The Architecture of a Dynamically Updatable, Component-Based System

Robert P. Bialek

Department of Computer Science
University of Copenhagen

COMPSAC 2002
Outline

- Background
- Kinds of updates
- DUCBS
- Update Request
- Examples
- Conclusions
- Issues
- Future work
Background

Updates perspectives:

– Implementation perspective: easiness of implementing the changes
  
  CBM – framework for updatable system
  
  – encapsulation
  
  – Isolation
  
  – DA -> (Packages/Services) -> Components -> Objects -> Methods/Data

– Distributed perspective: Reconfiguration
  
  DA = set of components + connections (configuration represented in ADL / IDL)

  Addition/removal/update of component

– Runtime perspective
## Kind of updates 1
### Runtime perspective

<table>
<thead>
<tr>
<th></th>
<th>Dynamic (in use)</th>
<th>Static (not used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-less</td>
<td>(Dynamic linking)</td>
<td>replace -&gt; restart</td>
</tr>
<tr>
<td></td>
<td>If dynamic then must be statefull!</td>
<td></td>
</tr>
<tr>
<td>State-full</td>
<td>State transfer function</td>
<td>Checkpoint -&gt; replace -&gt; recreate state -&gt; restart</td>
</tr>
</tbody>
</table>
Kinds of updates 2
Runtime-Distributed perspective

- Local updates
  - exchanging objects not connected to the interface

- Global
  - Exchanging object connected to the interface
    - Requires maintaining the communication channels
How to handle that many updates?

- Any of these updates are possible …
- Unanticipated updates
- Propose an architecture called DUCBS 😊
DUCBS

- Separation between application semantics and update semantics
- Meta level communication – asynchronous
- Connectors support synchronous communication
- Space for authentication
Update Request

- **update type** – addition/removal/update
- **list of updated objects**
- **new versions of the objects** - implementations
- **update method** for each object: replace/dynamic
- **update function** - state transfer function
- **update constrains** – order/relation of (sub)updates

```
<update_descriptor>
  <add_obj to="server01//compl">
    <object name="C">
      <implementation>...</>
    </object>
  </add>
  <remove_obj from="server01//compl">
    <object name="D">
      </object>
  </remove>
  <update_obj in="server01//compl">
    <object name="A" method="replace">
      <old_version>1.0</>
      <new_version>1.1</>
      <implementation>...</>
    </object>
  </update>
  <update_obj in="server01//compl">
    <object name="B" method="dynamic">
      <old_version>1.1</>
      <new_version>1.2</>
      <update_function>...</>
      <implementation>...</>
    </object>
  </update>
</update_descriptor>
```
Example 1. Update of a local component
Example 2. Update of a global component
Conclusion

- Multi-layered UDCBM
- Handles many kinds of updates by:
  - Dynamic: using update function
  - Distributed: propagating updates using ADL/IDL
- Separation of application and update logic
- ”Space” for future improvements security/control methods
- No restrictions for the update logic / process / protocol
Issues

- CBM: EJB, CORBA

- Reflection - Java reflection too primitive
  - Unload class missing
  - Method’s state missing
  - JVM separate language / byte code
Future work

- Prototyping
  - CBM architecture
  - Reflection issues
- Update request/process
  - Granularity (relations between updated parts)
  - Order