Model-Driven Engineering for Simulation: the PRISE platform

Jérôme Hugues, ISAE/DMIA
Pierre Siron, ISAE/DMIA
ISAE is a French “Grande Ecole”, with a strong focus on space and avionics application, reference school for the domain
  » Graduates 500+ engineers, PhDs, Master programs

Applied science, multi-domain engineering

Needs to build representative platform both
  » Research: new functions, implementation, certification, …
  » Teaching: system-scale down to understanding a particular block

PRISE is an ISAE-funded project, also supported by MRIS
  » “Pedagogical and Research Infrastructure for Software-intensive Embedded systems”

Build representative benches of avionics system
  » To support both teaching and research activities
> **Realistic simulation: mechanical, computer-based**
  » Real-time, connected to sensors, cockpit
  » Simulated environment (atmosphere, GPS, traffic, …)

> **Testcase: A320, in controlled environment**
  » Meteorological, atmosphere
  » Flight dynamics of A320
  » External traffic
  » Pilot/autopilot

> **Use cases**
  » Control/command,
  » Middleware, human factors, ..
PRISE/SDSE: a multi-disciplinary workbench

Software:
- Industrial tools: Matlab, SCADE, RTOS
- @ ISAE: HLA simulation framework, AADL

Hardware:
- Use state of the art hardware, equivalent to industry
- Same processors, field buses, etc.

Computer Science
Control/command
Avionics
Propulsion
Aerodynamics
SDSE := Simulation Distribuée de Systèmes Embarqués

- Goal: dispatch plane functions /environment in blocks
  - One block == one function, one concern

- Blocks synchronized using IEEE HLA standard
  - Real-time simulation, integration of various blocks
  - CERTI: libre software implementation of HLA, by ONERA & ISAE
> HLA standards: 1.3 [HLA98] and IEEE 1516-2000 [HLA10]
> Address distributed discrete-event simulations
> Promotes reusability and interoperability
> Complex simulation in HLA: the HLA Federation

» A federation is a collection of federates (e.g. simpler simulators), each performing a sequence of computations, interconnected by a Run Time Infrastructure (RTI).
Organized as a set of C++ classes

For each federation, a Federation Object Model (FOM) describes shared objects, attributes and interaction

Six management areas of HLA services:

- **Federation**, manages the whole federation (create, destroy…)
- **Declaration**, describes object attributes each federates will publish, subscribe to
- **Object**, how to produce or receive updated attributes from the federation
- **Time**, provides policies and mechanisms to negotiate time advances
- **Ownership**, handles property and responsibilities of object instances
- **Data distribution**, manage publication/subscription by interests

<table>
<thead>
<tr>
<th>Management areas</th>
<th>Services</th>
<th>Acronym</th>
<th>Description (non-formal)</th>
<th>Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federation</td>
<td>createFederationExecution()</td>
<td>TCK</td>
<td>create a federation</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>joinFederationExecution()</td>
<td></td>
<td>join a federation</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>resigFederationExecution()</td>
<td></td>
<td>quit a federation</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>destroyFederationExecution()</td>
<td></td>
<td>destroy a federation</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>registerFed...Sync...Point()</td>
<td></td>
<td>register a synchronization point</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>tick()</td>
<td></td>
<td>wait a synchronization point succeeded</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>announceSynchronizationPoint()</td>
<td></td>
<td>release from a synchronization point</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>federationSynchronized()</td>
<td></td>
<td>announce the federation is synchronized</td>
<td>normal</td>
</tr>
<tr>
<td>Declaration</td>
<td>publishObjectClass()</td>
<td></td>
<td>declare the publication of an object</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>subscribeObjectClass()</td>
<td></td>
<td>subscribe to an object</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>unsubscribeObjectClass()</td>
<td></td>
<td>unsubscribe to an object</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>unpublishObjectClass()</td>
<td></td>
<td>unpublish an object</td>
<td>normal</td>
</tr>
<tr>
<td>Object</td>
<td>registerObjectInstance()</td>
<td>UAV</td>
<td>register an object instance</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>discoverObjectInstance()</td>
<td></td>
<td>for object instances discovering</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>updateAttributeValues()</td>
<td>RAV</td>
<td>send and update value to the federation</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>reflectAttributeValues()</td>
<td></td>
<td>receive updated value from the federation</td>
<td>normal</td>
</tr>
<tr>
<td>Time</td>
<td>enableTimeRegulation()</td>
<td>TAR</td>
<td>declare federate is regulator</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>timeRegulationEnabled()</td>
<td>TAG</td>
<td>federate as regulator succeeded</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>enableTimeConstrained()</td>
<td>NER</td>
<td>declare federate constrained</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>timeConstrainedEnabled()</td>
<td></td>
<td>federate as constrained succeeded</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>timeAdvanceRequest()</td>
<td></td>
<td>ask to advance federate’s time (blocked)</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>timeAdvanceGrant()</td>
<td></td>
<td>notify time advancement granted</td>
<td>normal</td>
</tr>
<tr>
<td></td>
<td>nextEventRequest()</td>
<td></td>
<td>ask to advance federate’s time</td>
<td>normal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interaction characteristics</th>
</tr>
</thead>
</table>
| (Interactions Class InteractionRoot BEST_EFFORT RECEIVE | (Class RTI private BEST_EFFORT RECEIVE) ) |)

<table>
<thead>
<tr>
<th>Information: Federation's name Version Federate names</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Objects Class ObjectRoot (Attribute privilegeToDelete reliable timestamp) (Class RTI private) Class myObjectClass (Attribute VAL RELIABLE RECEIVE (Attribute VAL2 RELIABLE TIMESTAMP) ) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shared object attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RO: receive ordered event</td>
</tr>
<tr>
<td>TSO: time stamp ordered event</td>
</tr>
</tbody>
</table>
HLA is an “old-school” middleware, from the 90’s

» Many handwritten code for initialization, synchronization

» Mix user-code, real-time concerns, deployment information
  • Simulink models, sensors drivers ported to C++/HLA API
  • Use of HLA synchronization and time managements capabilities

» Yet, mature support for simulating complex systems

Could be revisited by leveraging MDE

» Provide domain-expert with an easier access to simulation elements: simulation models, communication patterns, configuration parameters

» Separation of concerns, allow for analysis tools to operate

A good use case for AADL
What you should know about AADL @ ISAE

ISAE contributions to SAE
AADL since 2009
Lead on the Ocarina toolset, used by ESA

Code generation: Ada, C (POSIX, ARINC653), RTOS
Integration 3rd party code from MBSE (SCADE, Simulink)
TRL 7 with ESA (ECSS E-40)

Scheduling: Cheddar, MAST
TRL 4 with ESA

Model checking: Petri Nets
TRL 2 (PhD contribution)

Architectural Constraints/Requirements checks
TRL 6 with SEI, being standardized

AADL covers many parts of the V cycle: model checking, scheduling, safety and reliability (ARP4754) and code generation

Non-functional properties

Link to code/model

Concrecy view

Physical view

Architecture helps you focusing on the actual system

Link with SysML models
Exploits AADL models information to generate application-tailored middleware
SDSE is implemented as a set of real-time blocks, with well-defined interfaces, yet buried in C++ code: reverse engineering

- Extract interfaces elements
- NFP elements
- Rebuild architecture in AADLv2

Defines a notionally equivalent AADL model

- AADL process + thread <-> HLA federate
- Thread compute entrypoint <-> functional code
- AADL NFP + generated glue code <-> HLA configuration code
- AADL transport layer <-> HLA middleware
AADL defines time semantics

- Attached to threads elements
- Three entry points
  - Activation, initialization, dispatch
  - Attach specific behavior to configure elements, perform computations

These can be supported directly using HLA services, as part of a specific variant of the PolyORB-HI/C AADL runtime

<table>
<thead>
<tr>
<th>Dispatch_Protocol</th>
<th>Enabled</th>
<th>Wait_For_Dispatch Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periodic</td>
<td>(t = \text{Period})</td>
<td>(t \leq \text{Period} \land 8t = 1)</td>
</tr>
<tr>
<td>Aperiodic</td>
<td>(\exists p \in E : p \neq \emptyset)</td>
<td>(\forall p \in E : p = \emptyset)</td>
</tr>
<tr>
<td>Sporadic</td>
<td>(t \geq \text{Period} \land \exists p \in E : p \neq \emptyset)</td>
<td>(t &lt; \text{Period} \land \forall (t &gt; \text{Period} \land \forall p \in E : p = \emptyset))</td>
</tr>
<tr>
<td>Timed</td>
<td>(\exists p \in E : p \neq \emptyset \lor t = \text{Period})</td>
<td>(\forall p \in E : p = \emptyset \land t &lt; \text{Period})</td>
</tr>
<tr>
<td>Hybrid</td>
<td>(t = \text{Period} \land \exists p \in E : p \neq \emptyset)</td>
<td>(\forall t \in E : p = \emptyset \land t \leq \text{Period})</td>
</tr>
<tr>
<td>Background</td>
<td>true</td>
<td>(t = 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Activate_Entrypoint</th>
<th>Instantiate proxies createFederationExecution joinFederation subscribeObjectClassAttributes publishObjectClass registerObjectInstance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize_Entrypoint</td>
<td>enableTimeRegulation timeRegulationEnabled registerFederateSynchronizationPoint announceSynchronizationPoint synchronizationPointAchieved</td>
</tr>
<tr>
<td>Compute_Entrypoint</td>
<td>updateAttributeValues reflectAttributeValues timeAdvanceRequest/nextEventRequest</td>
</tr>
</tbody>
</table>

AADL          | HLA Services |
From AADL model back to HLA simulation code
> Proposed a proof of concept of AADL/HLA coupling

  » Support for analysis of simulation model
    • Scheduling, resource usage

  » Validation of functional code under correct model of time
    • “just” need to perform timing analysis on real hardware target 😊

> Future work

  » Complete implementation as part of Ocarina AADL tools
  » Consider inclusion of FMUs, to model environment laws
  » Generation of co-simulation code when mixing heterogeneous model of computations