Static and Dynamic Modeling and Analysis of Architectures

Roshanak Roshandel

Computer Science Department
University of Southern California
roshande@usc.edu
Motivation

- Software architecture’s promise
  - Architecture Description Languages (ADLs)
  - Formal Modeling Notations
- Static vs. Dynamic
- Narrow focus to provide sophisticated analysis capabilities
  - Wright, SADL
Static Modeling

- Represent system structure in terms of components, connectors and their interconnection
- Provide declarative behavior, interaction, and distribution models
  - Specify properties that must hold true at discrete points during the system’s life span
  - Allow analysis of functional and extra-functional (mis)match
  - Do not specify how the properties are to be achieved
- C2SADeL, UniCon, ACME
Dynamic Modeling

- Describes the details of interaction among components and connectors
- Provides imperative behavior, interaction, and distribution models
  - A continuous view of how to arrive at a given property or desired state
  - Allow analysis and simulation of dynamic behavior
  - Suffer from large state space and (often) implicit model of architectural structure
- StateCharts, PetriNets, Wright, Rapide
Coupling Static and Dynamic Modeling

- Expand the scope of architectural analysis
- Static $\rightarrow$ C2SADEL
  - Design the topology, separate concerns, specify behavioral “snapshots”
- Dynamic $\rightarrow$ StateCharts
  - Well known, powerful dynamic modeling technique
  - One of the modeling notations in UML
  - Extend FSM with concurrency and hierarchy
## Mapping

<table>
<thead>
<tr>
<th>C2SADEL</th>
<th>StateCharts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Composite State Machine</td>
</tr>
<tr>
<td>Components</td>
<td>State Machine</td>
</tr>
<tr>
<td>Connectors</td>
<td>State Machine</td>
</tr>
<tr>
<td>State variables</td>
<td>State’s variable</td>
</tr>
<tr>
<td>Invariants</td>
<td>Implicit in designing the states</td>
</tr>
<tr>
<td>Required Interface/Operation</td>
<td>Events</td>
</tr>
<tr>
<td>Provided Interface/Operation</td>
<td>Actions or events</td>
</tr>
<tr>
<td>Interface/Operation parameters</td>
<td>Parameters on transitions</td>
</tr>
<tr>
<td>Pre-conditions</td>
<td>Guards</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>State entry variable</td>
</tr>
</tbody>
</table>
Troops Deployment System
Repository

Operation
prov pAskMapInfo
prov pResultMap

StrategyAnalyzer

Operation
req rAskMapInfo
req rResultMap

connector1
Tool Support

- Integration of Argus-I (UC Irvine) and DRADEL (USC)
- Combine structural and static analysis with dynamic and run-time behavior modeling and analysis
Component StrategyAnalyzer is {
  state {   status: Integer;   theRow: Integer;   theCol: Integer;   theType: Integer;   valid: Boolean;   ...}
  invariant {   (status > 0) && (status < 4) && (theRow < 64) && (theCol < 64)   ...}
  interface {   req irMapInfo: MapInfo(id: String);   req irMapResult: MapInfo(): FILE;   prov ipUpdateData: UpdateData(row: Integer; col: Integer; type: Integer);   ...}
  operations {   req orMapInfo: {let i: STATE_VARIABLE; pre (i <> null);   req orMapResult: {let PH: STATE_VARIABLE; post (result = PH);   prov opUpdateData: {let r: Integer; c: Integer; t: Integer; post (theRow = r) && (theCol = c) && (theType = t);   ...}}
  map {   irGetMapInfo -> orGetMapInfo();   irGetMapResult -> orGetMapResult(id -> i);   ipUpdateData -> opUpdateData(row -> r, col -> c, type -> t);   ...   }
}

Component Repository is {
  state {   theMap: File;   theRow: Integer;   theCol: Integer;   theID: String;   ...}
  invariants {   (theRow < 64) && (theCol < 64)   ...}
  interface {   prov ipMapInfo: MapInfo(id: String);   prov ipMapResult: MapResult(): FILE;   ...}
  operations {   prov opMapInfo: {let i: String; pre (theID <> null);   prov opMapResult: {post (result = theMap);   ...}}
  map {   ipMapInfo -> opMapInfo(id -> i);   ipMapResult -> opMapResult();   ...   }
}
Summary

- Ability to analyze the behavior both statically and dynamically
- Consistency of architectural models and implementation level artifacts
- *Extensions*: Subtyping, Interaction protocols, Code generation
- The framework is being applied to JPL’s Mission Data System