Formal, Model-based Development and Verification of Hardware, Software and Cyber-Physical Systems

John Colley, Michael Butler
Reading, May 21st 2015
Outline

• Cyber-Physical Systems
• The Rodin Technical Approach
• An Example
• Application Domains
• Summary
Cyber-Physical Systems

• Integrations of Computing and Physical Mechanisms
  – provide physical services
    • Transportation
    • Energy Distribution
    • Medical Care
    • Manufacturing
  – with increased
    • Adaptability
    • Autonomy
    • Efficiency
    • Safety
Cyber-Physical System Challenges

“…. the lack of temporal semantics and adequate concurrency models in computing, and today’s “best effort” networking technologies make predictable and reliable real-time performance difficult, at best.”

Cyber-Physical Systems - Are Computing Foundations Adequate?
Edward A. Lee, EECS, UC Berkeley, 2006
Rodin Technical Approach

Overview

• Focuses on the key role played by Modelling
• Modelling is used at all stages of the Development Process
  – From Requirements Analysis to System Acceptance Testing
• Augments Formal, Refinement-based Modelling and Verification with
  – Simulation
  – Testing

*in a Single Design and Verification Environment*
Rodin

Requirements Capture and Verification

• Event-B Formal Modelling Language
  – Graphical (UML-B)
  – Textual

• Proof Obligation Generation
  – Invariant Preservation
  – Refinement Checking

• Model Checking with ProB plug-in
  – Invariant Violations
  – Refinement Violations
  – Deadlocks
Rodin

Model Validation

• Requirements Tracing
  – ProR Plug-in uses open ReqIF exchange format for requirements

• Graphical Animation
  – ProB provides a simulation engine for Event-B
  – BMotionStudio allows interactive graphical animations to be constructed, driven by the simulation engine
Rodin

Coverage Sign-off

• Model-based Testing
  – ProB model-checking engine

• MC/DC Coverage
  – Measured at each refinement step
    • Related events refine a single abstract event
    • Can the individual events be disabled independently
      – reachability
  – Model Check or Directed Tests

• Co-simulation
  – With Discrete or Continuous Environment
  – Links Structural Coverage to Model Coverage
The Rodin Co-Simulation Framework

• Different Simulation tools are better suited to simulating different parts of a Cyber-physical system
  – Environments
  – Controllers
  – Physical Plant
• The Framework manages the co-operation of multiple simulators to enable effective Cyber-physical system verification
• Supports the Functional Mock Up Interface (FMI)
  – European Automotive Standard
Continuous / discrete co-simulation

Figure 5. Distribution voltage control system in Modelica

Figure 6. Event-B state machine of the OLTC controller
Figure 7. Co-simulation results of the OLTP voltage control (simulation time = 30s, step size = 0.1s)
An Aerospace Example

Landing Gear Doors

1. The Controller will *open* the Doors when the Pilot moves the Lever to Extend or Retract the Landing Gear

2. The Controller will then *close* the Doors when the Landing Gear is fully Extended or Retracted

3. The Doors will remain *open* while the Landing Gear is Extending or Retracting
Safety Requirements

“Any controller – human or automated – needs a model of the process being controlled to control it effectively”

“Accidents can occur when the controller’s process model does not match the state of the system being controlled and the controller issues unsafe commands.”

Engineering a Safer World, Leveson, 2012
System-Theoretic Process Analysis (STPA)

1. Identify Potentially Hazardous Control Actions and derive the Safety Constraints
2. Determine how Unsafe Control Actions could occur
The Door Sub-system Process Models

**Process Model**

**Door Position**
- Locked Open
- Locked Closed
- Opening
- Closing
- Unknown

**Process Model**

**Landing Gear**
- Extended/ing
- Retracted/ing
- Unknown

**Controlled Process**

- OpenDoor
- CloseDoor
- Extend
- Retract
Step I: Identify Potentially Hazardous Control Actions and Derive Safety Constraints

<table>
<thead>
<tr>
<th>Controller Action</th>
<th>Not Providing Causes Hazard</th>
<th>Providing Causes Hazard</th>
<th>Wrong Timing or Order Causes Hazard</th>
<th>Stopped too soon/Applied too long</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Door</td>
<td>Cannot extend Landing Gear for landing</td>
<td>Not Hazardous</td>
<td>Not Hazardous</td>
<td>Damage to Landing Gear/Not Hazardous</td>
</tr>
<tr>
<td>Close Door</td>
<td>Not Hazardous</td>
<td>Damage to Landing Gear</td>
<td>Damage to Landing Gear</td>
<td>Not Hazardous/Not Hazardous</td>
</tr>
</tbody>
</table>

**Safety Constraints**

1. If the Landing Gear is Extending, the Door must be Locked Open
2. If the Landing Gear is Retracting, the Door must be Locked Open
3. A “Close Door” command must only be issued if the Landing Gear is Locked Up or Locked Down
4. An “Open Door” command must only be issued if the Landing Gear is Locked Up or Locked Down
Deriving the Formal Safety Constraints

• Natural Language Constraints developed systematically by the Domain Experts

1. If the Landing Gear is Extending, the Door must be Locked Open
2. If the Landing Gear is Retracting, the Door must be Locked Open
3. A “Close Door” command must only be issued if the Landing Gear is Locked Up or Locked Down
4. An “Open Door” command must only be issued if the Landing Gear is Locked Up or Locked Down

• Formal, Event-B Safety Constraints
  – Derived systematically from the Natural Language Descriptions
  – Linked to Requirements
    • ProR
Deriving the Formal Safety Constraints

- Natural Language Constraints developed systematically by the Domain Experts

1. If the Landing Gear is Extending, the Door must be Locked Open
2. If the Landing Gear is Retracting, the Door must be Locked Open
3. A “Close Door” command must only be issued if the Landing Gear is Locked Up or Locked Down
4. An “Open Door” command must only be issued if the Landing Gear is Locked Up or Locked Down

\[
\text{gearstate} \in \{\text{locked\_down, locked\_up}\} \lor \text{doorstate} = \text{locked\_open}
\]

\[
\text{event Close}
\]
\[
\text{where}
\]
\[
\text{@grd1 gearstate} \in \{\text{locked\_down, locked\_up}\}
\]
\[
\text{@grd2 doorstate} \in \{\text{opening, locked\_open}\}
\]
\[
\text{then}
\]
\[
\text{@act1 doorstate} := \text{closing}
\]
end
The Model Extended FSM
Modelling Timing

• Specification refinement begins with an untimed model
• Refinement introduces sequences of temporal events
• Implementing synchronous timing semantics in the refined Event-B model enables synchronisation and communication between processes without race
  – Implements HDL cycle-based semantics
  – Enables HDL and Assertion generation from Event-B
Refinement: Introducing the Handle and Timing

[Diagram of state transitions involving G (locked_up, extending, locked_down), D (opening, closing), and H (UP, DOWN). The diagram includes states such as Open, Close, Extend, Retract, CompleteOpen, and CompleteClose.]

- G locked_up → D opening → H DOWN
- G locked_up → D locked_open → H DOWN
- G locked_up → D locked_closed → H UP
- G locked_down → D closing → H DOWN
- G locked_down → D locked_open → H UP
- G locked_down → D locked_closed → H UP
- G extending → D locked_open → H DOWN
- G retracting → D locked_open → H DOWN
- G locked_up → D closing → H UP
- G locked_down → D opening → H UP
- G locked_up → D locked_open → H UP
- G locked_down → D locked_open → H UP
- G locked_down → D locked_closed → H DOWN
- G extending → D locked_open → H DOWN
- G retracting → D locked_open → H DOWN
- G locked_up → D closing → H UP
- G locked_down → D opening → H UP
- G locked_up → D locked_open → H UP
- G locked_down → D locked_open → H UP
- G locked_down → D locked_closed → H DOWN

States: Idle, CompleteOpen, CompleteClose, Extend, Retract, Close.
Refinement: Introducing the Handle and Timing

Rodin

- Prove Responsiveness
  - Prove that illegal states of <G,D,H> are unreachable
  - Prove that for each legal state of <G,D,H> at least one event is enabled (disjunction of guards)
Refinement: Introducing the Handle and Timing

- Model Check
- Generate Tests (Constraint-based)
- Measure Coverage (MC/DC)
  - all the guards of all the events can be set independently to FALSE for all states

ProB
Application Domains

• Aerospace
  – DO-254/178C

• Rail
  – Signalling
  – Train Doors
  – Driverless Trains

• Energy Distribution
  – Smart Grids

• Automotive
  – Lane Centering
  – Cruise Control
  – Automatic Gearbox Control

• Defence
Rodin Summary

• **Formal Modelling** supported by strong Formal Verification Tools to establish deep understanding of Specification and Design

• **Simulation-based Verification** to ensure that the Formal Models exhibit the expected behaviour and timing in the target physical environment

• **Model-based Testing** for the systematic generation of high-coverage test suites

[http://www.advance-ict.eu](http://www.advance-ict.eu)