ReflectAll

Combining Reflective Component Model and Reflective Middleware

Gang HUANG, Ling LAN, Jie YANG, Hong MEI

School of Electronics Engineering and Computer Science
Peking University, Beijing, 100871, China
July 3, 2006, Nantes, France
Software Engineering Institute in Peking Univ.

- 7 full professors, 10 associate professors, 9 assistant professors
- >30 Ph.D students, >70 graduates
  - The biggest SE team in Chinese universities
- Cover almost all areas of software engineering with emphasis on component based reuse
  - domain engineering, object oriented modeling, software architecture, middleware, component repository, testing, program comprehension, software process
- http://www.sei.pku.edu.cn
Background of This Work

Component Model

- Fractal & ABC
  - Software architecture group is the core group of ABC
  - ABC/ADL & ABCTool
  - 7 PhD students, 6 graduates

Next Generation J2EE

- JonAS & PKUAS
  - PKUAS group is the biggest group in SEI@PKU
  - 7 PhD students, 5 graduates in experience sub-group
  - >20 graduates in practice sub-group

Autonomic System Management

- JADE & ABC/PKUAS
Agenda

- Motivation
  - Why leverage reflective component and reflective middleware
- Prototype and Demo
  - Prototype on J2EE (PKUAS & JonAS)
  - Demo of JPS: Password Protection
- Lessons Leant
  - Fractal v2 controllers are not sufficient & necessary
  - Evolution other than revolution to reflection
  - Managing reflective systems in the whole lifecycle
- Conclusion and Future Work
Reflection

- Also known as computational reflection, is originated by B.C. Smith to access and manipulate the LISP program as a set of data in execution.

- As a promising way to achieve high adaptability, reflection is propagated into more programming languages, operating systems and distributed systems, and so on.
  - 3-KRS, Prolog, CLOS, Smalltalk, Java, C# …
  - Apertos, MetaOS, 2K …
  - CodA, GARF …

- Component based systems also need reflection
**Reflection in Component based Systems**

<table>
<thead>
<tr>
<th>Component</th>
<th>Business</th>
<th>Reflection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflective Component</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reflective Component Model:**
What is a reflective component
e.g. Fractal, OpenORB, K-Component

**Middleware for Reflection:**
How can a component be reflective
e.g. Julia, AOKell, OpenCOM

**Reflective Middleware:**
Making traditional middleware reflective
e.g. OpenCORBA, dynamicTAO,
FlexiNET, MChaRM, PKUAS

*Different with middleware for reflection though some functions are similar*
Reflective Component vs. Reflective Middleware

Reflective Component:
+ Formal programming model
± Easy to understand (by application developers)
± Encourage as well as rely on application developers for implementing reflection
- Poor monitoring and controlling outside of components

Reflective Middleware:
- Ad hoc programming model
± Easy to understand (by middleware vendors)
± Release as well as prevent application developers from reflection impl
+ Well monitoring and controlling outside of components

Unfortunately, neither of them is sufficient for popular application of reflection
Recap of Reflection’s Promise

- Reflection is a promising way to achieve high adaptability
  - Everything in a runtime system may be to change
    - Reflective component cannot change middleware and vice versa
  - Everything is changed by a condition at a time
    - Different changes may be understood from different views (application or middleware)

- Usability is a key to practice of new technology
  - Easy to use (programming model of reflective component)
  - Easy to reuse (reusable functions of reflective middleware)

- It’s the time to combine RC & RM
Goal of ReflectAll

- Demonstrate the combination of RC & RM
  - The combination is feasible
    - Reflective component & reflective middleware can be combined
  - The combination is promising
    - Keep the advantages while remove the disadvantages

- Review existing RC & RM
  - Limitation
  - Killer application
Overview of ReflectAll

ABCTool:
Architectural model driven engineering tool covering the whole lifecycle of component based systems

Fractal Programming Model

RSA for PKUAS
Prototype in Feb.

RSA for JonAS
Prototype in June

Runtime Software Architecture:
Reference model for architecture based reflective middleware

ReflectAll: Server Level Architecture

- Leveraging reflective component and reflective middleware for reflecting all things in a component based system.
ReflectAll: Container Level Architecture

- All things can be reflected by the collaboration between middleware vendors and application developers

**Client Side**
- EJB Client
- EJB Interface
- EJB Stub
- Fractal Client
- Fractal Interface
- Fractal Impl

**Server Side**
- EJB Interface
- Reflective Interface
- Application Defined
- Middleware Defined
- EJB Impl

**Dynamic AOP**

**Extension Points of RM**

Meta Objects by App Developers

Meta Objects of Reflective Middleware

AOP is not enough for reflection
Demo of JPS: Password Protection

- Change JPS at runtime without any modification to the source code

- **Four steps**
  - Opening the design artifacts of the application to be managed
  - Incarnating the runtime software architecture
  - Customizing the reflective components when necessary
  - Managing the runtime system

- **JonAS Demo will be published in ObjectWeb**
  - Modified JonAS v4.7.1
  - Source code of controllers, JPS deployable package
  - ABCTool English version
Lessons Learnt

- Fractal v2 controllers are not sufficient & necessary
  - Binding controller and some of the following controllers are useless in some cases
  - Controllers should be customizable at runtime
  - Connectors may be complex and need to be reflective

- Evolution other than revolution to reflection
- Managing reflective systems in the whole lifecycle

<table>
<thead>
<tr>
<th></th>
<th>Specific to</th>
<th>Already implemented?</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in</td>
<td>middleware</td>
<td>Yes</td>
<td>Attribute controller</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lifecycle controller</td>
</tr>
<tr>
<td>Pre-defined</td>
<td>middleware</td>
<td>Yes but need</td>
<td>Persistence controller</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration</td>
<td>Polymorphism controller</td>
</tr>
<tr>
<td>User-defined</td>
<td>application</td>
<td>Not yet but</td>
<td>The two controllers in JPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reusable</td>
<td>demo</td>
</tr>
</tbody>
</table>
Lessons Learnt

- Fractal v2 controllers are not sufficient & necessary
- **Evolution other than revolution to reflection**
  - Legacy systems cannot be ignored
  - Reflective mechanisms can be added one by one
- Managing reflective systems in the whole lifecycle
Lessons Learnt

- Fractal v2 controllers are not sufficient & necessary
- Evolution other than revolution to reflection

Managing reflective systems in the whole lifecycle

- ABC: architectural model driven approach
Conclusion & Future Work

- Combination of reflective component and reflective middleware is necessary, feasible and promising
  - Demonstration on J2EE (PKUAS & JonAS)

- Combination identifies some future directions
  - A more flexible reflective component model
  - An evolutionary way to reflective systems
  - An architectural model driven approach to systematic use of reflection
  - In particular, deeper collaboration between PKU & ObjectWeb

- [http://www.sei.pku.edu.cn/~huanggang/](http://www.sei.pku.edu.cn/~huanggang/)
Thanks