Compiling Heterogeneous Models: Motivations and Challenges

Matteo Bordin, Tonu Naks, Marc Pantel and Andres Toom

ERTS$^2$ 2012 – February 1$^{st}$-3$^{rd}$, Toulouse
The topic of this presentation

Model Compilers for Safety-Critical Systems
Original scope of the project (2010)

• **Evolve/Productize GeneAuto technology**
  – A model compiler for Simulink, Stateflow and Scicos
  – ITEA project, 2005-2008 (see ERTS 2008/2010)
  – Mostly developed by IB Krates, IRIT and Alyotech
  – Validated in 2009-2010 by Airbus and Astrium (see ERTS 2010)

• **Emphasis on qualification in a DO-178 context**

• **Small, focused project**
Fast forward to October 2011...
Model-based integration
Qualifiable code generators
Tunable & Open-Source
Positioning the P toolset

- Specification Model
  - Compliance
  - Verifiability
  - Accuracy & Consistency
  - Compatibility

- Design Model
  - Compliance
  - Traceability
  - Accuracy
  - Consistency

- Src Code
  - Compliance

- Executable
  - Compliance
  - Robustness
Or, in simpler terms...

... the model compiler for TOPCASED
Technical approach
Project P genealogy
Ultimate goals: a tool provider perspective

• To put an end to segregation
  – Between system, software, control engineers

• To deploy a lean/agile qualification process
  – Qualify as you develop, easy re-qualification

• To build an economically viable ecosystem
  – For both tech providers and service companies
Ultimate goals: a scientist perspective

• To put an end to segregation between
  – System, software and control engineers
  – Industrial and academic partners
  – Formal methods and engineering tools

• To define a common intermediate language
  – Providing a well defined semantics
  – To connect analysis, synthesis and verification tools based on different elementary formal languages
  – To compare, improve and combine these tools
  – To connect with industrial modeling tools
The challenges
Semantic convergence of DSLs (I)

• P formalism is not the union of all input languages

• Example: functional behavior
  – Synchronous state machines: Simulink, Stateflow
  – Asynchronous state machines: UML, AADL-BA
  – Execution protocols: AADL, UML

• Example: architectural modeling
  – SysML/UML/MARTE
  – AADL/ARINC-653
  – Autosar/East-ADL2 (TIMMO)
Example: semantic convergence of behavioral DSLs
Semantic convergence: Current proposal

• Derived mainly from TOPCASED-AADL and GeneAuto

• Strong separation between
  – Function (derived from Synchronous MoC)
    ■ Black box: no interaction with architecture during execution
    ■ Synchronous w.r.t. Architecture
    ■ Can be internally concurrent
  – Architecture (derived from AADL MoC)
    ■ Clock management
    ■ Synchronous/Asynchronous communication
    ■ Distribution/Process/Thread management
Verification of model-based integration

• **Well beyond typing**
  – Clock constraints
  – Synchronization
  – Hard real time properties

• **Allocation of behavior**
  – Extra functional properties are assumed to hold in simulation
  – Are they respected by the architecture?

• **Support for the synthesis of software architecture**
  – Logical and physical
Flexible & Open Qualification

• **Support for future input DSLs**
  
  – Separable qualification evidence for:
    
    - Importers
    - Code generators

• **Support for new code generation strategies**
  
  – Separable qualification evidence for:
    
    - Target independent model transformations
    - “Printing phase”

• **Toolkit shipping with qualification infrastructure**
  
  – To allow incremental re-qualification
Current state at T0+4
Figuring out the consortium agreement
Synchronizing with on-going projects

- CHESS, VERDE
  - Inspiration for component model and architecture modeling
- MBAT
  - Model verification of the P formalism
- OPENCOSS & Safecer
  - Tooling for qualification artifact management
- TASTE
  - AADL tooling
- openETCS
- CESAR
- OPEES/PolarSys
- ...
Technical work

• **Migrating GeneAuto technology**
  – Due to unclear DO-178 qualiﬁability of technology

• **DO-178C –compliant qualification data**
  – Migrating GeneAuto data
  – User-TOR, Developer-TOR, …

• **Experimenting the Open-DO collaborative platform**
  – Interactive wiki instead of emails
  – Wiki pages instead of Word for deliverables
A recap of DO-178C vision

The DO-178C Model-based development supplement identifies two different kinds of models:

- **Specification models** to formalize High-level requirements (HLR). In this case, the model represents the set of system requirements allocated on software.
- **Design models**, to formalize Low-level requirements (LLR). In this case, the model is traceable (and must be verified against) the HLR.

We address the use of the qualifyable P toolset to generate code from **design models**.

![Diagram](image)

**Picture 1 – Model based development when developing code from design models**