Complexity-Reducing Design Patterns for Cyber-Physical Systems

SysML-AADL Model Translation
Outline

• Project vision
• Tool environment
• Technologies
  • System-level modeling and translation
  • Complexity-Reducing Architectural Patterns
  • Compositional verification
• Next steps
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

Global Product Data Interoperability Summit | 2012

- Improve effectiveness and scalability of system design and verification through pre-verified design patterns and compositional reasoning

![Diagram with nodes and arrows representing system design and verification processes.]

- VOTE MULTIPLE DATA
- SENSOR 1 → SENSOR 2 → SENSOR 3
- LRU
- COMPUTING RESOURCE A → COMPUTING RESOURCE B
- FAIL-SILENT NODE FROM REPLICAS
- VERIFIED AVAILABILITY
- ARCHITECTURE MODEL
- VERIFIED INTEGRITY
- COMPOSITIONAL PROOF OF CORRECTNESS (ASSUME – GUARANTEE)
- SAFETY, BEHAVIORAL, PERFORMANCE PROPERTIES

COMPOSITION
**Approach**

**Global Product Data Interoperability Summit | 2012**

### 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

Global Product Data Interoperability Summit | 2012
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

Global Product Data Interoperability Summit | 2012

Enterprise Architect

Eclipse

SysML

AADL

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

PivotConnection

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

PivotCategory

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

PivotFeature

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

PivotImpls

- parts 0..*
- pkg 1
- types 0..*
- type 1
- impls 0..*

PivotParts

- ports 0..*
- parent 1
- pkg 1
- types 0..*
- type 1

PivotConnections

- connections 0..*
- parent 1
- pkg 1
- parts 0..*
- parent 0..1

PivotPortTypes

- in 0..*
- dest 1
- out 0..*
- source 1

AADL

SysML
### AADL components and features in SysML

**AADL components**

<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><strong>Not Supported</strong></td>
<td>Abstract</td>
</tr>
<tr>
<td><strong>Not Supported</strong></td>
<td>Thread Groups</td>
</tr>
<tr>
<td><strong>Not Supported</strong></td>
<td>Subprograms</td>
</tr>
</tbody>
</table>

* Default if SysML block is not stereotyped

### AADL features

<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><strong>Not Supported</strong></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
Next steps

- Extend compositional verification to more complex models of computation
  - Multiple rates, delays, asynchrony
- Incorporate additional design patterns in library
  - Especially fault tolerance patterns with existing verification artifacts
- Improved annotation of contracts in architecture models
  - AADL annex? Alternate representations (e.g., sequence diagrams?)
- More general mechanism for composing evidence from multiple sources
  - Evidence graph, assurance case
META Design Flow

Global Product Data Interoperability Summit | 2012

Initial System Design
- Provides basic control system functionality
- Automatic translation between SysML and AADL design environments
- No fault-tolerance: A single failure causes violation of system requirements

System architecture design and verification environment provides correct-by-construction system synthesis

Contract-based Reasoning
- Basis for scalability
- Assume/guarantee contracts provide the information from components needed to reason about system-level properties
- Pattern contracts inserted automatically
- Component contracts come from component library

Requirement counterexample
Contact

Darren Cofer
Advanced Technology Center
Rockwell Collins
ddcofer@rockwellcollins.com
319-263-3571

Gregory Pollari
Advanced Manufacturing Technology
Rockwell Collins
gmpollar@rockwellcollins.com
319-295-1629