CODES: An Integrated Approach to Composable Modeling and Simulation

Yong Meng Teo** and Claudia Szabo
Department of Computer Science
National University of Singapore

**Asia-Pacific Science & Technology Centre
Sun Microsystems Inc.
email: teoym@comp.nus.edu.sg

Composable Simulation

*Composability* is the capability to select and assemble simulation components in various combinations into simulation systems to satisfy specific user requirements.
Outline

- Motivation
- Objective
- Related Work
- Design
  - Syntactic & Semantic Composability
  - Component Representation
  - COSMO Ontology
  - Example
- Conclusions

Motivation

- Development of Modeling and Simulation
  - program-centric → model-centric → model sharing
  - model reuse to reduce simulation development time and cost
  - ...

- Simulation Environment
  - increasing trend to use Internet-based infrastructure for more pervasive sharing of resources – grid, p2p, web services, service-oriented architecture, etc.
  - advance simulation models and knowledge sharing on a larger and wider scale
  - quantum leap in capability and the size of the simulation applications
**Objective**

To develop methodology, abstraction and techniques to facilitate service-oriented simulation model sharing through *composable models.*

---

**Component-based Modeling & Simulation**

<table>
<thead>
<tr>
<th></th>
<th>Syntax</th>
<th>Semantics</th>
<th>Reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery</td>
<td>NP Complete problem</td>
<td>Component interoperability (data transfer, time assumptions)</td>
<td>“As-is” or as model components</td>
</tr>
<tr>
<td></td>
<td>Expose components as services</td>
<td>Loose coupling</td>
<td>Meaningful and stateful component behavior representation</td>
</tr>
<tr>
<td></td>
<td>Support complex queries</td>
<td>Algorithms for semantic checking</td>
<td></td>
</tr>
</tbody>
</table>
Related Work

- Current approaches can be classified as:
  - **Theoretical**
  - **Meta-modeling**
  - **Ontology**
    - DEMO (Discrete Event Modeling Ontology) (2005) – classifies simulations with respect to their particular worldview
    - PiMODES (2006) – focuses on models adhering to the process interaction worldview

Design of CODES

- Why CODES (Component-based Discrete-Event Simulation)?

  allows for the composition of simulation models out of geographically distributed, heterogeneous simulation components

- Heterogeneous:
  - Application domains
  - Simulation languages
  - ……
Component based Simulation Model Development

Semantic Composability

- A composition is *semantically valid* iif all connected components are aligned with respect to
  - **data exchange** – data passed between components is correct and meaningful to both parties
  - **behavior** – data exchange does not hinder the advancement of components towards a final state or a state that produces *valid output*
- Data exchange alignment
  - all neighboring components communicate meaningfully
- Behavior alignment
  - composition produces valid output
Simulation Model Reuse

- process of building or assembling applications from previously assembled parts designed for reuse

- Simulation model reuse
  - as a standalone simulator
  - as a model component interoperating with other model components in a larger simulation model
  - as base components

- 4 main stages
  - conceptual model creation, component discovery, component selection, model integration

Component Representation

\[ C_i = <R, A_i, B_i> \]

- \( R \) - mandatory attributes common to all components
- \( A_i \) – component specific attributes
- \( B_i \) – component behavior

- Attribute
  - has a name, a value and a description (optional)

- Behavior
  - External – constraints on the input and output Data
  - Internal - a finite state automaton
    - Transition between states is triggered by Input data
    - The next state can be decided by conditions
    - A transition can produce Output data
Component Representation

- Data
  - Constraints
    - Type – integer/string/etc
    - Range – in the form of minValue, maxValue
    - Origin – for input data, where the data came from
    - Destination – for output data, where the data is going to

- Constraint description is semantically enriched, with correspondents in the COSMO ontology

Component Behavior

\[ I_l, S_p, [\Delta t] \xrightarrow{Cond_m} S_r, [O_r], [A_m] \]

- \( I_l \) – the set of input data
- \( S_p \) – the current state
- \( \Delta t \) – the time it takes for the transition to finalize
- \( O_r \) – the set of output data
- \( A_m \) – the specific attributes that the transition changes
- \( Cond_m \) – the set of conditions that determine the next state
COSMO Ontology

- COmponent Simulation and Modeling Ontology
  - focuses on describing component-oriented simulation within and across application domains
  - Sets of classes to describe simulation components and the composition of simulation components
    - includes application domain taxonomies as well as component-oriented definitions
    - semantically enriches the description of model components to support model discovery, model reuse and semantic composability validation

COSMO Ontology (contd’)

```
Component
  - has_Attribute
  - is_a Base Component
  - has_Behavior
  - has_Worldview

Attribute
  - has_Data

Behavior
  - has_Input

Data
  - has_Output

Transition
  - has_Final_State

State
  - has_Conditions

Constraints
Type
  - Source
  - Destination

Worldview

Condition
```
Semantic Composability Validation

Conditions:
1. validate data alignment
   - constraint validation using the COSMO ontology
2. validate behavior alignment
   - verify that there exists a simulation run to produce the user specified validity points
Develop a simulator for a new application domain

1. Add the new application domain to the framework
   - Provide new composition grammar
   - Extend the COSMO ontology

2. Develop the conceptual model using the CODES GUI
   - Verify the conceptual model using the composition grammar

3. Discover the model components

4. Perform semantic validation

Queueing Networks Application Domain

Composition Grammar

```
CODES Composition Rules
Simulator ::= (Comp Con+)
Con ::= ConO | ConF | ConJ
B_Comp ::= QN_B_Comp

#Application Specific -- Queueing Networks (QN)
#Base Components
QN_B_Comp ::= Source | Server | Sink

#Composition Rules
QN_Simulator ::= Source BlockNT +Terminal? | Source BlockT+
Terminal ::= ConO Final IC ConF ("Final")* | ConF Final
Final ::= Source | Sink ...
```
Develop Conceptual Model

- Drag and drop base component icons on a drawing panel
- **Conceptual model**: no component implementation exists before discovery
- Syntactically verify production string with the composition grammar:

\[
\text{SimpleQN} = \text{Source ConO Server ConO Sink}
\]

---

Discover Component

- Assume each base component exists and has been discovered
- **Server1**:

<table>
<thead>
<tr>
<th>Mandatory Attributes</th>
<th>Specific Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: server1</td>
<td>servTimeMean:</td>
</tr>
<tr>
<td>type: Server</td>
<td>noJobsQueue:</td>
</tr>
<tr>
<td>version: 0.0.1</td>
<td>seed:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>State Machine</td>
<td></td>
</tr>
<tr>
<td>(S = {S_0, S_1, S_2, S_3, S_4})</td>
<td>Duration=[s]</td>
</tr>
<tr>
<td>(I = {I_{\text{server1}}})</td>
<td></td>
</tr>
<tr>
<td>(O = {O_{\text{server1}}})</td>
<td></td>
</tr>
<tr>
<td>(I_{\text{server1}} = \text{Job}(\text{type=double, range=10:35, origin=Source</td>
<td>Server}))</td>
</tr>
<tr>
<td>(O_{\text{server1}} = \text{Job}(\text{type=int, range=14:18, destination=Server</td>
<td>Sink}))</td>
</tr>
<tr>
<td>(C_1 = \text{noJobsQueue} = 0)</td>
<td>(C_2 = \text{noJobsQueue} = 0)</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
I_{S_1} & \rightarrow S_2 \\
S_2 & \rightarrow S_3 \\
I_{S_2} & \rightarrow [\text{noJobsQueue}++] S_3 \\
S_3 & \rightarrow O_{S_3} \\
S_3 & \rightarrow O_{S_3} \\
S_4 & \rightarrow [\text{noJobsQueue}++] S_2
\end{align*}
\]
Semantic Composability Validation

- Data alignment validation

- Validity points validation

- Returns the pairs

\[(O_{\text{server} 1}, VP_1), (I_{\text{server} 1}, O_{\text{server} 1}), (O_{\text{source} 1}, I_{\text{server} 1}), (-, O_{\text{source} 1})\]
Conclusions

- CODES – approach to component-based simulation covering component discovery & selection, syntactic & semantic composition, model reuse & integration

- Components are represented as a blackbox with attached meta-component describing their attributes and behavior

- Semantic composability validation is a two staged process: validates the data alignment between components + behavior alignment