A Time-based Formalism for the Validation of Semantic Composability

Claudia Szabo*, Yong Meng Teo* and Simon See**
*Department of Computer Science
National University of Singapore
**Sun Microsystems, Inc.
claudias@comp.nus.edu.sg

December 15, 2009
Outline

- Introduction
- Related Work
- CoDES Framework Overview
- Proposed Approach
- Conclusion
What is Composability?

• “Capability to select and assemble simulation components in various combinations to satisfy user requirements”

• Simulation component - “a reusable, self contained unit that is independently testable and usable in a variety of contexts. [It] interacts with its environment only through a well defined interface of inputs and outputs.”

• Various levels of composability:
  ▫ technical, syntactic (engineering), semantic, pragmatic, dynamic, conceptual ...
Semantic Composability

• Does the composed model containing the reused simulation components produce semantically correct results?

• Key issues in validation:
  ▫ Semantic composability is not a closed operation
  ▫ Emergent properties
  ▫ Different validation perspectives:
    • Logical
    • Temporal
    • Formal
  ▫ ...
Formal Validation Overview

- **Software engineering**
  - Model properties - safety, liveness, deadlock-free, ...
  - Simple state machines and few attributes; often timeless

- **Simulation**
  - Necessary to validate model properties with complex state machines & larger #attributes that change over time => **insufficient**
  - Composed model is a valid abstraction of real system => requires validation of model execution over time
Related Work

- Petty and Weisel’s formal theory [Petty and Weisel 2003b]
  - Components statically represented as integer functions;
    Valid model = close enough (undefined) to a perfect model
  - No concept of time – composition based on linear order of components

- DEVS model formal validation [Traore 2006]
  - Composition specified in Z language
  - No concept of time
  - Validates model properties - model execution validation is important

- BOM model validation [Moradi et al. 2007]
  - Informal XML-like representation of composition & scenario
  - Validation requires user-specified scenario at low level of details
CoDES Framework Overview

MODEL COMPOSITION

Model Input
Model Discovery & Selection
Model Integration & Validation

CoDES GUI

Conceptual Model
Query Request(s)
Selected Model
Candidate Simulator
Validated Simulator

Syntax Verifier
Model Locator (MI)
Model Selector
Semantic Validator

Model Repository
Candidate Models
Validate Communication (CI)

Concurrent process
Meta-simulation
Perfect model

December 15, 2009
Winter Simulation Conference
CoDES Component & Simulator

- Component: abstracted as a *black-box* with *in* and/or *out* communication channel, and represented as a *meta-component*:
  - mandatory & specific attributes
  - behavior:
    - External – constraints on input and output (destination, origin, data type, range, etc.)
    - Internal – a finite timed state automaton
      \[
      [I_l]S_p[\Delta t] \xrightarrow{Cond_n} S_t[O_l][A_m]
      \]

- Simulator: represented by *components* (base, model) linked using *connectors*
Proposed Approach

1. Validate desired model properties
   ✓ Correct and meaningful component communication
   ✓ Safety, liveness, deadlock-free – with time
   - Covers software engineering + simulation perspectives

2. Formal model execution validation
   ✓ Comparison with perfect model using state machine
   - New time-based formalism defines validity
   - 5-step process validates model execution over time
Mathematical Formalism - Requirements

1. Representation of simulation components – large #attributes & possible values over time

2. A formal definition of composability

3. Representation of composition and its execution over time

4. A formal definition of validity
Time-based Formalism

• Component
  ▫ function representing input, states & time vs integer function

\[ f_i : X_i \rightarrow Y_i \]
\[ X_i = I_i \times S_i \times T_i, Y_i = I_i \times S_i \times T_i \]

• Mathematical composability
  ▫ time vs integer domains

Given a composed model \( M = \{(f_i, f_j) | i, j = 1, n\} \). Then \( f_i \) and \( f_j \) are composable iif there exists the bijective binary relation

\[ R = \{ (t_n^{(j)}, t_m^{(i)}) \in T_j^{\text{in}} \times T_i^{\text{out}} | t_n^{(j)} > t_m^{(j)} \} \]
Time-based Formalism (2)

- **Simulation**
  - Time-ordered functions vs linear component order (fork?, join?)

\[ \mathcal{S}(M) = \{ \ldots f_i(I_p^j, S_p^j, t_p^j) \rightarrow (O_p^j, S_p^j, t_p^j), \ldots, f_j(I_q^j, S_q^j, t_q^j) \rightarrow (O_q^j, S_q^j, t_q^j), \ldots \} \]

- **Validity** = Comparison with perfect model over time
  - Exact match
  - New degree of similarity index based on related simulation states \( (V_e) \)

**Dynamic execution representation**

**Measure of similarity**
Degree of Semantic Similarity ($V_\varepsilon$)

$$V_\varepsilon(p, q) = \{(p, q) \mid \|p - q\|_\sigma \leq \varepsilon\}$$

- composition states are related:
  - related attributes (in the ontology)
    - same value or
    - same trend

- the same component is executed
5-Step Validation Process

**Composed Model**

1. **Formal Representation**
   - $C_1 = f$
   - ... 

2. **Unfolding and Sampling**

3. **Composition**

4. **Simulation**

**Perfect Model**

1. **Formal Representation**
   - $\bar{C}_1 = \bar{f}$
   - ... 

2. **Unfolding and Sampling**

3. **Composition**

4. **Simulation**

5. **Validation**

   - **Strong equivalence?**
     - Yes: **Valid**
     - No: **L(M) \lor L(M^*)?**
       - Yes: **Valid**
       - No: **Invalid**

*sampled values*
Petty & Weisel’s Formal Composability Theory

1. Formal component representation

Our approach

- static
- unfeasible

2. Composition

- component order

- execution order wrt time

\[ M = f_3 \circ f_2 \circ f_1 \]

3. Validation

- less meaningful
- \( R \): validation relation undefined

- strong
- semantically close:

\[ V \epsilon \]
Implementation

- Java program transforms component COML files into functional representation and subsequently into LTS
  - **Exact match:**
    - the BISIMULATOR tool from the CADP toolset determines strong equivalence
  - **Close match:**
    - Java program determines related states according to proposed semantic metric relation; related states are subsequently validated by BISIMULATOR

- **Weakness**
  - Based on sampling - #samples vs computation cost
  - Degree of semantic similarity—formal measure vs semantic meaning of validity
## Experimental Results

<table>
<thead>
<tr>
<th>Model</th>
<th>#Components</th>
<th>#LTS States</th>
<th>Result</th>
<th>Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Queueing Networks Application Domain</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-Server Queue</td>
<td>3</td>
<td>12</td>
<td>Valid</td>
<td>1.97</td>
</tr>
<tr>
<td>Single-Server Queue – 2 job classes</td>
<td>3</td>
<td>12</td>
<td>Invalid</td>
<td>4.72</td>
</tr>
<tr>
<td>Grid System*</td>
<td>11</td>
<td>51</td>
<td>Valid</td>
<td>5.54</td>
</tr>
<tr>
<td>Grid System* - 2 job classes</td>
<td>11</td>
<td>51</td>
<td>Invalid</td>
<td>8.29</td>
</tr>
<tr>
<td><strong>Military Training Application Domain</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tank vs Soldier troop** (shoot &amp; scoot)</td>
<td>2</td>
<td>21</td>
<td>Valid</td>
<td>6.2</td>
</tr>
<tr>
<td>Tank vs. Solider troop</td>
<td>2</td>
<td>21</td>
<td>Invalid</td>
<td>22.3</td>
</tr>
</tbody>
</table>

*Grid meta-scheduler with two virtual organizations, two and three nodes respectively

**Tank vs. a troop of soldiers attack scenario – data-driven components
Conclusion

• New time-based formalism to validate semantic composability
  ▫ Measure of semantic similarity - $V_\varepsilon$
  ▫ Model composition execution sequences validated using bisimulation

• Open problems
  ▫ Composition using model components
  ▫ Valid vs invalid models
  ▫ ...
Questions?


