



Design of Simulation - Analysis

- Model Interpretation
  - Relies on the existence of a Virtual Execution Environment (VEE)
  - Interpretation can be done dynamically
- Code generation
  - Uses M2T transformation to generate lower-level system representation
  - Platform-dependent



Design of Simulation - Analysis

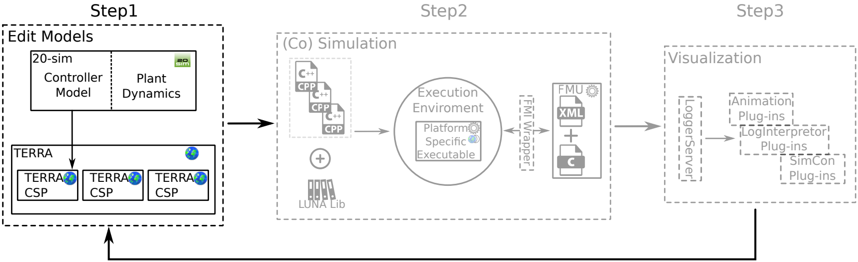
- From a practical perspective: code generation is preferred
  - Control algorithms generated from 20-sim must be taken into account
  - TERRA is able to generate C++ code
  - Model interpretation will just be simulation without code implementation

Design of Simulation - Analysis

- Coupling strategies
  - Loose-Coupling Execution
    - Generate source code from different models
    - Generate APIs for interacting purpose
    - Execution coordinator
  - Tight-Coupling Execution
    - Integrate different models by using M2M transformations
    - Generate code from an integrated model

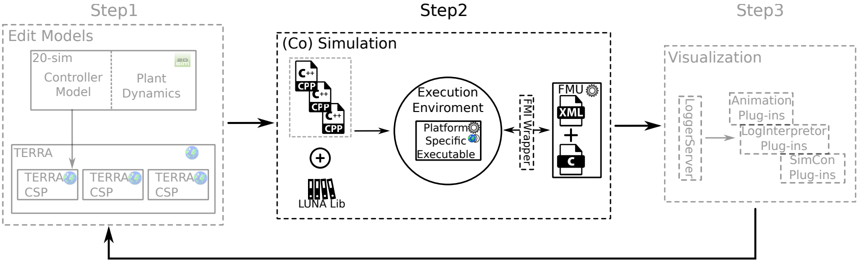
Design of Simulation - A hybrid simulation approach

- Tight-Coupling Execution
  - Integrate 20-sim controller model into TERRA
  - Generate code from the integrated model



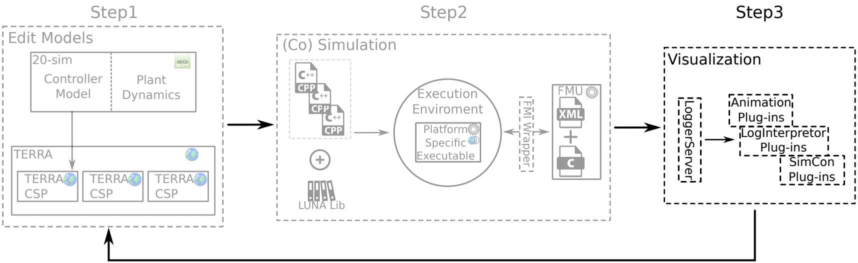
Design of Simulation - A hybrid simulation approach

- Loose-Coupling Execution
  - Generate code from C/P models
  - Generate APIs from TERRA FMI interface model
  - FMI wrapper as coordinator



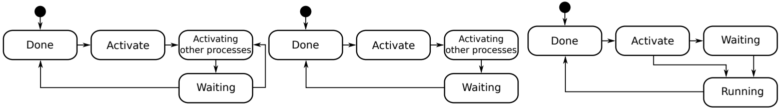
Design of Simulation - A hybrid simulation approach

- Visualizing simulation results
  - Process execution order
  - Results of algorithms
- Iterative and incremental design and development



Visualizing Simulation Results

- Five states for CSP constructs and processes
  - Activate, Activating other processes, Waiting, Running, Done



P ; Q

P || Q
P [] Q

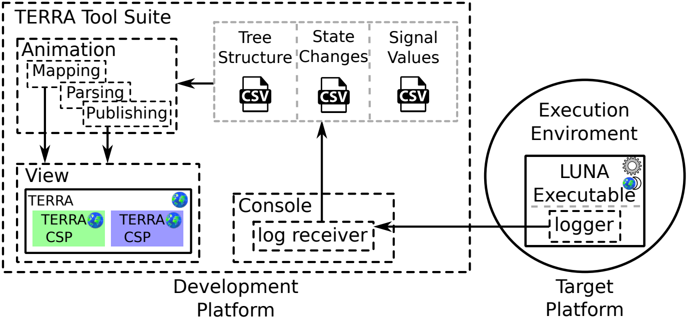
writer !<variable>
reader ?<variable>

- Logging facilities were designed to capture state changes
  - Registration phase
  - States recording phase

	Process ID:1	Process ID:2	Process ID:3		Process ID:n-1	Process ID:n
time stamp	State Index 1	State Index 2	State Index 3	....	State Index n-1	State Index n

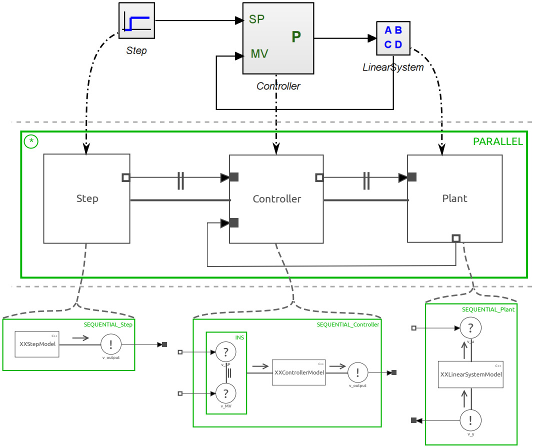
Visualizing Simulation Results

- Overall structure of the visualization
  - Execute models
  - Logged data will be stored as CSV files
  - Mapping model elements
  - Parsing logged data
  - Publishing states to a graphical view



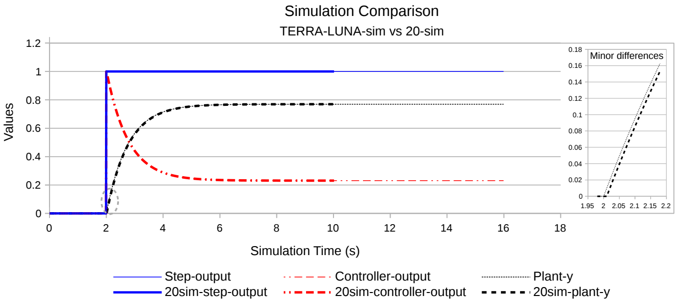
Results

- Loop control model for testing



Results

- Simulation Comparison

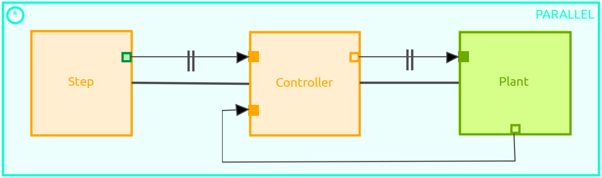


Results

- One snapshot of logged process states

Process ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
time stamp 0.230	1	1	1	1	0	0	3	0	0	0	1	0	2	4	4

- Using different colors to represent process states



Conclusions and Recommendations

- Conclusions
  - The simulation provides comparable results as the ground truth
  - The animation can sufficiently indicate process execution order
  - Opportunity to implement a rapid prototyping system
  - Opportunity to obtain an executable and deployable binary which can be right-first-time
- Recommendations
  - Signal values are not automatically visualized as state changes
  - Options to include or exclude processes/states from animations
  - Timing analysis need to be implemented
  - FMI interfacing and wrapping facilities

Thanks!

Results

- Tree structure of the example model