Simulink integration

Julien Delange, Jérôme Hugues
<delange@enst.fr, hugues@enst.fr>
Rationale

Different approaches, common goals

- AADL models system architecture
- Simulink models software
- Common goal: design, validate and implement real-time systems

Purpose: integrate Simulink in AADL models

- Full MBE approach, no code is required!
- Deploy Simulink code according to AADL models
Integrating Simulink & AADL

- Separates software in software blocks
  - Recursively, blocks can contain other blocks
  - Blocks communicate with signals/parameters
    - #1: Mapping of Simulink blocks into AADL components

- Defines its own types
  - #2: Integration of these types into AADL models
Integration challenge

- Simulink blocks mapping
  - How to map Simulink blocks in AADL?

- Simulink types mapping
  - Integration of Simulink types in AADL?

- Code generation approach
  - Combine AADL and Simulink generated code together?
Use case

- Use the F14 example from Simulink
  - Aircraft guidance system

- Map it in a distributed system designed with AADL
  - Three processes, inter-process communications

- Integrate Simulink and AADL generated code
  - Implement complex systems without writing any line of code!
Use case

- Map each block to a thread, add relevant ports
- Map data types in an AADL data component

F-14 Longitudinal Flight Control

This demonstration models a flight control system for the longitudinal motion of a Grumman Aerospace F-14.
F-14 Longitudinal Flight Control

This demonstration models a flight control for the longitudinal motion of a Grumman Aerospace F-14.
subprogram pilot_spf
features
  aircraft1 : in parameter simulink_realt
    {Simulink::Signal => "sldemo_f14/Aircraft Dynamics Model/Vertical Channel/Sum";};
  aircraft2 : in parameter simulink_realt
    {Simulink::Signal => "sldemo_f14/Aircraft Dynamics Model/Pitch Channel/Sum";};
  blibli    : in parameter simulink_realt
    {Simulink::Signal => "sldemo_f14/Aircraft Dynamics Model/Pitch Channel/Integrate qdot";};
properties
  source_language => Simulink;
  source_name    => "sldemo_f14";
  source_text    => "../../simulink-code";
end pilot_spf;

Types mapping

data simulink_realt
  properties
    Type_Source_Name => "real_T";
    Source_Name     => "sldemo_f14";
    Source_Data_Size => 8 Bytes;
    source_language => Simulink;
end simulink_realt;

Simulink block mapping
Implementation status

Implementation in our AADL-toolsuite: Ocarina

- Use case with a significant AADL and Simulink models

Integration of Simulink generated code

- Currently use real-time workshop (RTW)
Mapping rules, pros/cons

- **Simple approach, works fine**
  - Use case with significant models
  - Only one thread can execute Simulink code

- **Need to check mapping consistency**
  - Mapping can contain semantic errors
  - E.g. no respect of scheduling properties

- **Need to integrate other Simulink code generators**
  - Simulink embedded coder
  - From a tooling perspective: address variations in code generated when new releases of Simulink come out
Conclusion

- Introduce the 0 line of code programming
  - Full MBE approach
  - Integrate and deploy Simulink models in a distributed systems

- Some work remains for a full integration

- Extend this approach to other modeling approaches
  - Integration of SCADE/Lustre/Esterel generated code