Common Executable Architectures for Multiple Level V&V

Ronald Fernandes
Biyan Li
Perakath Benjamin
Knowledge Based Systems, Inc.

Jong Hwang
Kenneth Littlejohn
AFRL, WPAFB

2012 Safe and Secure Systems and Software Symposium (S5)
Fairborn, OH
Outline

• Background and challenge
• Enterprise Architectures & Executable Architectures
• Executable Architecture Representation Language
• Levels of Abstractions
• Tools to support Executable Architectures
• Applications
• Conclusions
Background

• Enterprise Architectures alphabet soup
  – DoDAF
  – ToGAF
  – FEAF
  – Zachman

• Modeling languages & syntax
  – UML
  – IDEF
  – BPMN ; WfMC
  – SysML
  – UML profiles
Technical Challenges

• Is a model in DARS valid and verified?
  – In order to reuse models, they must be *validated*
  – In order to reuse models, they must be *verified* – correctness
    • E.g. the “liveness” of an activity model
    • Consistency at the interface between two model types
    • “Executable models”

• Need support for V&V and executable architecture
  – Representation, experimentation, results, methodology

• Can executable models interoperate across tools?
DoDAF 2.0 Approach

• A new approach – separate data from notation
• Three thrusts (working groups)
  – Data (i.e., content, rather than the notation)
  – Methods
  – Presentation
• No executability support!
• V&V not explicit
What is model executability?

- Qualitative – Verification & Validation
- Quantitative – simulation; various flavors
- Web and SOA – WSDL, BPEL, workflow
- Project management
- Risk analysis
- Information infrastructure generation – SOA
- Code generation; model driven architecture
- xUML
Executable Architecture through M&S: different places; different abstractions

Levels of system abstraction

Interoperability Assessments

Technology Partitions

Layer Partitions

Foundation Partitions

Partition Table

V&V at the interface: interoperability

V&V between products: model composability

V&V within the product: model mixed-mode executability
From the system architectures can we generate a set of Discrete and Continuous and Mixed models that can be used to evaluate “Will this system of systems execute (interoperability) and will it provide the desired capability (performance)?”
Scope of Architectural Framework V&V

Level 1: Consistency within a model

Level 2: Consistency between two products

Level 3: Consistency within the OV or SV

Level 4: Consistency between OVs and SVs

Level 5: Consistency between systems
Simulation-based executable architecture issues

• Different types of simulation
  – Monte Carlo
  – Colored Petri Nets
  – Discrete Event Simulation

• Need for mixed-mode simulation modeling

• Representation that’s neither too high-level (e.g., OV2, OV5, OV6b, OV6c) nor too low-level (e.g., SIMAN / SLAM) but somewhere in between so that design rationale is captured

• Various simulation languages and vendors
Levels of Abstraction

DoDAF

MoDAF

UML

UPDM

Enterprise Architecture modes

CADM linkages

CPN Abstraction (PNML, CPNRL, etc.)

CPN Abs. → CPN Tools translator

CPN Tools model

VSO model

EARL

abstraction linkages

CPN Abs → VSO translator

DES Abstraction (SRML, etc.)

DES Abstraction → Witness translator

Witness model

DES Abstraction → Arena translator

Arena model

WORKFLOW / BPEL engines

Enterprise Architecture modes
Overall Solution Concept

• Extend CADM to include executable models
• Executable Architecture Representation Language (EARL)
  • EARL is abstract; define simulation flavor-specific language that can be auto-generated from EARL
• Show how an EARL model can be executed, verified, & validated
• Provide a framework for executable models to interoperate among tools
• Document the methodology for developing and sharing executable models
Traceability Among Models

CADM

EARL

BGML + CPNRL

SRML + XMILE

CAMP-G

CPN Tools

Simulink

Stateflow

XMILE

Vensim

Arena

Powersim

Vensim

Refers to

Refers to

Refers to

Refers to

Refers to

Refers to

Refers to

Refers to

Refers to

Refers to
Top Level EARL

**Variables**
a list of variables, which can be discrete or continuous, used as a global or system variables. To be finalized.

**ObjectType**
a list of objecttype, which is reverted back from the combination of object type and variable type to the previous definition i.e., without combining ObjectDataType, DiscreteVariable, and ContinuousVariable in previous version.

**Objects**
a list of object which can be used as different roles in the execution unit.

**ExecutionModules**
A list of ExecutionModule. Each of them is independent of each other. However, they might share some common objecttype or object.

**EAPRODUCT**
modified from DodalProductType, EA means Enterprise Architecture
Colored Petri Net Representation Language (CPNRL)

A module has a unique identifier (id) and consists of an interface (Interface) and several net objects (net.object).

A module hides its net objects. Only the objects within its interface are visible outside the module. In general, a module is defined in its own file.

An EANL file represents an architectural model as a colored petri net, with execution info and results.

Contains the global list of color sets (i.e., data types) used in this file.

execution_results
Simulation Reference Markup Language (SRML)
## Data Dictionary & Mapping

<table>
<thead>
<tr>
<th>IDEF</th>
<th>EARL</th>
<th>CPNRL</th>
<th>DES/SRML</th>
<th>WS-BPEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Flow</td>
<td>Execution Module</td>
<td>Net</td>
<td>Simulation</td>
<td>Process</td>
</tr>
<tr>
<td>Decomposable Process</td>
<td>ExecutionUnit decomposable</td>
<td>Transition with or without subpage</td>
<td>ItemClass and Item</td>
<td>Basic Activity, Container Activity</td>
</tr>
<tr>
<td>IDEF Junction</td>
<td>Junction for merge, split or exclusive selection</td>
<td>Transition</td>
<td>ItemClass and Item</td>
<td>Start or End Point for Flow, Switch, Throw, Repeat Until, While Activity</td>
</tr>
<tr>
<td>Precedence Link</td>
<td>Connection</td>
<td>CPN Place with an input and output arc</td>
<td>Link Element</td>
<td>Link, Source, Target</td>
</tr>
<tr>
<td>Entity</td>
<td>Input or output object</td>
<td>Colorset</td>
<td>EventClass</td>
<td>Input or Output</td>
</tr>
<tr>
<td>Entity inter-arrival time</td>
<td>Inter arrival time element inside input element</td>
<td>Time value under colorset element inside colorset pool element</td>
<td>Script Element</td>
<td>Variable</td>
</tr>
<tr>
<td>Resource requirement</td>
<td>Precondition inside execution unit with structured text</td>
<td>Multiset element for transition and input arc inscription</td>
<td>Script Element</td>
<td>Input</td>
</tr>
<tr>
<td>Required time</td>
<td>Timing element under ExecutionUnit</td>
<td>Timeinfo element for transition</td>
<td>Script Element</td>
<td>Variable</td>
</tr>
</tbody>
</table>
ModelMosaic

• Structural & Behavioral modeling
• Built-in translator generator
• Incompleteness analysis
• Generates executable model
• Built in translators for downstream execution
  – Colored Petri Net; Discrete event; Simulink
• Ontology-driven XSLT generator
  – Easy to import XML-based formats (e.g., UPDM)
  – Easy to export executable models / scripts (e.g. PNML)
  – Easy to import execution results (e.g. Simulink reports)
  – Easy to export reports and views (e.g., XMI)
ModelMosaic – Incomplete Analysis
Simulation Results
Scope of Architectural Framework V&V

- **TYPE 1**: Syntax verification – IDEF, UML, and EARL embellishment
- **TYPE 2**: EARL model verification against core architectural model elements
- **TYPE 3**: Model verification using simulation
- **TYPE 4**: Model validation against real world execution traces or instances
- **TYPE 5**: Model structural consistency checking using rule-based / automated reasoning systems

### Applicability of V&V Type to Levels of Consistency

<table>
<thead>
<tr>
<th>Applicability</th>
<th>Level 1 Consistency: within a model</th>
<th>Level 2 Consistency: between two products</th>
<th>Level 3 Consistency: within an OV / SV structure</th>
<th>Level 4 Consistency: between OVs and SVs</th>
<th>Level 5 Consistency: between systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE 1 Verification</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TYPE 2 Verification</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TYPE 3 Verification</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>TYPE 4 Validation</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>TYPE 5 Verification</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Application of ModelMosaic for Sensor Web

Sensor Web Enablement Components – Web Services
SOS Sensor Description in SensorML, OWL or RDF format
SPS Task and Command System Description

WSDL Import
Service def -> PFD model
Operation -> PFD UOB

Ontology Model Repository

Process Flow Modeling
Define UOBs and connections
Define Inputs, Outputs and Operation

Discovery using SA-WSDL Enhanced UDDI

Add Reference Ontology Model
Reference Inputs, Outputs and Operation to Concepts in Ontology

Association of PFD Process with WSDL Operations

BPEL Generation

BPEL Execution

WSDL Semantic Annotation
Input -> concept in ontology
Output -> concept in ontology

Sensor Web Enablement Components – Web Services
SOS Sensor Description in SensorML, OWL or RDF format
SPS Task and Command System Description

Application of ModelMosaic for Sensor Web
Executable Architectures for Business Processes

1. Capture / design process
2. Simulate & Validate process
3. Execute / implement process
4. Monitor process

- Process Modeling Tools
- Data Analysis Tools
- Simulation Tools
- SOA / BPEL Environment
Conclusions

• EARL is robust to support different types of executable contents
• EARL is a format that is distinct from CADM
  – However, traceability to CADM constructs is a must
• Tools support includes:
  – Extracting & modeling executable content
  – Goal modeling and incompleteness analysis
  – EARL generation, followed by tool-specific generation
  – Translators to simulators and SOA implementation