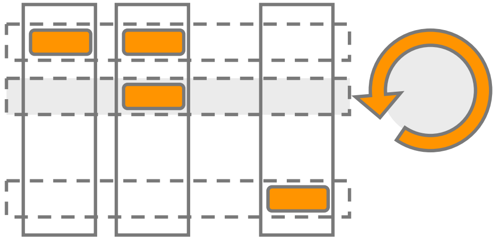


Context-oriented Programming for Software Variability at Runtime

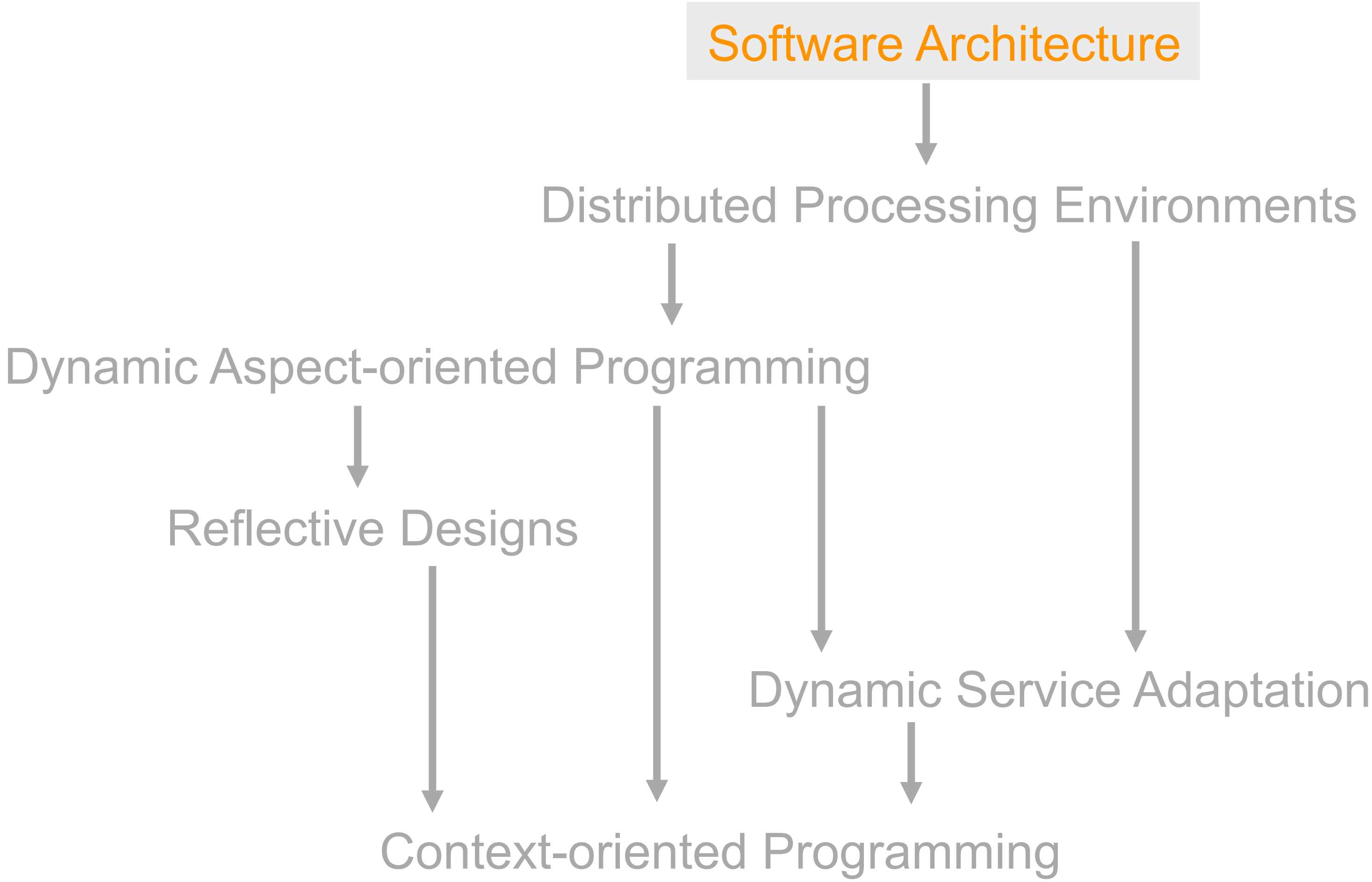
Robert Hirschfeld
Hasso-Plattner-Institut
hirschfeld@hpi.uni-potsdam.de

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

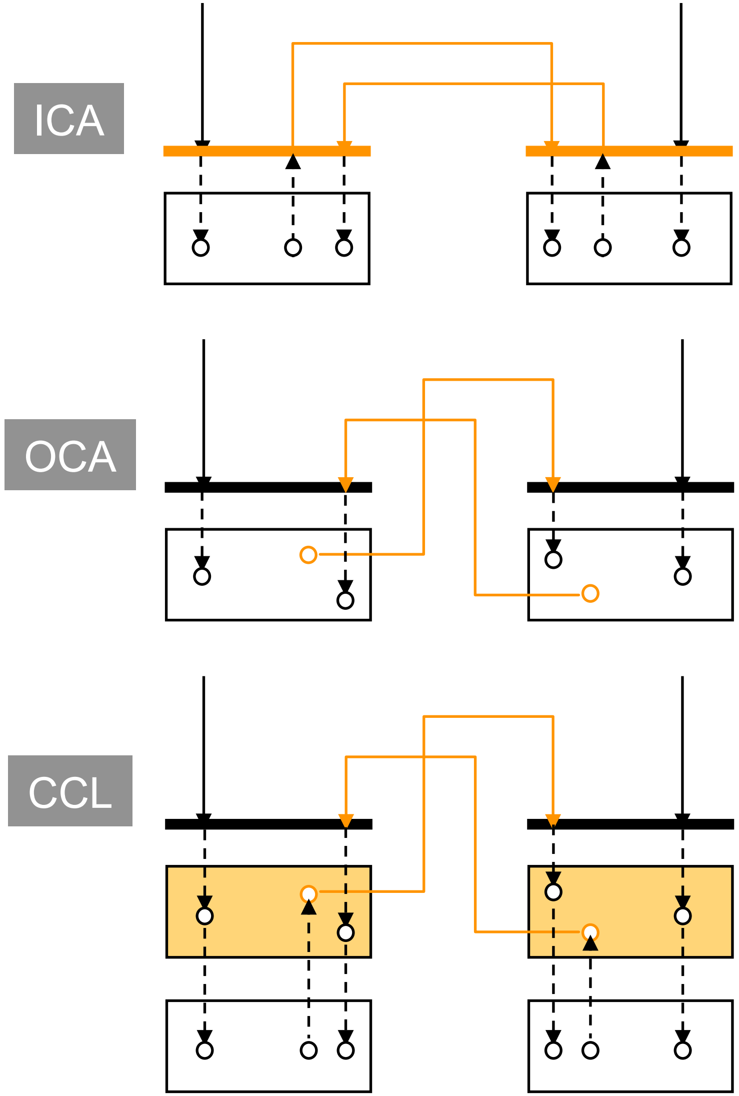
Dynamic Aspect-oriented Programming

Reflective Designs

Dynamic Service Adaptation

Context-oriented Programming

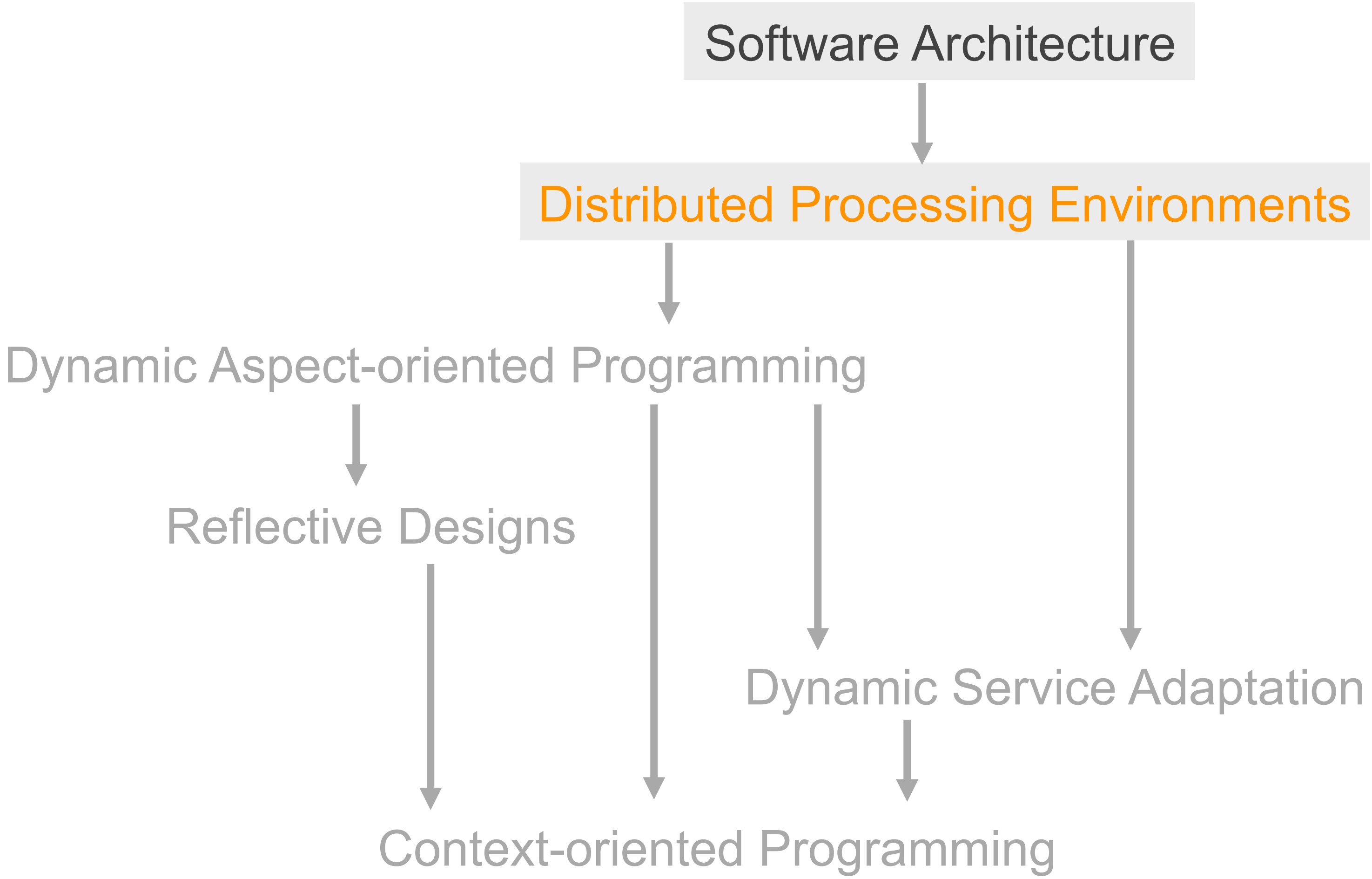
Component Connection Layers



- 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)

Outline



Software Architecture

Distributed Processing Environments

Dynamic Aspect-oriented Programming

Reflective Designs

Dynamic Service Adaptation

Context-oriented Programming

Aero: Dynamic Composition

ODL module description

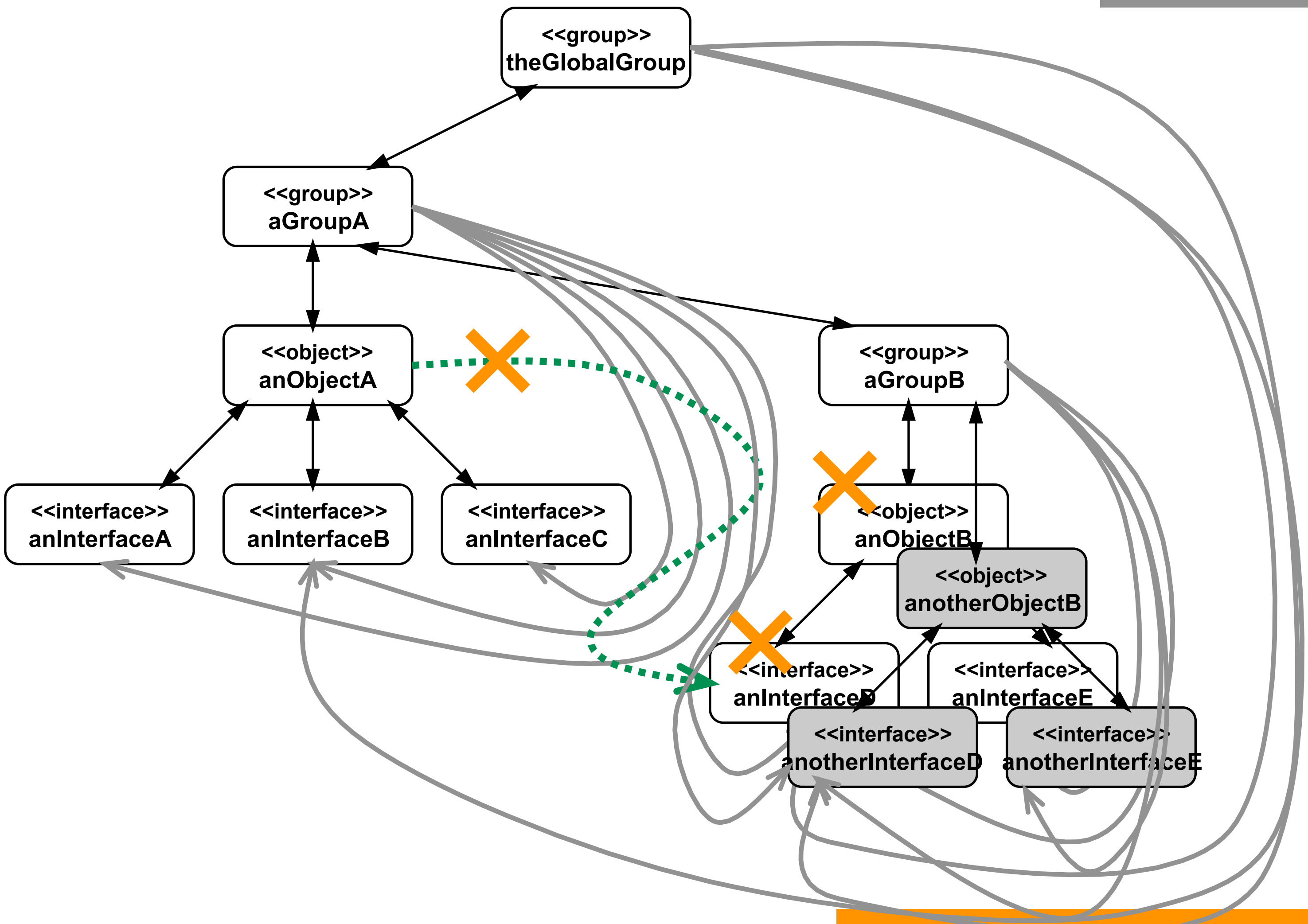
object definition language

```
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 (Windward Solutions, Sunnyvale, California)

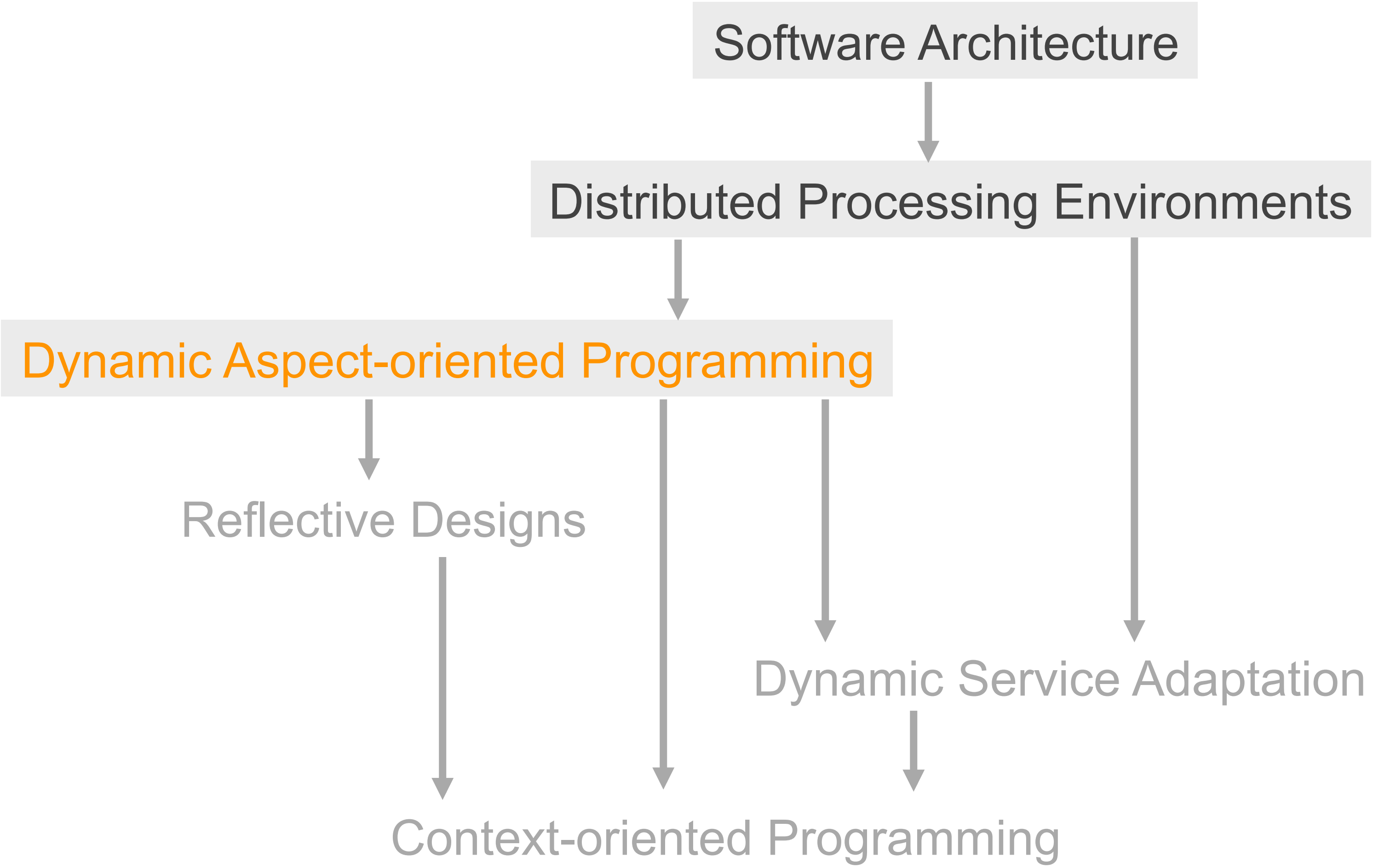
Aero: Dynamic Composition

backups and alternatives



Aero (Windward Solutions, Sunnyvale, California)

Outline


Software Architecture

Distributed Processing Environments

Dynamic Aspect-oriented Programming

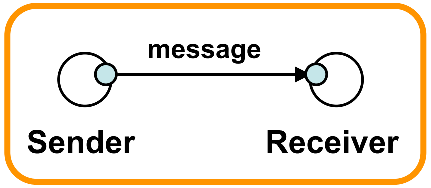
Reflective Designs

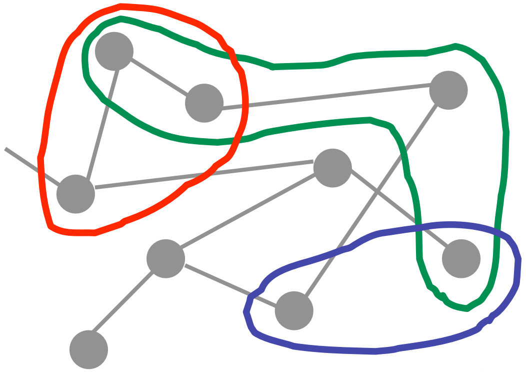
Dynamic Service Adaptation

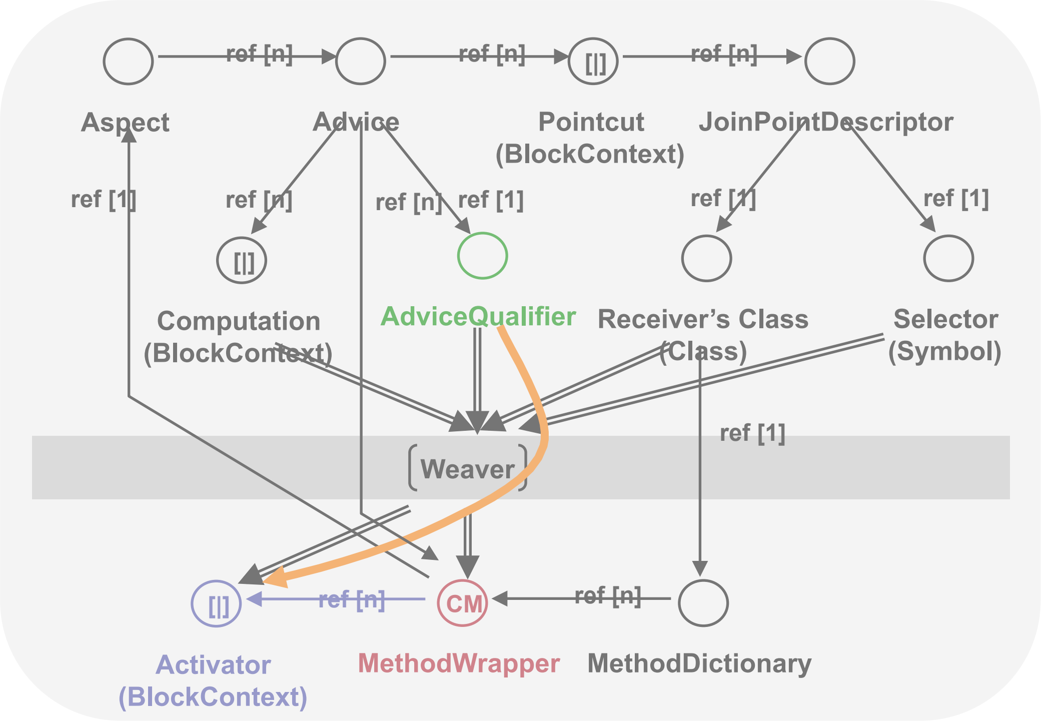
Context-oriented Programming

AspectS

Dynamic Aspect-oriented Programming

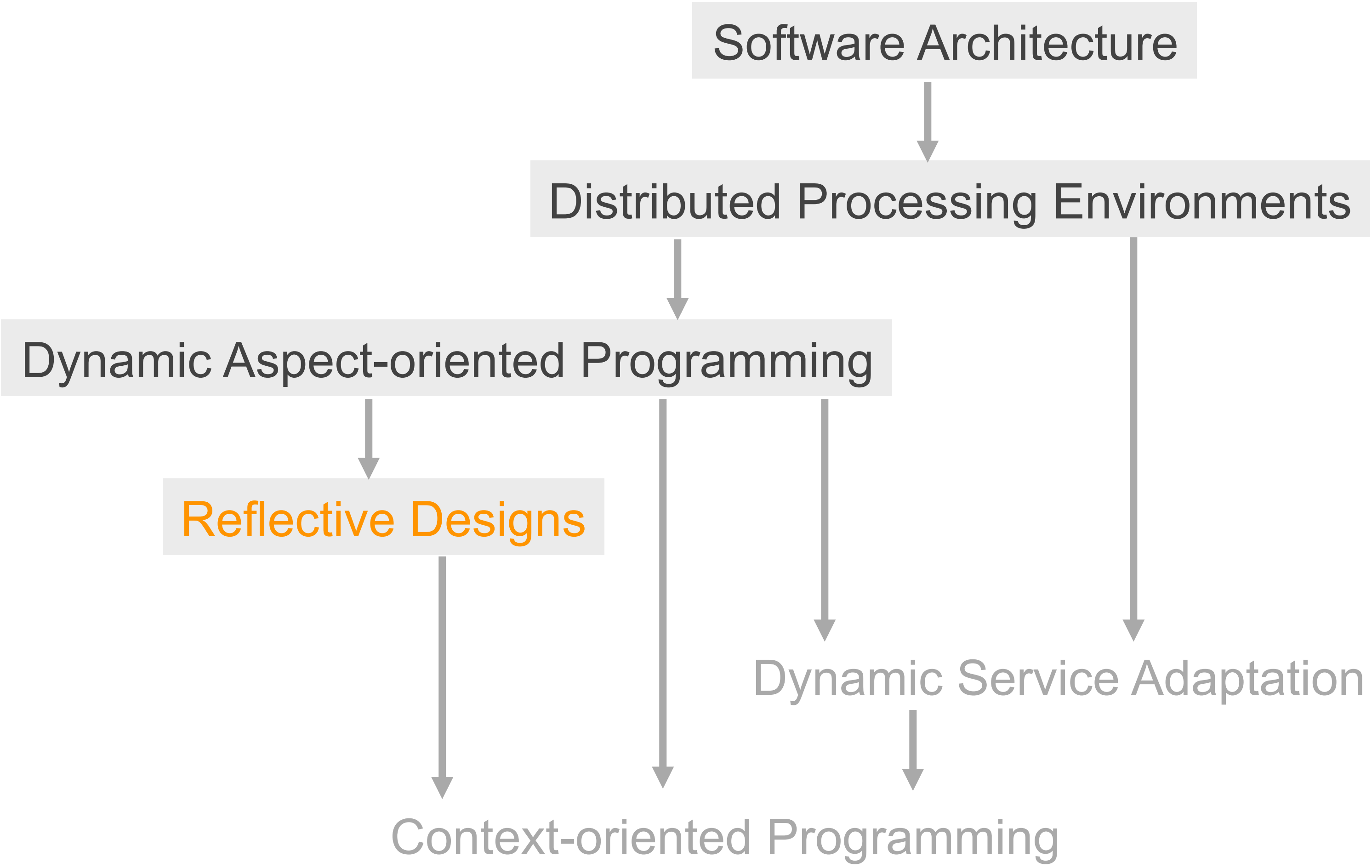






AspectS (Windward Solutions & DoCoMo Euro-Labs)

Outline



Software Architecture

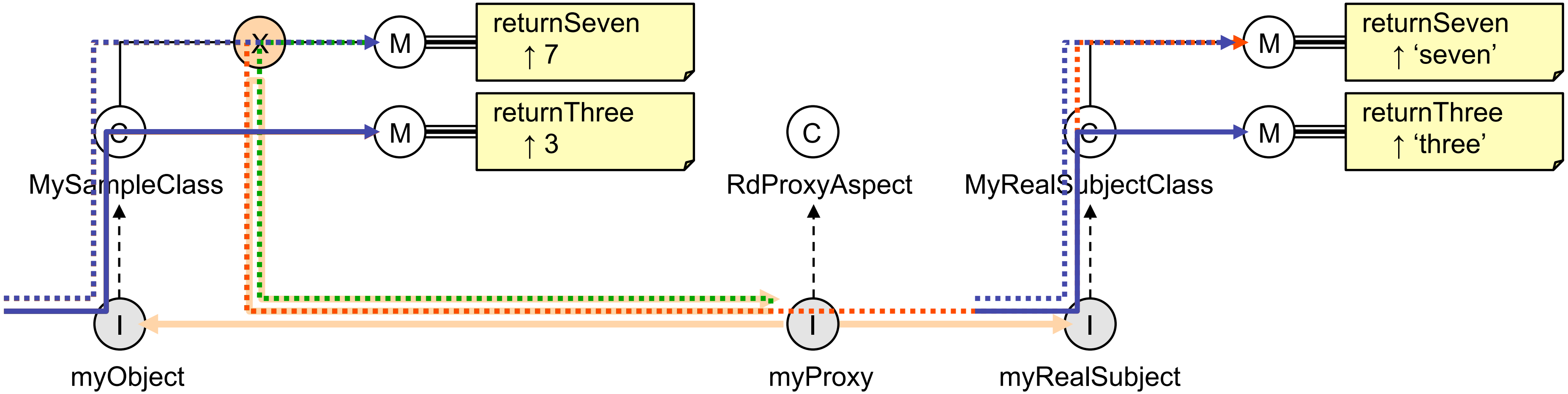
Distributed Processing Environments

Dynamic Aspect-oriented Programming

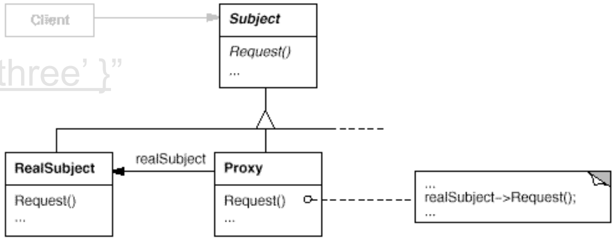
Reflective Designs

Dynamic Service Adaptation

Context-oriented Programming



```
| myObject myProxy myRealSubject |
myObject ← MySampleClass new.
{ myObject returnSeven. myObject returnThree }. “→ { 7. 3 }”
myRealSubject ← MyRealSubjectClass new.
{ myRealSubject returnSeven. myRealSubject returnThree }. “→ { ‘seven’. ‘three’ }”
myProxy ← RdProxyAspect new.
myProxy proxy: MySampleClass selectors: { #returnSeven }.
myProxy activate.
{ myObject returnSeven. myObject returnThree }. “→ { 7. 3 }”
myProxy addSubject: myObject realSubject: mySubject.
{ myObject returnSeven. myObject returnThree }. “→ { ‘seven’. 3 }”
myProxy deactivate.
{ myObject returnSeven. myObject returnThree }. “→ { 7. 3 }”
```

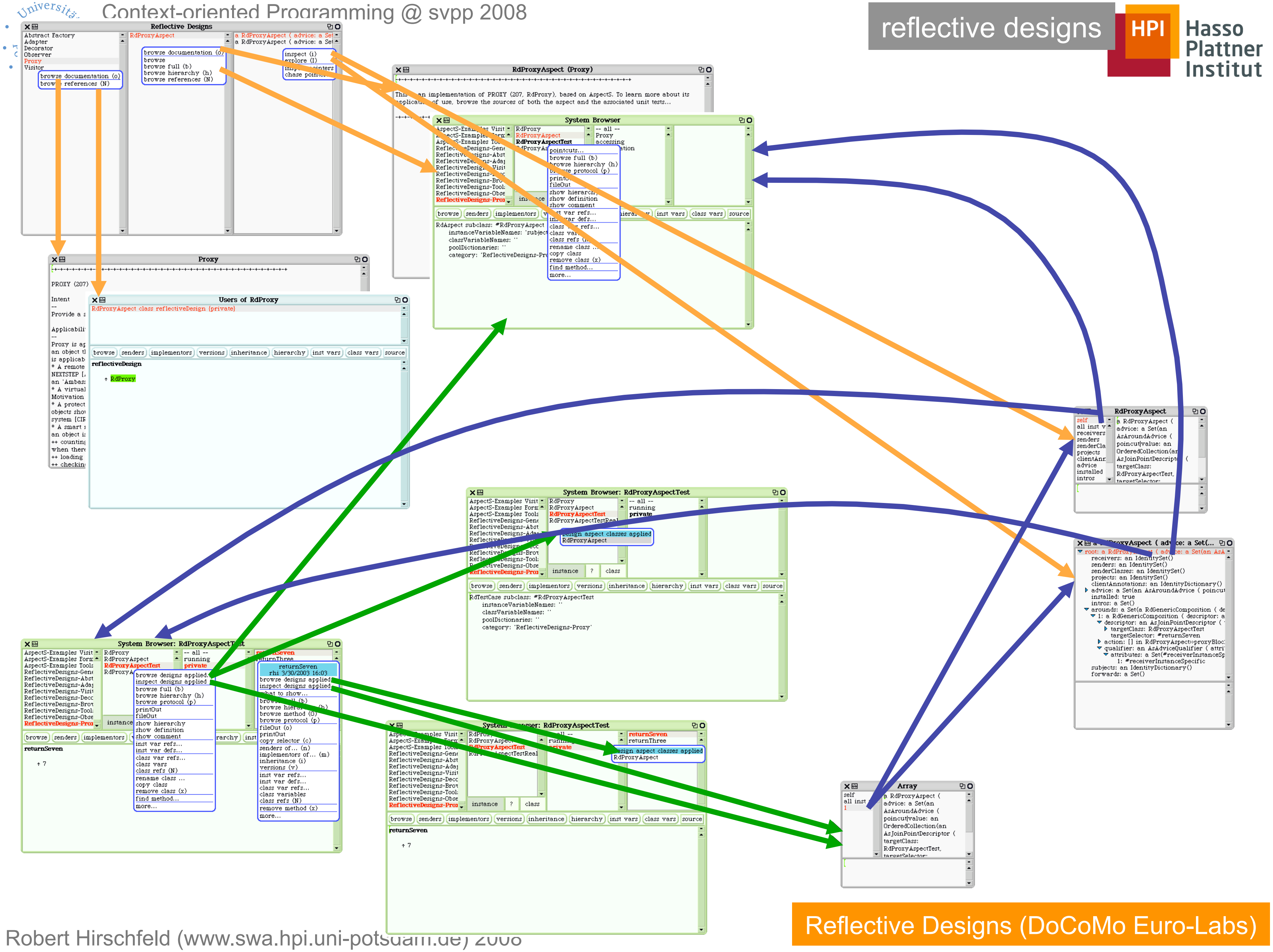


Proxy: Provide a surrogate or placeholder for another object to control access to it.

Reflective Designs (DoCoMo Euro-Labs)

reflective designs

Hasso Plattner Institut



This is an implementation of PROXY (207, RdProxy), based on AspectS. To learn more about its application of use, browse the sources of both the aspect and the associated unit tests...

PROXY (207)

Intent

Applicability

Motivation

RdProxyAspect class reflectiveDesign {private}

```
reflectiveDesign

	↑ RdProxy
```

```
RdAspect subclass: #RdProxyAspect
	instanceVariableNames: 'subject
	classVariableNames: ''
	poolDictionaries: ''
	category: 'ReflectiveDesigns-Pro
```

```
RdTestCase subclass: #RdProxyAspectTest
	instanceVariableNames: ''
	classVariableNames: ''
	poolDictionaries: ''
	category: 'ReflectiveDesigns-Proxy'
```

```
returnSeven

	↑ 7
```

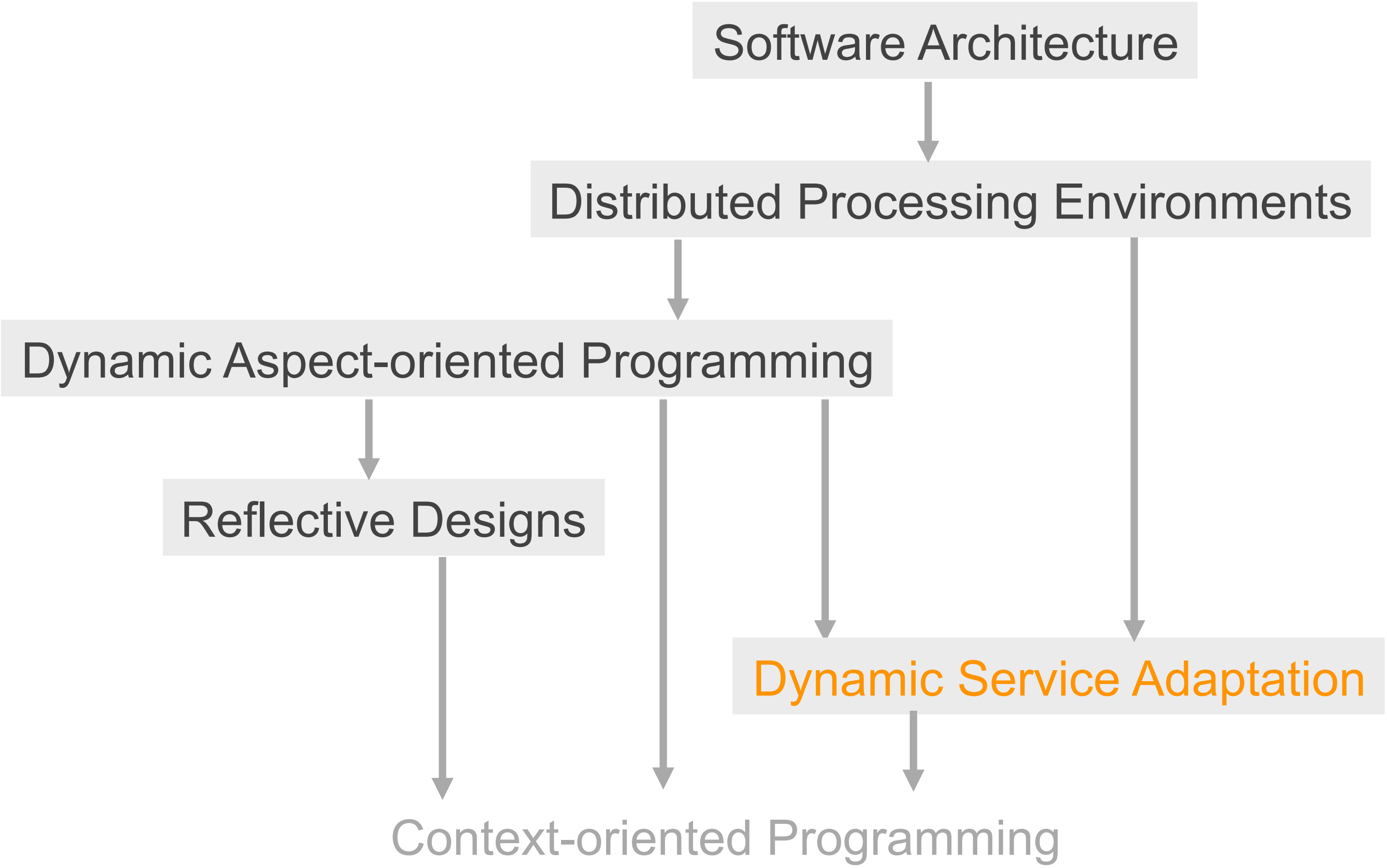
returnSeven
rhi 3/30/2003 16:03

```
a RdProxyAspect (
advice: a Set(an
AsAroundAdvice (
poincut|value: an
OrderedCollection(an
AsJoinPointDescriptor (
targetClass:
RdProxyAspectTest,
targetSelector:
```

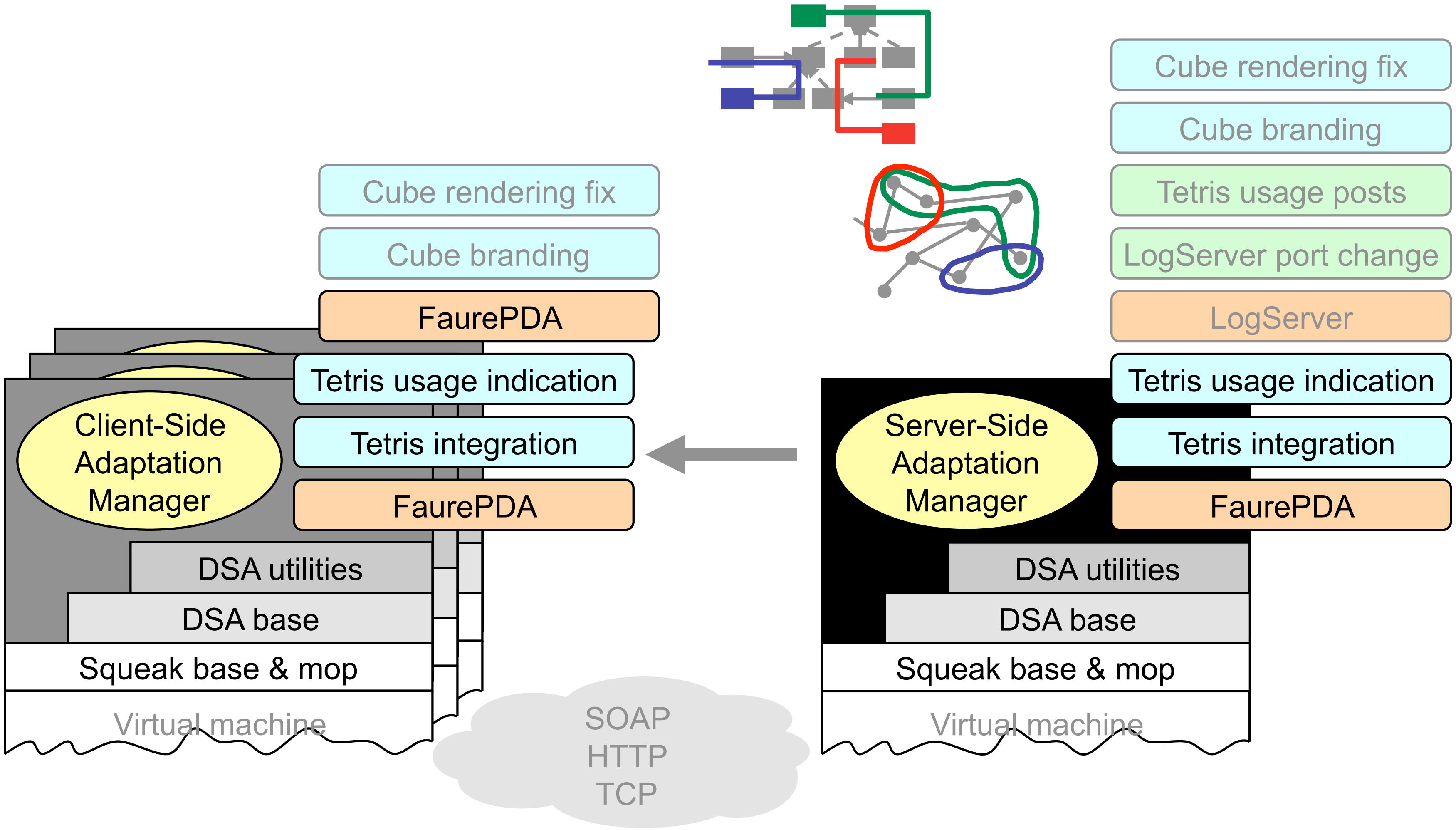
```
root: a RdProxyAspect ( advice: a Set(an AsA
    receivers: an IdentitySet()
    senders: an IdentitySet()
    senderClasses: an IdentitySet()
    projects: an IdentitySet()
    clientAnnotations: an IdentityDictionary()
  ▸ advice: a Set(an AsAroundAdvice ( poincut
    installed: true
    intros: a Set()
  ▾ arounds: a Set(a RdGenericComposition ( de
    ▾ 1: a RdGenericComposition ( descriptor: a
      ▾ descriptor: an AsJoinPointDescriptor (
        ▸ targetClass: RdProxyAspectTest
          targetSelector: #returnSeven
      ▸ action: [] in RdProxyAspect>>proxyBloc
      ▾ qualifier: an AsAdviceQualifier ( attri
        ▾ attributes: a Set(#receiverInstanceSp
            1: #receiverInstanceSpecific
    subjects: an IdentityDictionary()
    forwards: a Set()
```

Reflective Designs (DoCoMo Euro-Labs)

Outline

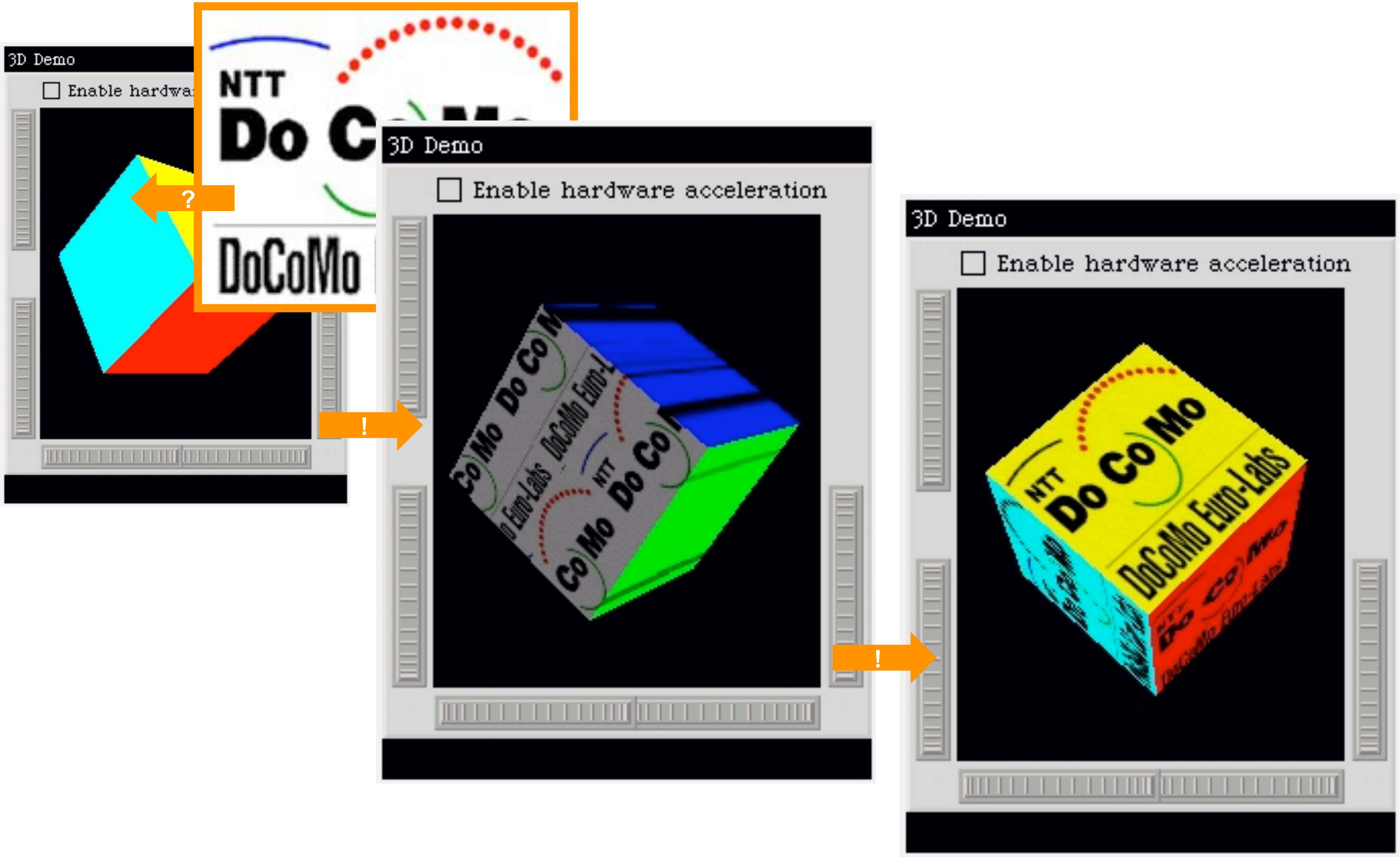

Software Architecture
Distributed Processing Environments
Dynamic Aspect-oriented Programming
Reflective Designs
Dynamic Service Adaptation
Context-oriented Programming

Deployment Scenarios



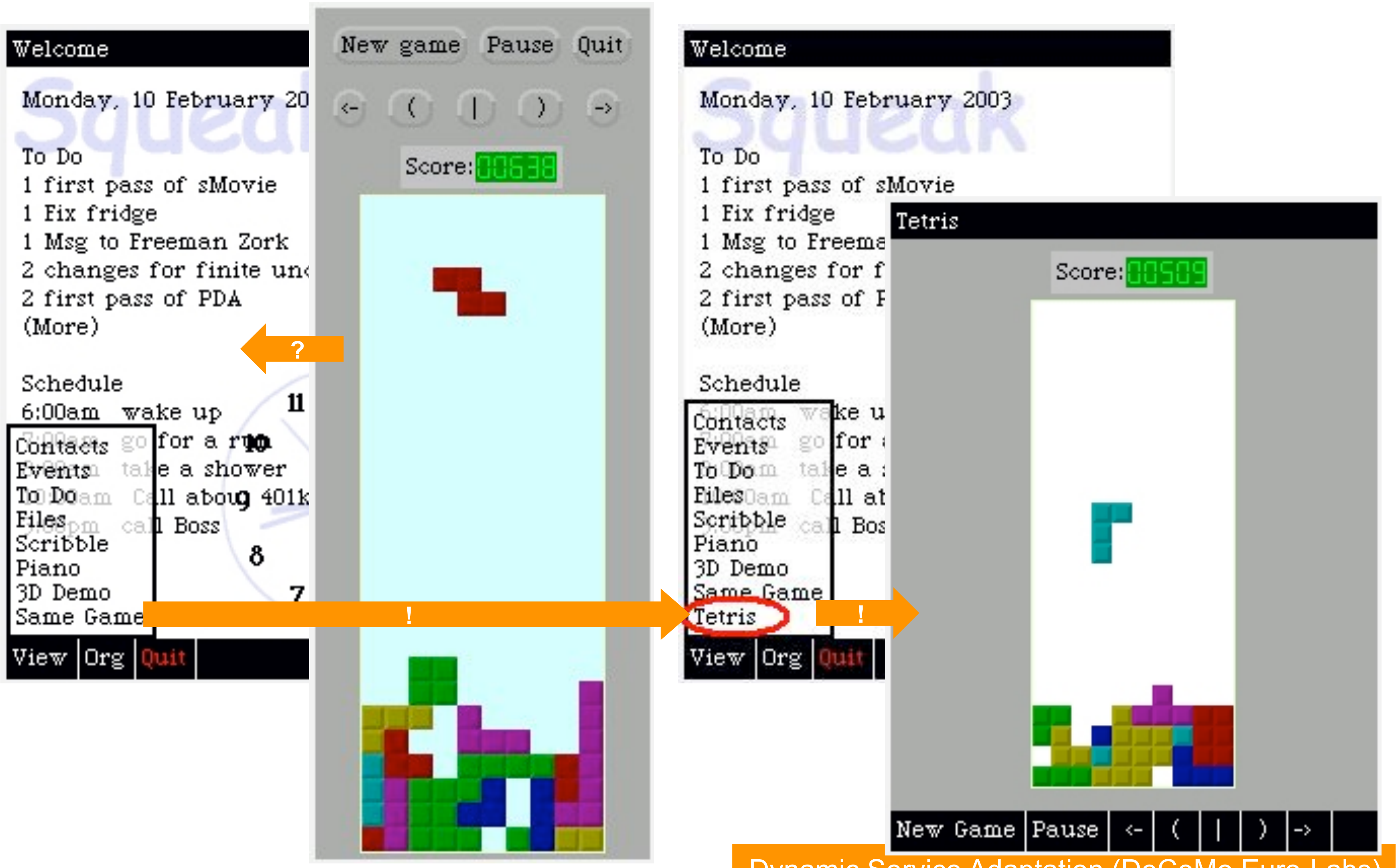
Dynamic Service Adaptation (DoCoMo Euro-Labs)

UI Branding and Bugfixing…


3D Demo
□ Enable hardwa
NTT
Do CoMo
DoCoMo
?
!
3D Demo
□ Enable hardware acceleration
!
3D Demo
□ Enable hardware acceleration

Dynamic Service Adaptation (DoCoMo Euro-Labs)

Service Integration…



Welcome

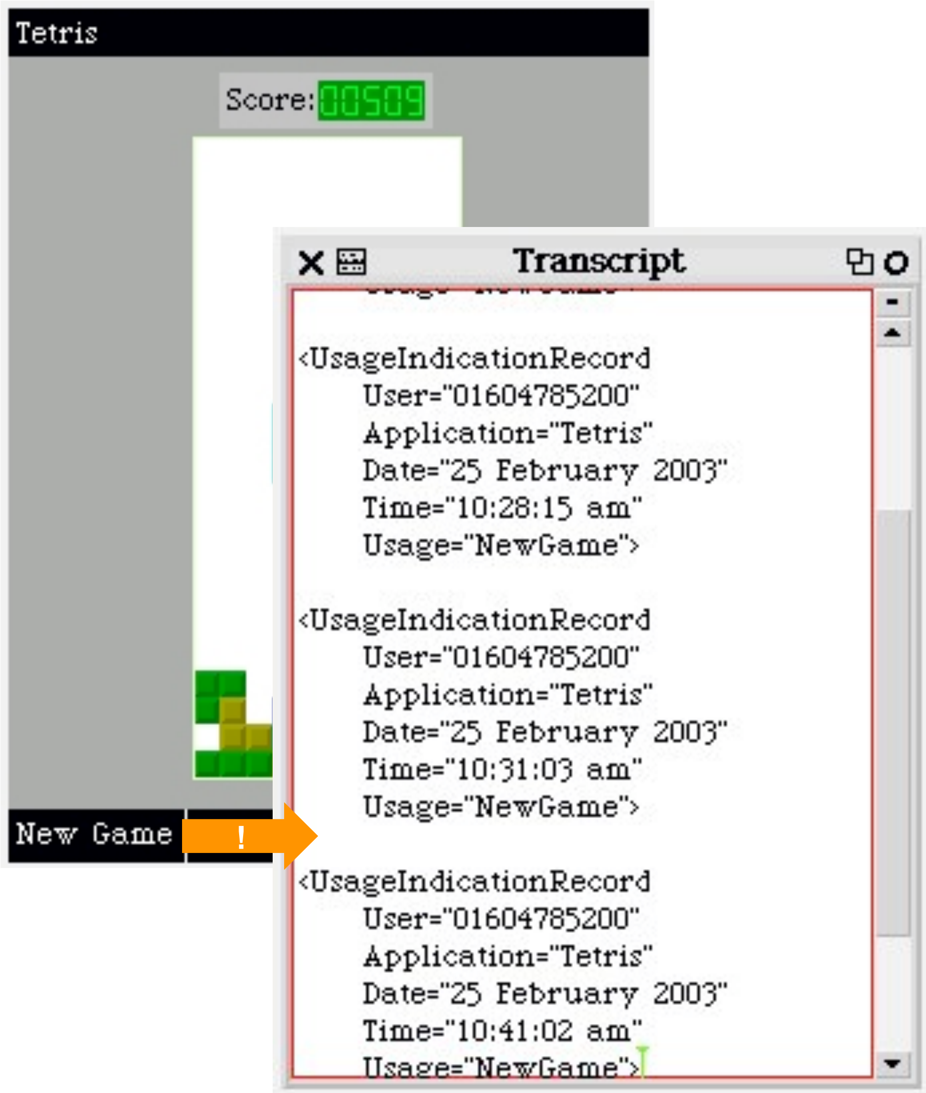
Monday, 10 February 2003

To Do
1 first pass of sMovie
1 Fix fridge
1 Msg to Freeman Zork
2 changes for finite un
2 first pass of PDA
(More)

Schedule
6:00am wake up
7:00am go for a run
7:30am take a shower
9:00am Call about 401k
3:00pm call Boss

Dynamic Service Adaptation (DoCoMo Euro-Labs)

…Usage Indication



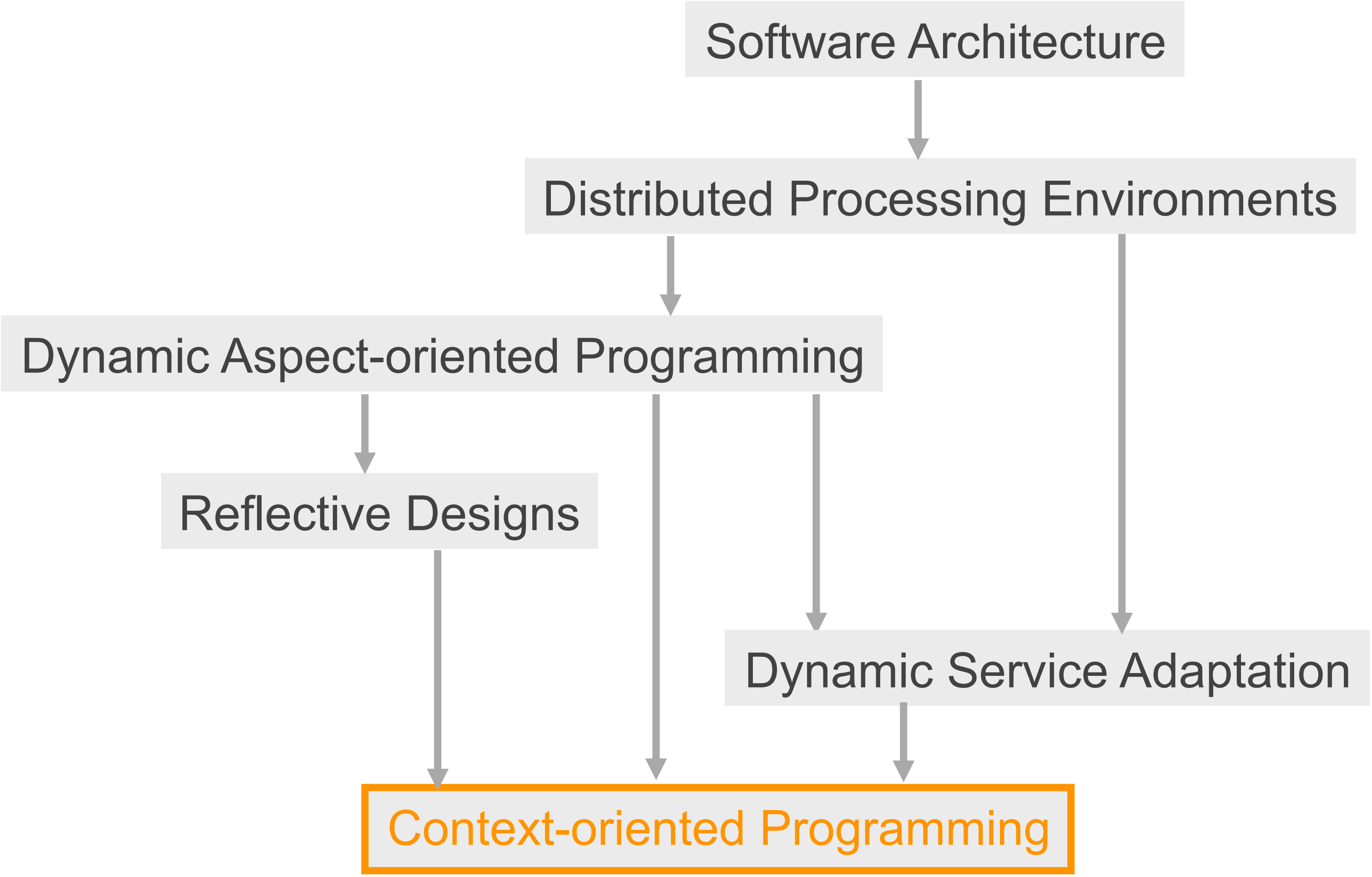
```
<UsageIndicationRecord
     User="01604785200"
     Application="Tetris"
     Date="25 February 2003"
     Time="10:28:15 am"
     Usage="NewGame">

<UsageIndicationRecord
     User="01604785200"
     Application="Tetris"
     Date="25 February 2003"
     Time="10:31:03 am"
     Usage="NewGame">

<UsageIndicationRecord
     User="01604785200"
     Application="Tetris"
     Date="25 February 2003"
     Time="10:41:02 am"
     Usage="NewGame">
```

Dynamic Service Adaptation (DoCoMo Euro-Labs)

Outline



Software Architecture

Distributed Processing Environments

Dynamic Aspect-oriented Programming

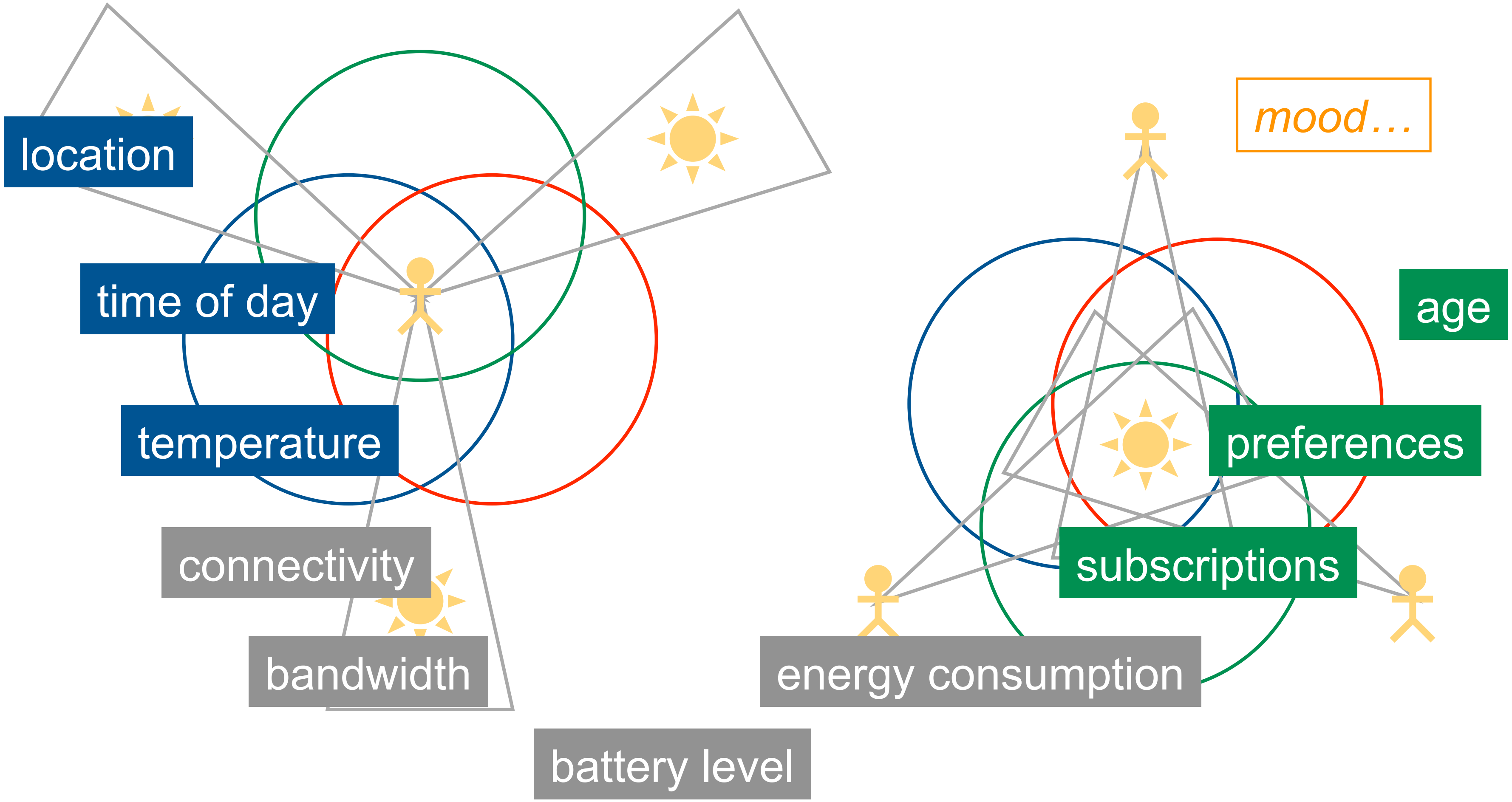
Reflective Designs

Dynamic Service Adaptation

Context-oriented Programming

Context

context = everything computationally accessible



location

time of day

temperature

connectivity

bandwidth

battery level

energy consumption

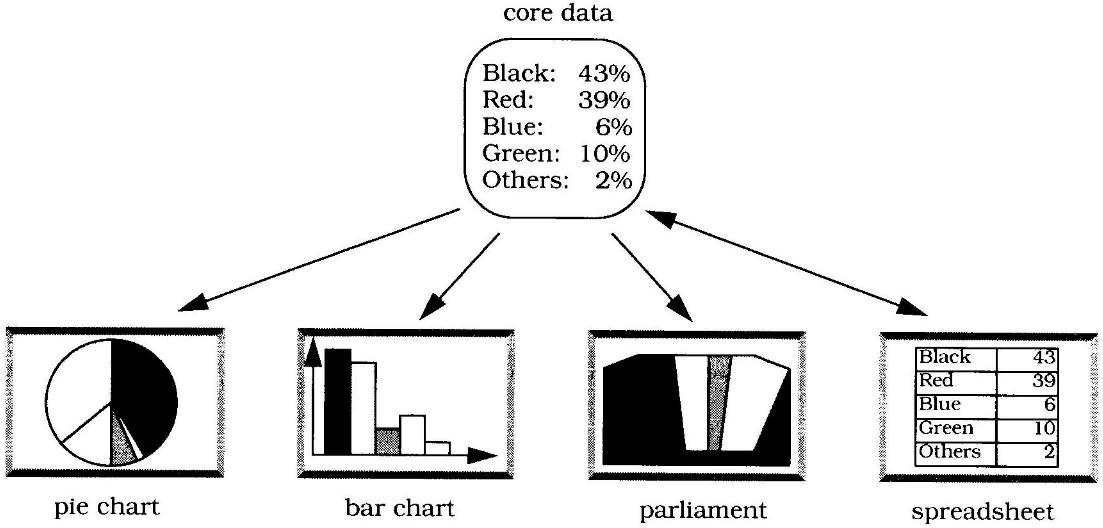
subscriptions

preferences

age

mood…

MVC

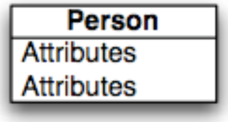


Black:	43%
Red:	39%
Blue:	6%
Green:	10%
Others:	2%

Black	43
Red	39
Blue	6
Green	10
Others	2

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

Increased Complexity

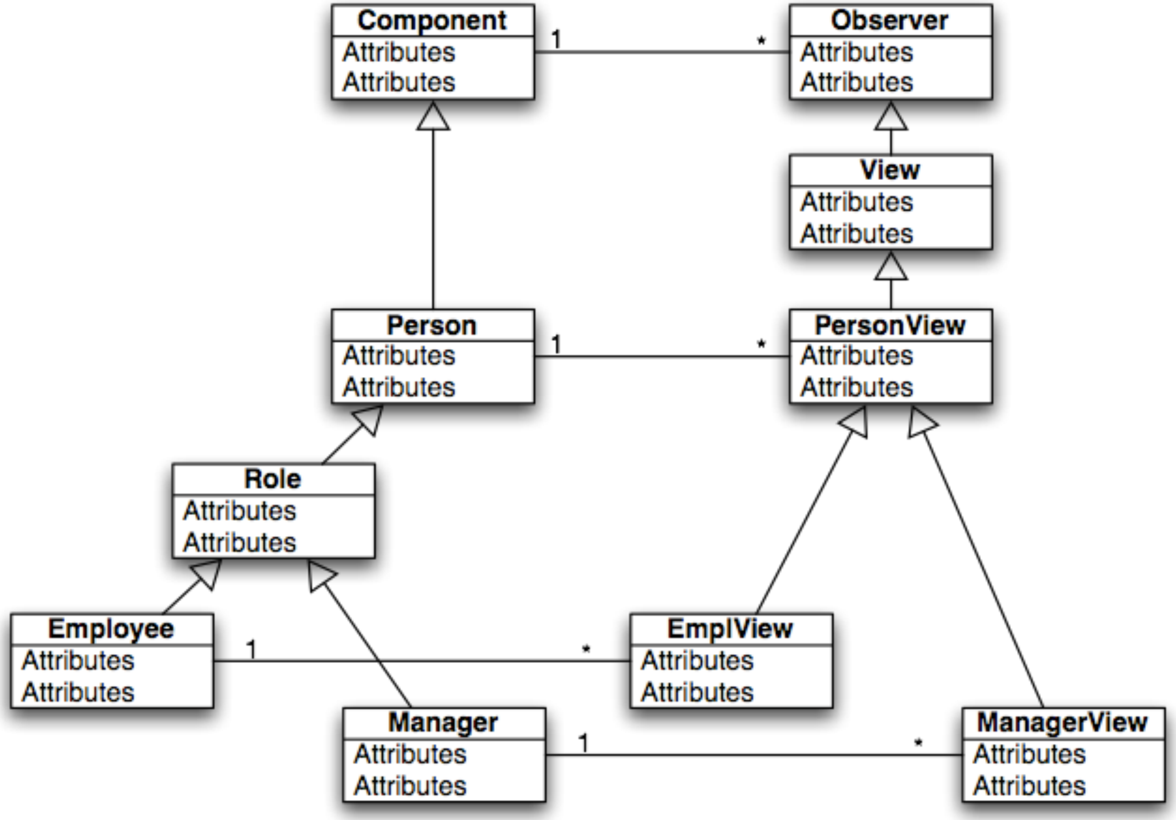


Person

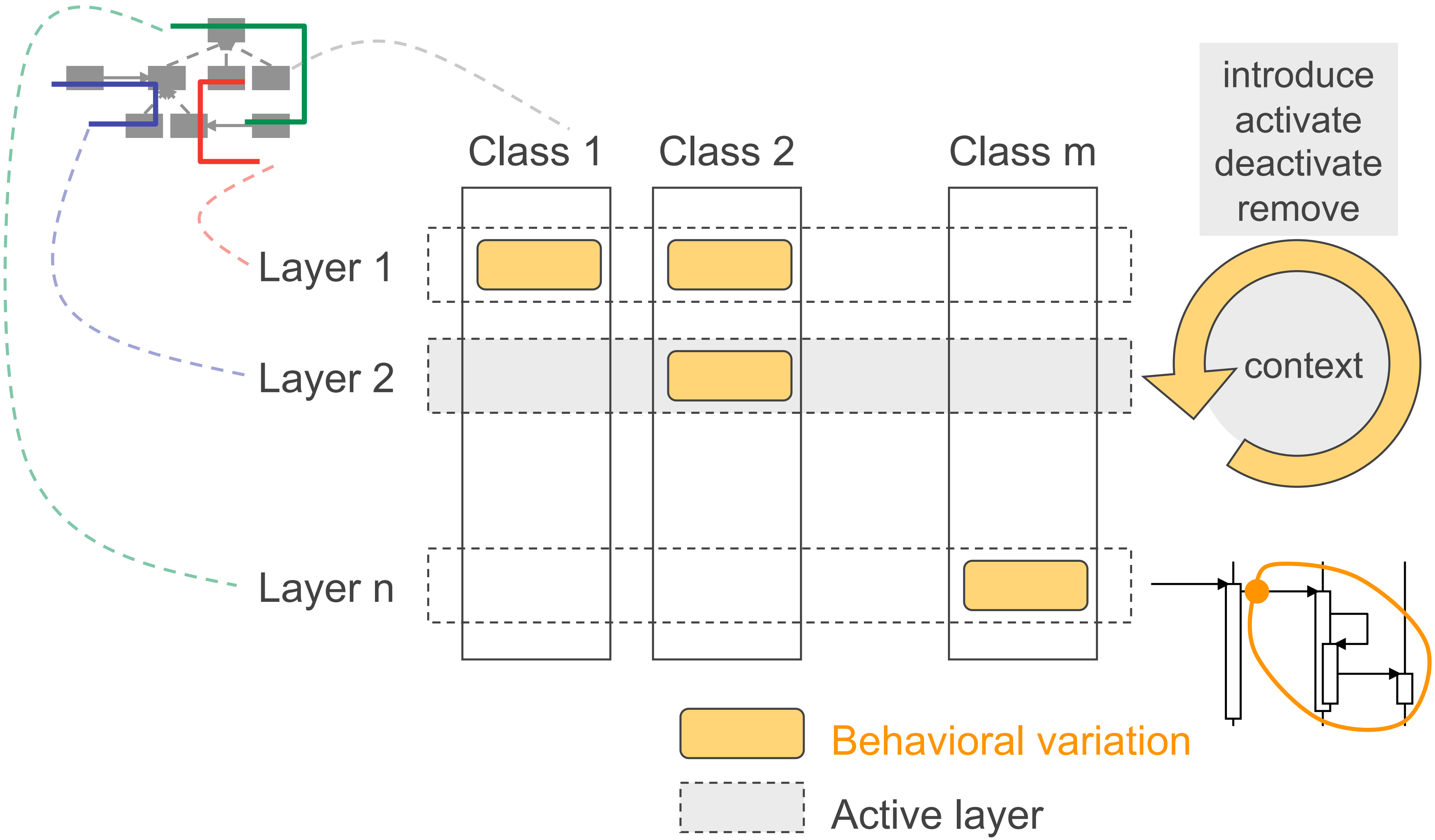
Attributes

Attributes

Increased Complexity



Partial Layer and Class Definitions



Class 1

Class 2

Class m

Layer 1

Layer 2

Layer n

introduce
activate
deactivate
remove

context

Behavioral variation

Active layer

COP Basics

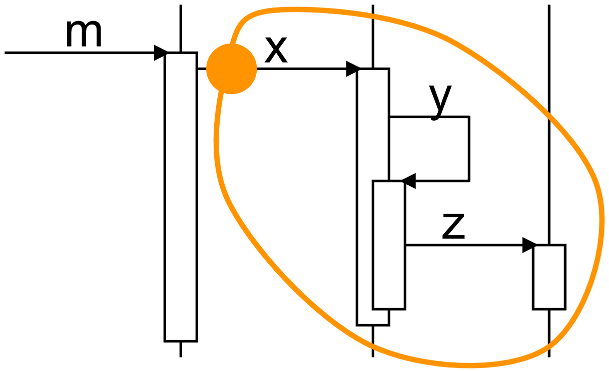
- Behavioral variations
  - Partial class and method definitions
- Layers
  - Groups of related context-dependent behavioral variations
- Activation
  - Activation and deactivation of layers at runtime
- Context
  - Anything computationally accessible
- Scoping
  - Well-defined explicitly-controlled scopes

Dynamically-scoped Layer Activation

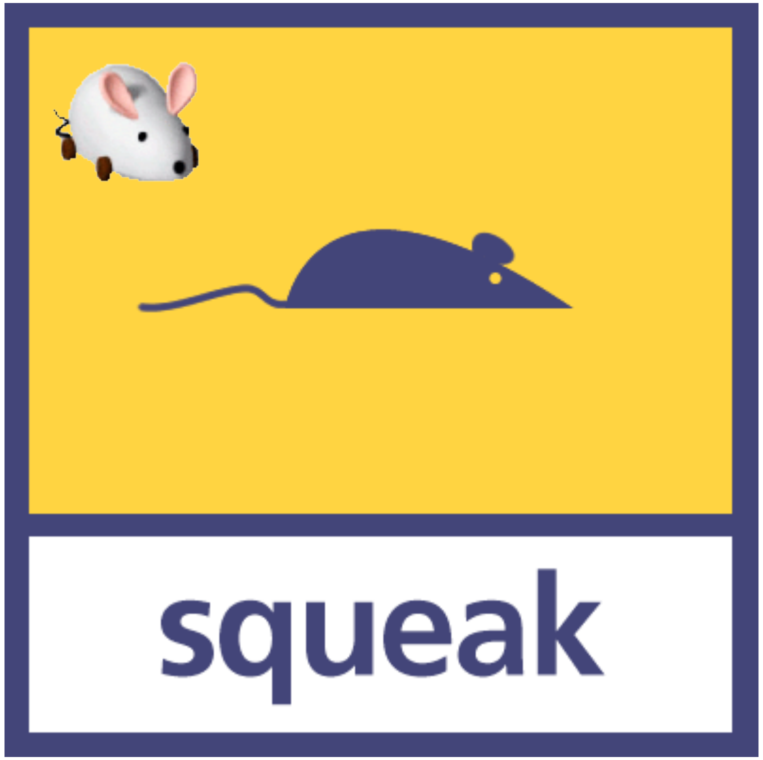
- Constructs

(with-active-layers (…) …)	ContextL
(with-inactive-layers (…) …)	
[…] useAsLayersFor: […]	ContextS
with (…) {…}	ContextJ

- Activate (deactivate) layers for the current thread
  - Does not interfere with other layer activations/deactivations in other threads
- Layers are activated/deactivated only for the dynamic extent of the associated code block
- Activation order determines method precedence



Demo



AOP vs. FOP vs. COP

	AOP	FOP	COP
Inverse dependencies	●		
1:n relationships	●		
Layers		●	●
Dynamic activation			●
Scoping	●		●

first-class layers
explicit meta-objects
scoped adaptation
improved comprehension

COP Implementations

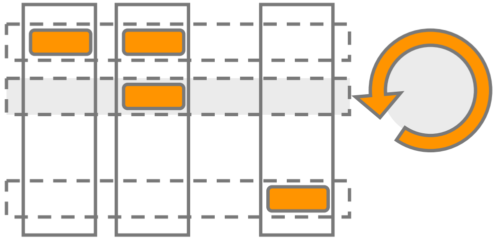
- ContextL (VUB/PROG)
- ContextS (HPI/SWA)
- ContextJ* (VUB/PROG)
- ContextR (HPI/SWA)
- ContextPy (HPI/SWA)
- ContextJ (HPI/SWA)
- ContextG (HPI/SWA)
- PyContext (HPI/DCL)
- Context# (HPI/DCL)
- …

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

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