Aspect Formulation for Developing Small-Footprint CORBA Event Channels

http://www.cs.wustl.edu/~doc/RandD/PCES/facet/

Ravi Pratap, Frank Hunleth and Ron K. Cytron

Center for Distributed Object Computing
Department of Computer Science
Washington University

Sponsored by DARPA under contract F33615-00-C-1697
Traditional Middleware Development

• First release
  – Clean and well decomposed design – minimal set of features

• Subsequent releases
  – More features are added to the core
  – As core bloats, functionality is re-factored into features

• Problems
  – Programs typically need only a subset of the features
  – Re-factoring can be difficult and time consuming
  – Hooks need to be scattered throughout the code

• Case study
  – ACE and TAO
Separation of Concerns with Aspects

- Concerns are logically related areas of a design
- Object Oriented Programming
  - Class based decompositions
  - Not all types of concerns can be decomposed
    - Synchronization, memory management
    - Crosscutting features and functionality
- Aspect Oriented Programming (AOP)
  - Concerns are encapsulated into *Aspects*

```java
package edu.wustl.doc.facet.tracing;

import edu.wustl.doc.facet.*;
import org.apache.log4j.Category;
import org.apache.log4j.NDC;
import org.aspectj.lang.Signature;

/**
 * Enabling this aspect weaves in tracing code throughout the event channel. The tracing is done using the Log4J logger. A good pattern for printing out the trace messages is "%-4r [%t]%x%m%n"
 */

aspect EventChannelTraceAspect {
    protected static Category cat = Category.getInstance(EventChannelTraceAspect.class);
    protected void enterMethod(Signature signature) {
        /* Update the indentation for tracing in this thread. */
        NDC.push(" ");
        /* Log the message */
        cat.debug("Entering " + signature);
    }
    protected void exitMethod(Signature signature) {
        /* Update the indentation for tracing in this thread. */
        NDC.pop();
        /* Log the message */
        cat.debug("Exiting " + signature);
    }

    // Define pointcuts...
}
```
A Compositional Approach Using Aspects

• Develop a simple base
  – Contains minimal code common to all configurations

• Add functionality with features
  – Code is introduced using aspects
  – User selectable

• Results
  – What You Want Is All You Get – no code bloat or performance degradation due to redundant features
  – Flexible and simple – no need to strategize base code
FACET

- Framework for Aspect Composition for an Event channel
- Modeled after features from the TAO Real-time Event Service

Features
- Payload choices
- Filtering
- Correlation
- Pull semantics
- Federation
- Policing
- Timeouts
- Event Scheduling
  - RMS, EDF, etc.
The Event Channel Base

• Provides a basic CORBA “Interrupt” Service

• Contains
  – IDL specifications for simple “push” consumers and suppliers
  – No event data is passed
    • push () method call takes no arguments
  – Event channel forwards all events to all consumers
    • No filtering
FACET Feature Dependence Graph

Automatic testing and validation through build system – 9192 valid combinations out of the $2^{21}$ possibilities
Common Event Channel Configurations

- TAO-users poll was conducted

<table>
<thead>
<tr>
<th></th>
<th>Tracing</th>
<th>Any Payload</th>
<th>Octet Seq Payload</th>
<th>Event sets</th>
<th>TTL</th>
<th>Time stamp</th>
<th>Type filtering</th>
<th>Event correlation</th>
<th>“Pull” Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config 1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config 3</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config 5</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Config 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Config 7</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Config 8</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Library Footprint

FACET Footprint vs. Configuration

Bytes

Configuration

OctetSeq payload/Correlation

Base Only

Class Files
GCJ Object Files

Ravi Pratap M
Footprint Results

Average Feature Sizes

Key
A: Simple event filtering
B: Event correlation
C: Event Set support
D: Event Pull support
E: Octet sequence payload
F: Any payload
G: String payload
H: CORBA Any events
I: Calculated: Event header
J: Event type
K: Timestamp
L: Time to live field and processing
M: Profiling support
N: Event statistics counters
Throughput for Common Configurations

Event Throughput vs. Configuration

Percent of Base Throughput

Configuration

0 1 2 3 4 5 6 7 8
Feature Performance Results

Average Feature Effect on Raw Throughput

Key
A: Simple event filtering
B: Event correlation
C: Event Set support
D: Event Pull support
E: Octet sequence payload
F: Any payload
G: String payload
H: CORBA Any events
I: Calculated: Event header
J: Event type
K: Timestamp
L: Time to live field and processing
M: Profiling support
N: Event statistics counters
Additional Code From External Libraries

Additional External Library Footprint vs. Configuration

Bytes vs. Configuration

Configuration

0 1 2 3 4 5 6 7 8

Additional External Library Footprint

0 50000 100000 150000 200000 250000 300000 350000 400000
Non-CORBA Contexts

• Boeing’s *Bold Stroke* architecture
  – Requires complex features – filtering, correlation etc.
  – Needs events to be dispatched in real-time based on various parameters
  – JacORB is “too heavy” and not decomposable

• FACET was found to be ideal for *Bold Stroke*
  – Real-time event dispatching added through a feature since the dispatcher is pluggable
  – Simple Java objects in place of CORBA objects
Current Work

• FACET needs to be used in wide contexts
  – Make the use of CORBA optional by making it a feature
  – Enable support for CORBA replacements – Java RMI for example

• Challenges
  – Aspects cannot be applied to CORBA IDL
  – Using CORBA’s standard naming conventions to apply advice
  – Ensure there is no predication towards CORBA – something which currently exists
Conclusions

- Highly customizable CORBA event channel
- AOP is a novel and practical approach to building customizable middleware
- Footprint and throughput results support the use of AOP techniques
More Information

• FACET is available from :

• AspectJ – AOP bindings for Java :
  – http://www.aspectj.org

• The DOC Group at Washington University
Thanks

• Advisor - Dr. Ron K. Cytron

• Original FACET Author - Frank Hunleth

• The DOC Group at Washington University

• The audience for bearing with me!