Context-oriented Programming for
Software Variability at Runtime

Robert Hirschfeld
Hasso-Plattner-Institut
hirschfeld@hpi.uni-potsdam.de

svpp 2008, Brussels, Belgium
August 8, 2008
Background

• 1994-1997
  – Mercedes-Benz & Daimler-Benz
  – Software Architecture

• 1997-2001
  – Windward Solutions
  – Distributed Processing Environments
  – CORBA & TINA

• 2001-2006
  – NTT DoCoMo Euro-Labs
  – Dynamic Aspect-oriented Programming
  – Dynamic Service Adaptation

• Since 2006
  – Hasso-Plattner-Institute, Software Architecture Group
  – Context-oriented Programming
  – Dynamic Programming Environments
Outline

Software Architecture

Distributed Processing Environments

Dynamic Aspect-oriented Programming

Reflective Designs

Dynamic Service Adaptation

Context-oriented Programming

Context-oriented Programming @ svpp 2008

Robert Hirschfeld (www.swa.hpi.uni-potsdam.de) 2008
Component Connection Layers

- **Interface-connection architecture (ICA)**
  - Architecture description languages (ADLs)
  - Structure declared by interfaces, not their implementations

- **Object-connection architecture (OCA)**
  - Structure determined by implementation

- **Component connection layer (CCL)**
  - Framework for ICA in OCA-based systems

CCL (Mercedes-Benz & Daimler-Benz)
module Example {
  group GroupA {
    components ObjectA, GroupB;
    contracts InterfaceB, InterfaceD;
  };
  object ObjectA {
    behavior behaviorText "This object does something useful";
    requires InterfaceD;
    supports InterfaceA, InterfaceB, InterfaceC;
  };
  interface InterfaceA {};
  interface InterfaceB {};
  interface InterfaceC {};
  group GroupB {
    components ObjectB;
    contracts InterfaceD;
  };
  object ObjectB {
    behavior behaviorText "This object does something useful, too";
    supports InterfaceD, InterfaceE;
  };
  interface InterfaceD {};
  interface InterfaceE {};
};
Aero: Dynamic Composition

backups and alternatives

Robert Hirschfeld (www.swa.hpi.uni-potsdam.de) 2008
Outline

1. Software Architecture
2. Distributed Processing Environments
3. Dynamic Aspect-oriented Programming
   - Reflective Designs
4. Dynamic Service Adaptation
5. Context-oriented Programming
AspectS
Dynamic Aspect-oriented Programming
myObject myProxy myRealSubject |
myObject ← MySampleClass new.  
{ myObject returnSeven. myObject returnThree }  \rightarrow \{ 7. 3 \}  
myRealSubject ← MyRealSubjectClass new.  
{ myRealSubject returnSeven. myRealSubject returnThree }  \rightarrow \{ \text{‘seven’. ‘three’} \}  
myProxy ← RdProxyAspect new.  
myProxy proxy: MySampleClass selectors: \{ \text{#returnSeven} \}.  
myProxy activate.  
{ myObject returnSeven. myObject returnThree }  \rightarrow \{ 7. 3 \}  
myProxy addSubject: myObject realSubject: mySubject.  
{ myObject returnSeven. myObject returnThree }  \rightarrow \{ \text{‘seven’. 3} \}  
myProxy deactivate.  
{ myObject returnSeven. myObject returnThree }  \rightarrow \{ 7. 3 \}  

Proxy: Provide a surrogate or placeholder for another object to control access to it.

Reflective Designs (DoCoMo Euro-Labs)
Outline

Software Architecture

Distributed Processing Environments

Dynamic Aspect-oriented Programming

Reflective Designs

Dynamic Service Adaptation

Context-oriented Programming

Robert Hirschfeld (www.swa.hpi.uni-potsdam.de) 2008
Deployment Scenarios

Client-Side Adaptation Manager
- DSA utilities
- DSA base
- Squeak base & mop
- Virtual machine

Server-Side Adaptation Manager
- DSA utilities
- DSA base
- Squeak base & mop
- Virtual machine

Tetris usage indication
- FaurePDA

Cube rendering fix
Cube branding
Tetris integration
LogServer port change

FaurePDA

Tetris usage posts

Dynamic Service Adaptation (DoCoMo Euro-Labs)
UI Branding and Bugfixing…
Service Integration...
...Usage Indication
Outline

Software Architecture

Distributed Processing Environments

Dynamic Aspect-oriented Programming

Reflective Designs

Dynamic Service Adaptation

Context-oriented Programming
Context

context = everything computationally accessible

- location
- time of day
- temperature
- connectivity
- bandwidth
- battery level
- subscriptions
- preferences
- energy consumption
- battery level
- age
- mood
MVC

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad: Pattern-Oriented Software Architecture – A System of Patterns. John Wiley and Sons 1996
Increased Complexity

- Person
- Attributes
- Attributes
Increased Complexity
Partial Layer and Class Definitions

- Layer 1
- Layer 2
- Layer n

Classes:
- Class 1
- Class 2
- Class m

Behavioral variation:

Active layer:

Context:

- Introduce
- Activate
- Deactivate
- Remove
COP Basics

- **Behavioral variations**
  - Partial class and method definitions

- **Layers**
  - Groups of related context-dependent behavioral variations

- **Activation**
  - Activation and deactivation of layers at runtime

- **Context**
  - Anything computationally accessible

- **Scoping**
  - Well-defined explicitly-controlled scopes
Dynamically-scoped Layer Activation

- Constructs
  
  \( (\text{with-active-layers} \ (\ldots) \ \ldots) \) \( \text{ContextL} \)

  \( (\text{with-inactive-layers} \ (\ldots) \ \ldots) \)

  \([\ldots] \text{useAsLayersFor:} \ [\ldots] \) \( \text{ContextS} \)

  \( \text{with} \ (\ldots) \ \{\ldots\} \) \( \text{ContextJ} \)

- Activate (deactivate) layers for the current thread
  
  – Does not interfere with other layer activations/deactivated in other threads

- Layers are activated/deactivated only for the dynamic extent of the associated code block

- Activation order determines method precedence
Demo

squeak
## AOP vs. FOP vs. COP

<table>
<thead>
<tr>
<th></th>
<th>AOP</th>
<th>FOP</th>
<th>COP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverse dependencies</td>
<td><img src="circle.png" alt="Circle" /></td>
<td><img src="circle.png" alt="Circle" /></td>
<td><img src="circle.png" alt="Circle" /></td>
</tr>
<tr>
<td>1:n relationships</td>
<td><img src="circle.png" alt="Circle" /></td>
<td><img src="circle.png" alt="Circle" /></td>
<td><img src="circle.png" alt="Circle" /></td>
</tr>
<tr>
<td>Layers</td>
<td></td>
<td><img src="circle.png" alt="Circle" /></td>
<td><img src="circle.png" alt="Circle" /></td>
</tr>
<tr>
<td>Dynamic activation</td>
<td></td>
<td><img src="circle.png" alt="Circle" /></td>
<td><img src="circle.png" alt="Circle" /></td>
</tr>
<tr>
<td>Scoping</td>
<td><img src="circle.png" alt="Circle" /></td>
<td><img src="circle.png" alt="Circle" /></td>
<td><img src="circle.png" alt="Circle" /></td>
</tr>
</tbody>
</table>

- **AOP**
  - Inverse dependencies
  - 1:n relationships
  - Layers
  - Dynamic activation
  - Scoping

- **FOP**
  - Inverse dependencies
  - 1:n relationships
  - Layers
  - Dynamic activation
  - Scoping

- **COP**
  - Inverse dependencies
  - 1:n relationships
  - Layers
  - Dynamic activation
  - Scoping

**Notes:**
- First-class layers
- Explicit meta-objects
- Scoped adaptation
- Improved comprehension
COP Implementations

- ContextL (VUB/PROG)
- ContextS (HPI/SWA)
- ContextJ* (VUB/PROG)
- ContextR (HPI/SWA)
- ContextPy (HPI/SWA)
- ContextJ (HPI/SWA)
- ContextG (HPI/SWA)
- PyContext (HPI/DCL)
- Context# (HPI/DCL)
- ...
Collaborators

• Pascal Costanza
  – Programming Technology Lab (PROG)
  – Vrije Universiteit Brussel (VUB)

• Oscar Nierstrasz
  – Software Composition Group (SCG)
  – University of Bern

• Michael Haupt
  – Software Architecture Group (SWA)
  – Hasso-Plattner-Institut (HPI)

• Hans Schippers
  – Formal Techniques in Software Engineering Group (FoTS)
  – University of Antwerp

Robert Hirschfeld (www.swa.hpi.uni-potsdam.de) 2008
Papers and Downloads

http://www.swa.hpi.uni-potsdam.de/cop/
Context-oriented Programming for Software Variability at Runtime

Robert Hirschfeld
Hasso-Plattner-Institut
hirschfeld@hpi.uni-potsdam.de

svpp 2008, Brussels, Belgium
August 8, 2008