Complexity-Reducing Design Patterns for Cyber-Physical Systems

SysML-AADL Model Translation

30 January 2012
AADL Workshop
Darren Cofer / Steve Miller
Outline

- Project vision
- Tool environment
- Technologies
  - *System-level modeling and translation*
  - Complexity-Reducing Architectural Patterns
  - Compositional verification
- Demo
Team

- Rockwell Collins / Advanced Technology Center
  - Darren Cofer, Steven Miller, Andrew Gacek
  - System modeling & analysis, tooling, integration
- UIUC
  - Lui Sha
  - Design pattern development
- University of MN
  - Michael Whalen
  - Pattern verification, compositional analysis
- WWTG
  - Chris Walter, Brian LaValley
  - Pattern implementation & analysis tools
Vision

• Improve effectiveness and scalability of system design and verification through pre-verified design patterns and compositional reasoning.
**Approach**

Complexity-reducing design patterns
- Capture best solutions to architectural design problems
- Reuse of formally verified solutions
- Increase level of design abstraction

Compositional verification
- Reason about system behavior based on contracts and system design model structure
- Compositional approach scales to large software systems

System architecture modeling
- Apply formal specification and analysis tools to system-level design
- Separate component specification and implementation
- Automated model translation

Design Flow

1. **System architecture modeling**
   - Apply formal specification and analysis tools to system-level design
   - Separate component specification and implementation
   - Automated model translation

2. **Complexity-reducing design patterns**
   - Capture best solutions to architectural design problems
   - Reuse of formally verified solutions
   - Increase level of design abstraction

3. **Compositional verification**
   - Reason about system behavior based on contracts and system design model structure
   - Compositional approach scales to large software systems
Tool chain

**SysML**

**AADL**

**Lustre**

**SysML-AADL translation**

**OSATE:**
AADL modeling

**EDICT:**
Architectural patterns

**Lute:**
Structural verification

**AGREE:**
Compositional behavior verification

**Enterprise Architect**

**Eclipse**

**KIND**

© Copyright 2011 Rockwell Collins, Inc. All rights reserved.
System architecture modeling

- We have been very successful at applying formal methods to software components produced in model-based development environments
  - Gryphon translation framework
- Objective
  - Leverage this knowledge and apply formal methods to the system design process
- Issues
  - Modeling language and tools
  - Different models of computation
  - Scalability
System modeling and translation

- AADL is a good fit and provides sufficiently formal notation
  - Available tools do not provide stable graphical environment
  - OSATE: open source, Eclipse-based
- SysML is being adopted by many organizations for system design
  - But has no formal semantics
  - No common textual representation across tools
- Solution: Eclipse plugin that provides bidirectional translation
  - Based on Enterprise Architect SysML tool used by Rockwell Collins
  - Define block stereotypes that correspond to AADL objects
Scale and composition

- Architectural model does not capture implementation details
  - Component descriptions, interfaces, interconnections
- Assume/guarantee contracts provide the information needed from other modeling domains to reason about system-level properties
  - Guarantees correspond to the component requirements
  - Assumptions correspond to the environmental constraints that were used in proving the component requirements
  - Contract specifies precisely the information that is needed to reason about the component’s interaction with other parts of the system
  - Supports hierarchical decomposition of verification process
- Contract can be applied to both components and design patterns
  - Mechanism for verification reuse
Internal representation

class Translator Types

PivotPackage
- name : String
- GetPath() : String
- GetRelativePath() : String

PivotPart
- facts : String
- name : String
- properties : Collection<String>
- GetPath() : String
- GetRelativePath() : String

PivotPort
- properties : Collection<String>
- GetPath() : String
- GetRelativePath() : String

PivotType
- category : PivotCategory
- contract : String
- name : String
- properties : Collection<String>
- GetPath() : String
- GetRelativePath() : String

PivotPortType
- direction : String
- feature : PivotFeature
- isConjugated : String
- name : String
- properties : Collection<String>
- GetPath() : String
- GetRelativePath() : String

PivotConnection
- name : String
- properties : Collection<String>
- GetPath() : String
- GetRelativePath() : String
# AADL components and features in SysML

<table>
<thead>
<tr>
<th>SysML Block Stereotype</th>
<th>AADL Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>AADL_system</td>
<td>System*</td>
</tr>
<tr>
<td>AADL_data</td>
<td>Data</td>
</tr>
<tr>
<td>AADL_process</td>
<td>Process</td>
</tr>
<tr>
<td>AADL_thread</td>
<td>Thread</td>
</tr>
<tr>
<td>AADL_memory</td>
<td>Memory</td>
</tr>
<tr>
<td>AADL_bus</td>
<td>Bus</td>
</tr>
<tr>
<td>AADL_device</td>
<td>Device</td>
</tr>
<tr>
<td>Not Supported</td>
<td>Abstract</td>
</tr>
<tr>
<td>Not Supported</td>
<td>Thread Groups</td>
</tr>
<tr>
<td>Not Supported</td>
<td>Subprograms</td>
</tr>
</tbody>
</table>

* Default if SysML block is not stereotyped

<table>
<thead>
<tr>
<th>SysML Port Stereotype</th>
<th>AADL Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>AADL_port</td>
<td>Port*</td>
</tr>
<tr>
<td>AADL_provides_data_access</td>
<td>Provides Data Access</td>
</tr>
<tr>
<td>AADL_requires_data_access</td>
<td>Requires Data Access</td>
</tr>
<tr>
<td>AADL_provides_bus_access</td>
<td>Provides Bus Access</td>
</tr>
<tr>
<td>AADL_requires_bus_access</td>
<td>Requires Bus Access</td>
</tr>
<tr>
<td>Not Supported</td>
<td>Port Groups</td>
</tr>
</tbody>
</table>

* Default if SysML port is not stereotyped
Defining AADL stereotypes in EA
Contracts in SysML

- Contracts describe behavior of components and design patterns in system design
- Used for formal verification of system requirements and checking design validity
- Currently expressed in PSL
- Implemented in AADL as string property and processed separately
- Implemented in SysML as constraint referencing a text file
Initial Avionics System
Final Avionics System (after pattern transformations)
Verification tools

Lute

AGREE

Counterexample