Architectural Models at Runtime


Holger Giese
System Analysis & Modeling Group
Hasso Plattner Institute for Software Systems Engineering
at the University of Potsdam, Germany
holger.giese@hpi.uni-potsdam.de
Hypothesis

Linking the Architectural Model(s) and the Programming Language is not enough? You must also link them to the Runtime System!
**Why Runtime Representation?**

**Autonomic Computing/Self-Managed/Self-Adaptive Software:**

- Sensors and effectors usually at the abstraction level of APIs
  - Self-Management limited to simple parameters
  - The position of sensor/effectors in the architecture are not captured

**How can we do self-management also for dynamic architectures?**

- Architectural views on a managed elements is required?

⇒ **Runtime representation (model) of the architecture is required**

[Kephart and Chess, 2003]
One Architectural Model at Runtime

- Runtime model of a managed element: **Source Model**
- Model-based sensors and effectors for monitoring and adaptation

**Problem:**
- One *complex* model: types; deployed components and their configurations; concrete instances and interactions

**Observation:**
- Each self-* capability only requires its specific architectural view, but the autonomic managers have to cope with the whole application specific complexity
**Multiple Architectural Models at Runtime**

- **Target models**: higher level of abstraction and a specific view on a managed element
- Maintenance of target models by incremental and bidirectional model synchronization based on **Triple Graph Grammars (TGG)**
- Changes of the source model are reflected incrementally in target models (**monitoring**) and vice versa (**adaptation**)

[Vogel+2009a, Vogel+2009b]
Experiment with EJB and 3 Views

EJB Prototype
- EJB Glassfish application server + mKernel extension for monitoring
- Source models conform to EMF and is updated event-driven

View 1: Architectural constraints for self-configuration
- Simplified runtime architectures of EJB-based applications for checking architectural constraints using OCL

View 2: Performance data for self-optimization
- Architectural information enriched with performance data

View 3: Failure data for self-healing
- Architectural information enriched with occurred failures
## Efficient enough?

<table>
<thead>
<tr>
<th>Size</th>
<th>Model-Driven Approach</th>
<th>Non-inc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n=0</td>
<td>n=1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>163</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>157</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>178</td>
</tr>
</tbody>
</table>

| Event processing | 0% | 92.8% | 94.1% | 95.6% | 95.2% | 96.3% | -     |
| Synchronization  | 0% | 7.2%  | 5.9%  | 4.4%  | 4.8%  | 3.7%  | -     |

### Performance:
- **Size**: number of deployed beans
- **Processing**: $n$ events and invoking once the transformation engine
- **Problem**: glue code to mKernel results in enormous overhead

⇒ direct support for architectural representation would boost performance
Conclusion & Implication

Future Work/Open Points

- Architectural adaptations (even in case of abstraction)
- Dependencies among views
- Coordinate autonomic managers
- Distributed setting

Observations

- Similar views are also required for software maintenance activities
- The Source model is platform specific while target models are often platform independent (reuse!)
- Generic EJB + mKernel based approach results in unnecessary overhead

Implication

- Efficient support for architectural information at runtime is required!

Options:

a) Generating application specific code that directly provides an interface in form of a source model

b) Programming language support for architectural concepts and efficient reflection (with write capabilities)
The Big Picture

Development/Maintenance:
Model → System models ➔ Architecture models ➔ Design models ➔ Implementation models ➔ Deployment models

Runtime:
Deployment models ➔ Architecture runtime models ➔ Design runtime models ➔ Implementation runtime models

Reflection+
Support for associations
Presented

Spectrum
PL

11/2009 | Architectural Models at Runtime

