Components in an Adaptive and QoS-based Architecture

Claudia Raibulet, Francesca Arcelli, Mussino Stefano, Mario Riva, Francesco Tisato, Luigi Ubezio

Università degli Studi di Milano-Bicocca
DISCo – Dipartimento di Informatica Sistemistica e Comunicazione
Milan, Italy

SEAMS 2006
ICSE 2006 Workshop on Software Engineering for
Adaptive and Self-Managing Systems
Shanghai, China
Issue

- Today information systems are composed of various entities, which can provide similar or identical services

- How to:
  identify
  choose
  exploit

  at runtime the entity able to provide the desired service, the one that suits best for the current request?

=> Adaptive Resource Management (ARM)
Solution

- Explicitly represent and exploit additional information about the entities providing the services:
  - qualities, location, cost
  - structural, topological information
  - …

  to adapt dynamically to the users’ requests

- Examples of common scenarios:
  - print on the nearest A3-format printer
  - display an image on a monitor supporting a specific resolution
  - send this message using the most appropriate device and network connectivity
  - …
Architectural Reflection

- Reflection enables a system to observe and control itself through meta-representations
- Architectural reflection introduces:
  - additional components within the logical layer
  - additional layer(s) between the application and the logical layer
- Reflective components/layers are causally connected to logical components/layers
ARM Architecture
Reflective Layers

- Application
  - Service Manager
  - Executor Manager
    - Functional Object
  - Reflective Object
    - Causal Connection

- Services
  - Network Manager
  - Reflective View Manager
    - Service View
    - QoS
    - Reflective Property
    - Views

Extended Reflective Layer
Base Reflective Layer
Representation of the Reflective Knowledge

- Reflective objects represent non-functional aspects of the systems components
- R_Objects are characterized by their related:
  - QoS
  - Properties
A view = an organizational structure on the reflective objects with its own semantics and computational strategies to evaluate the objects under its control.
The Service View

- Catalogs the resources based on the services they provide
Location, Topology, Structural, ... Views

Diagram:
- Peer 1
- Peer 2
- Peer 3
- Peer 4
- Peer 5

Connections:
- a: Peer 1 to Peer 4
- e: Peer 2 to Peer 4
- f: Peer 2 to Peer 5
- g: Peer 4 to Peer 5

Structural Views:
- Pc
- Case
- Modem
- Cpu

Table:

<table>
<thead>
<tr>
<th>Latitude</th>
<th>Longitude</th>
<th>Address</th>
<th>Building</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Causal Connection Mechanism
Observer Design Pattern

The causal connection between functional and reflective objects is implemented by:

- **update()** - adapts reflective objects to functional objects
- **force()** - adapts functional objects to reflective objects
The Chain of Services Views
Chain of Responsibility Design Pattern
Strategies

Strategy and Composite Design Patterns

- Strategies implement decisions
- A best effort strategy chooses the most appropriate resource for the current service request

- Strategies may be simple or compositions of other strategies

- Strategies implement decisions
- A best effort strategy chooses the most appropriate resource for the current service request

- Strategies may be simple or compositions of other strategies
Validation of ARM

- A totally distributed solution exploiting the P2P paradigm (JXTA)

- Three types of requests:
  - Non adaptive – specify the service and the input data
  - Low-adaptive – specify the service, the entity that should execute the service and the required QoS
  - High-adaptive – specify the service and the QoS

- Currently, two services:
  - display
  - print
  - …
Conclusions (I)

- ARM proposes a solution to address dynamic adaptivity through architectural reflection

- Representation of the reflective knowledge:
  - reflective objects and QoS
  - properties:
    - model important aspects of the system (structural, topological, location, etc.) for achieving adaptivity
    - allow an efficient organization and management of the reflective knowledge

- Management of the reflective knowledge:
  - views
  - strategies: implement decision support

- Validation of ARM concepts through a prototype
Conclusions (II)

- **Disadvantages of architectural reflection:**
  - significant increase of the number of software components which may reduce overall efficiency
  - modification of the reflective components may cause overall damages

- **Advantages of architectural reflection in ARM:**
  - separation of concerns: functional vs. reflective objects
  - modularity: separation of the reflective entities from their management mechanisms
  - improving maintainability of the overall architecture
  - reusability and extensibility both of individual components and of the entire approach in various application domains
Further Work

- Allocation and negotiation of resources
- Extension of our approach to mobile devices which are not characterized by high computational capacities (e.g., mobile phones)
- Extension of our current approach with other services
- Using ARM in various application domains