



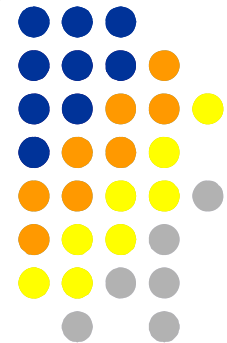
LANCASTER
UNIVERSITY



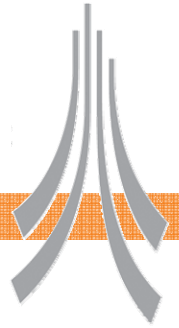
On the Quantitative Analysis of Architectural Stability in Aspectual Decompositions

**Ambra Molesini, *Alessandro Garcia*,
Christina Chavez, Thais Batista**

ambra.molesini@unibo.it, garciaa@comp.lancs.ac.uk
flach@ufba.br, thais@ufrnet.br



Software Stability vs. Aspect “Quarks”



- The presence of the so-called *crosscutting concerns* is detrimental to software stability
 - ripple effects [Greenwood et al, ECOOP.07]
 - error-prone modifications [Eaddy et al, IEEE TSE 2008]
- “Quarks” of aspect-oriented software architectures
 - architectural aspects
 - join points and join point models
 - pointcuts (bindings + quantification mechanisms)
 - etc...



Greenwood, P., Bartolomei, T., Figueiredo, E., Dosea, M., Garcia, A. et al.. ***On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study***. Proc. 21st ECOOP, July 2007, Berlin.



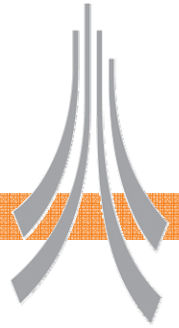
Eaddy, M. et al. ***Do Crosscutting Concerns Cause Defects?*** IEEE Transactions on Software Engineering, to appear, 2008.

Conventional Wisdom

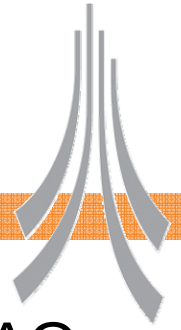


- Architecturally-relevant crosscutting concerns (CCC) should be always aspectised – valid for:
 - different relationships between base components and aspects
 - all architectural styles and respective joinpoint models
 - ... thereby achieving architectures with superior stability
- **The Saga:** aggressive incarnations of aspect-oriented extensions to ADLs: DAOP-ADL, AspectualACME, AO ADL, Fractal, Prisma, FuseJ, etc...

Stability of AO Architectures is Unknown



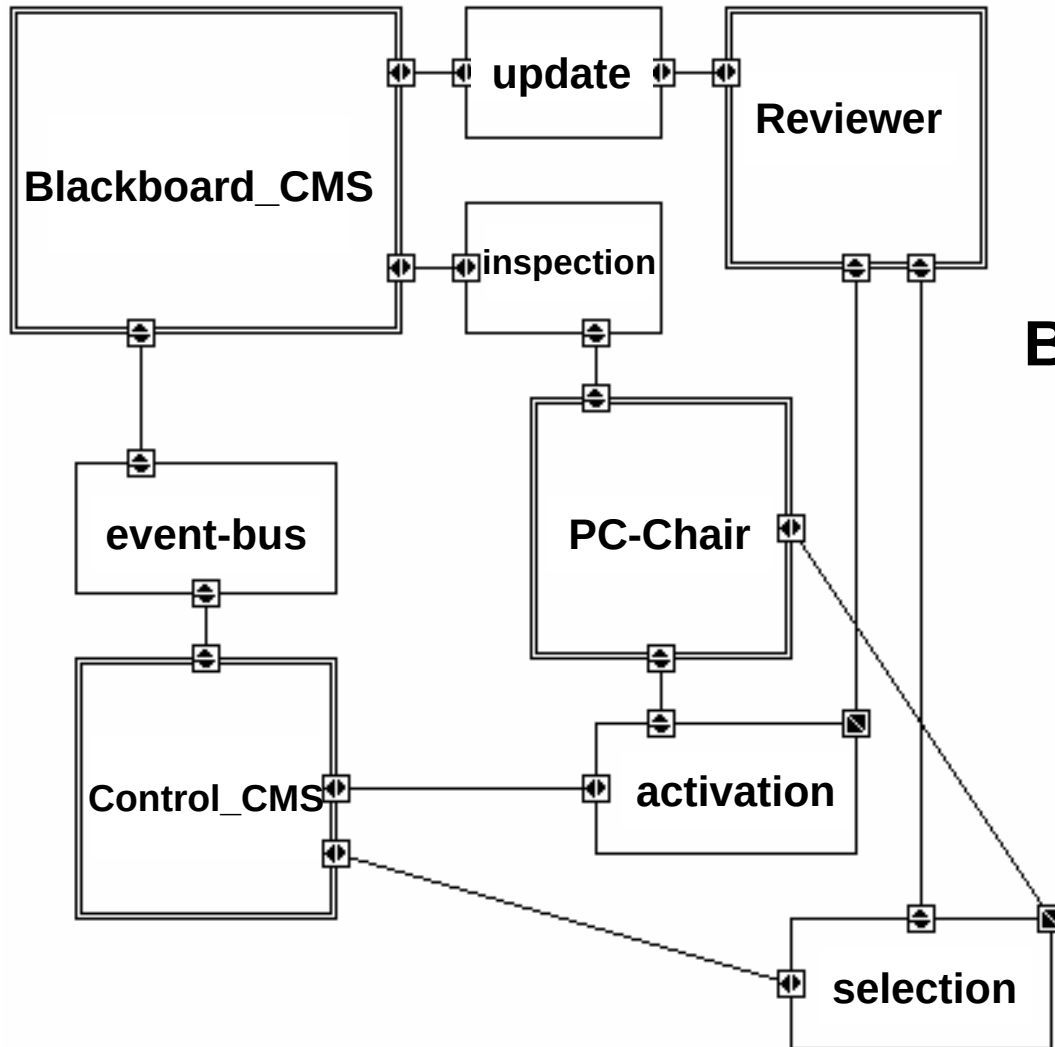
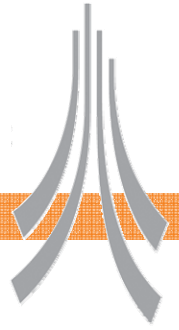
- Use of fine-grained composition mechanisms, such as, multiple inheritance were sources of major ripple effects
- Do aspect “quarks” entail stable software architectures?
 - to what extent ripple effects increase or decrease?
 - how different types of CCCs, styles, and joinpoint models influence AO architecture stability?
 - correlation with implementation-level stability studies?



Study Goal and Procedures

- **Goal:** exploratory analysis of the influence exerted by AO mechanisms in the architectural stability of CCCs
- Target Application: a Conference Management System (CMS) - supports management of scientific conferences
 - Original version: three-tiered architecture that embedded a blackboard component to manage conference information
 - CMS progressively evolved in accordance with different hybrid styles
 - reflective blackboard
 - reactive coordination
 - stigmergic coordination

CMS – Blackboard Architecture

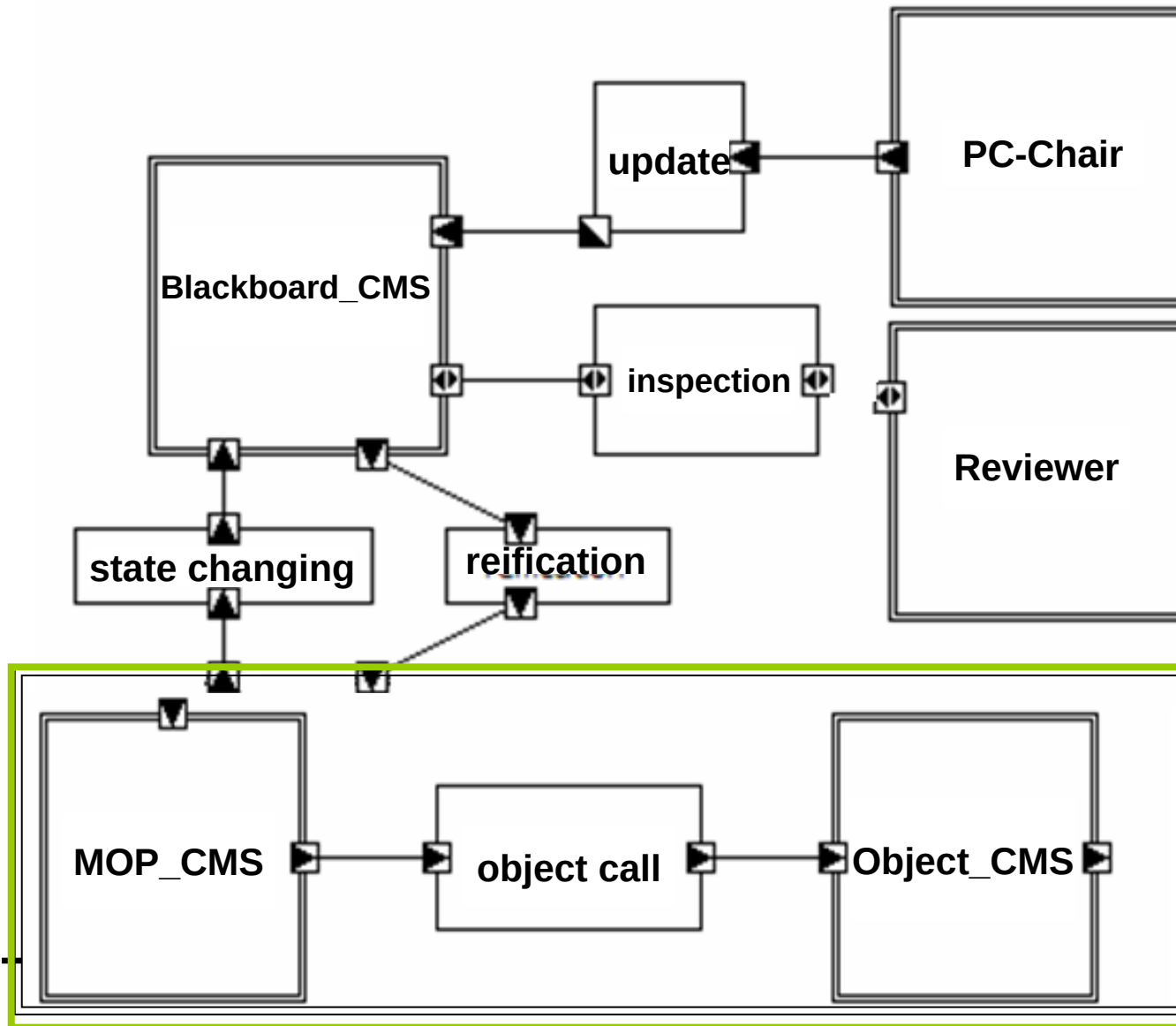


**“Slice” of the
Blackboard Architecture
Version**





CMS - Reflective Blackboard Architecture



“Slice” of the
Reflective
Blackboard
Version



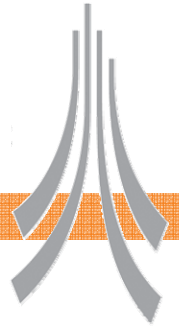
Study Procedures - II



- Identification of the recurring crosscutting concerns in the original blackboard architecture
 - error handling
 - security
 - coordination
- Generation of an aspect-oriented architecture counterpart for each non-AO decomposition
 - E.g.: Non-AO vs. AO blackboard architecture
Non-AO vs. AO reactive coordination architecture
etc...
 - activities: specification of the pointcuts, aspectual connectors, etc...



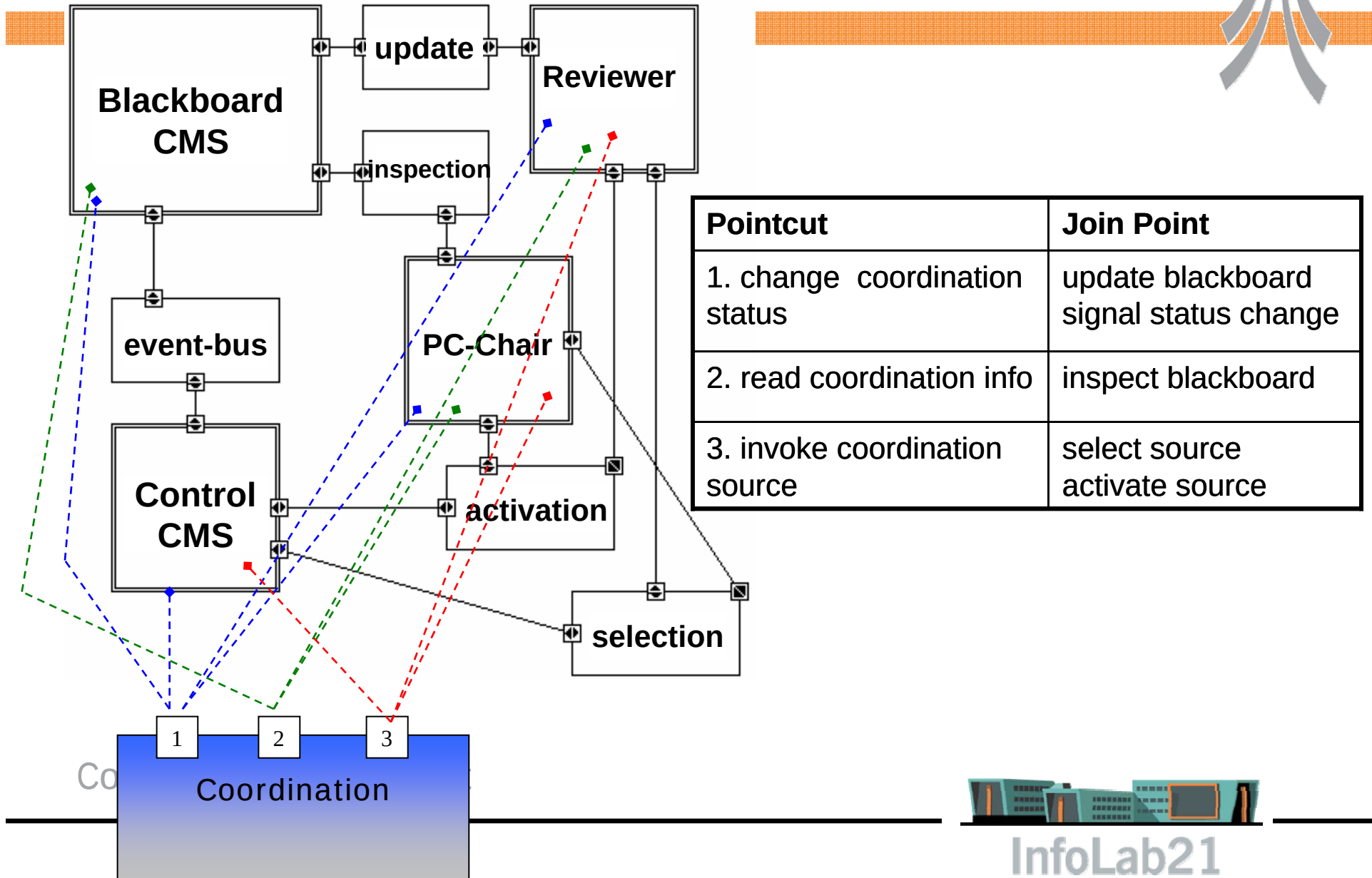
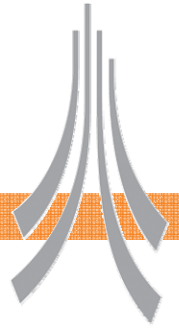
Identification of Join Point Models



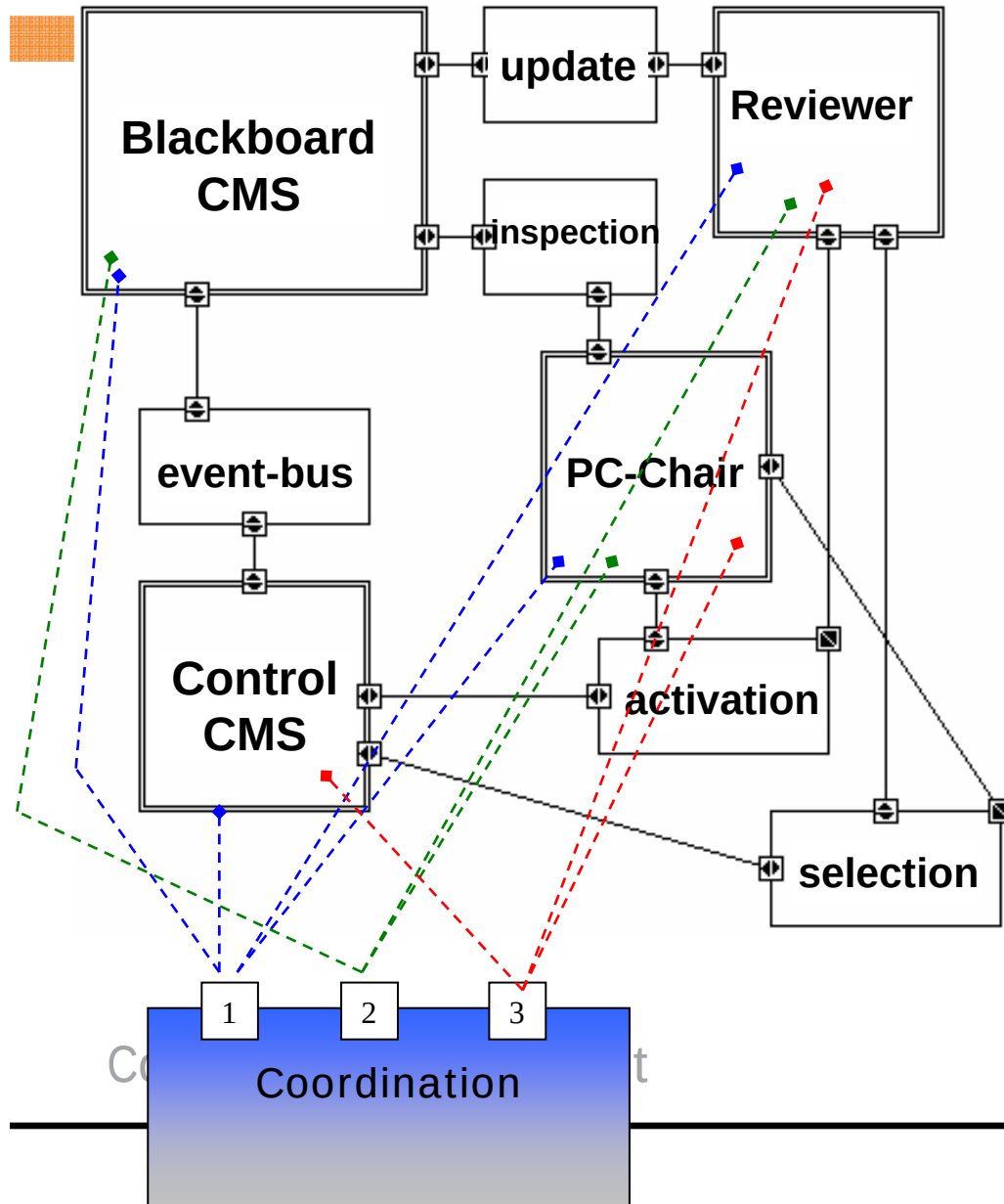
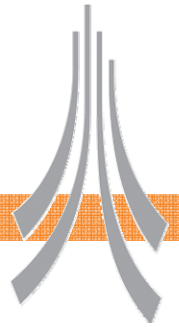
- Syntax-based joinpoints: names of the components, interfaces, etc...
- Style-based joinpoints
 - e.g. blackboard joinpoint model

Join point	Where
inspect blackboard update blackboard	KS, Blackboard
signal status change	Blackboard, Control
query source, select source, activate source	Control, KS

Pointcuts in Aspect-Oriented Blackboard



Pointcuts in Aspect-Oriented Blackboard



```

System CMS : BlackboardFam =
    new BlackboardFam extended with {
    Component PC Chair: KS;
    Component Rev1, Rev2, Rev3: KS;
    ...
    \* Coordination Crosscutting Concern *\
    Component Coordinator = {
        Port changeCoordStatus: ... ;
        Port readCoordInfo: ... ;
        Port invokeCoordSource: ... ; };
    Connector coord = f
        crosscuttingRole source;
        baseRole sink;
        glue source after sink; };
    
```

```

\* Pointcut description - AO bindings *\
Attachments {
    Coordinator.readCoordInfo to coord.source;
    coord.sink to *.inspectBB;
    ... }
}
    
```

Stability Metrics



- Architecture stability assessment across the releases of all the non-AO and AO versions of the CMS system
- The metrics suite was composed by:
 - quantifying change propagation
 - components added/changed
 - connectors added/changed
 - quantifying binding stability
 - bindings added or changed in the non-AO architectures
 - pointcuts added or changed in the AO architectures



C. Sant'Anna et al. "**On the quantitative assessment of modular multi-agent architectures**". In D. Weyns et al., editors, Multiagent Systems and Software Architecture, pages 111– 135. Katholieke Univ. Leuven, Belgium, 2006.

Change Scenarios



- In order to perform the stability measurements, a set of eight change scenarios have been applied
 - S1: introduction of sub-committees and vice-chairs
 - S2: introduction of the external reviewers
 - S3: refinement of security constraints
 - S4: introduction of new coordination protocols
 - S5: improvement in the error handling strategies
 - S6: introduction of organizational rules and changes in the existing rules
 - S7: evolution in the agents' capabilities
 - S8: introduction of a new paper assignment strategy

Change Scenarios



- In order to perform the stability measurements, a set of eight change scenarios have been applied
 - S1: introduction of sub-committees and vice-chairs **non-CCCs**
 - S2: introduction of the external reviewers
 - S3: refinement of security constraints
 - S4: introduction of new coordination protocols **CCCs**
 - S5: improvement in the error handling strategies
 - S6: introduction of organizational rules and changes in the existing rules
 - S7: evolution in the agents' capabilities **non-CCCs**
 - S8: introduction of a new paper assignment strategy

Change Scenarios



- In order to perform the stability measurements, a set of eight change scenarios have been applied
 - S1: introduction of sub-committees and vice-chairs
 - S2: introduction of the external reviewers
 - S3: refinement of security constraints
 - S4: introduction of new coordination protocols
 - S5: improvement in the error handling strategies
 - S6: introduction of organizational rules and changes in the existing rules
 - S7: evolution in the agents' capabilities
 - S8: introduction of a new paper assignment strategy

**fine-grained vs.
coarse-grained
changes**

Analysis: Stability of Components/Connectors

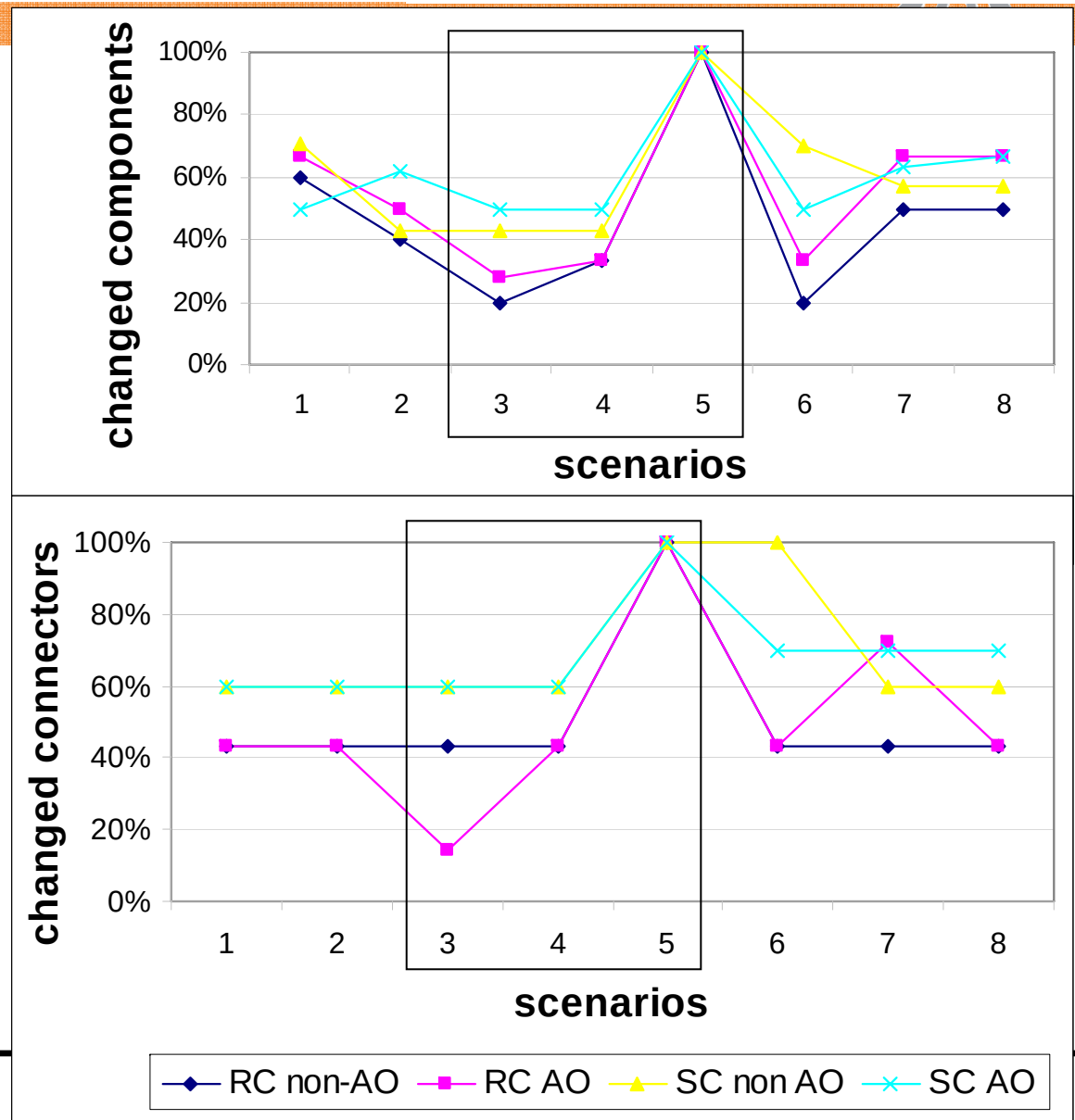
Examples:

Reactive Coordination
Stigmergic Coordination

AO architectures were
sensibly more stable:

*focus of a change was a
crosscutting concern*

Computing Department



Analysis: Stability of Components/Connectors

Examples:

Reactive Coordination
Stigmergic Coordination

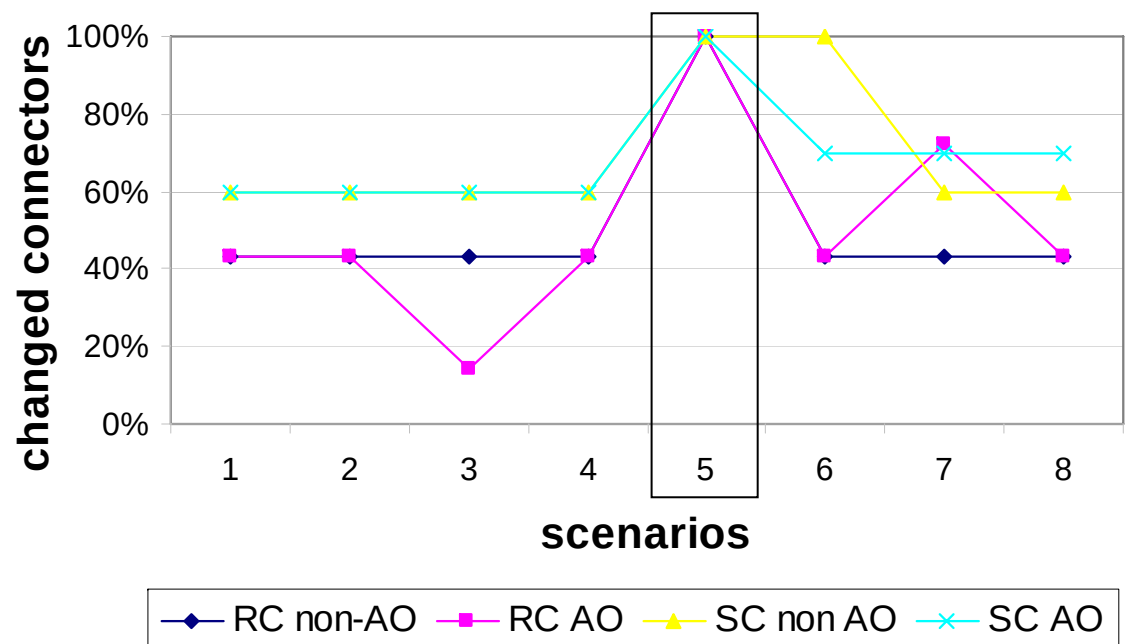
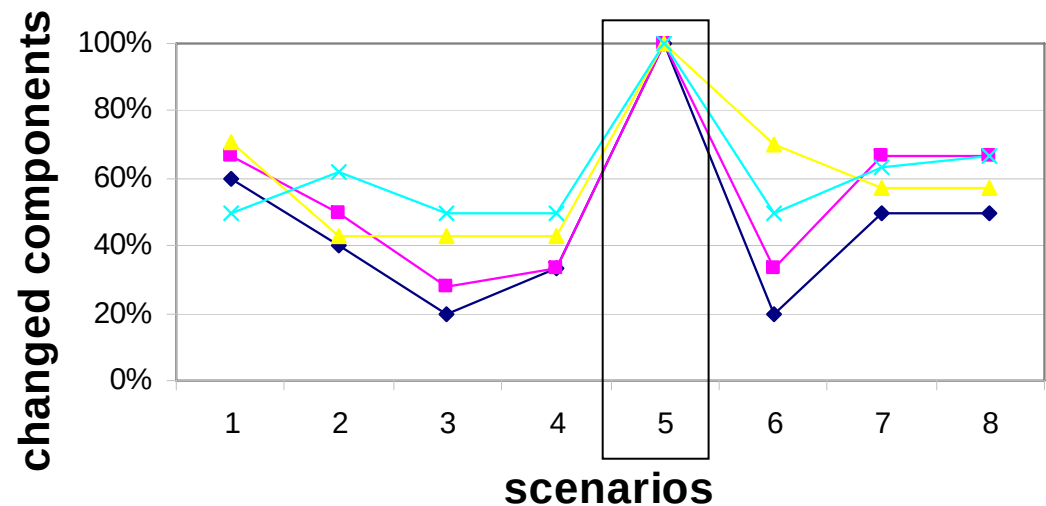
AO architectures were
sensibly more stable:

*focus of a change was a
crosscutting concern*

Exception:

exception handling

Computing Department



Analysis: Stability of Components/Connectors

Examples:

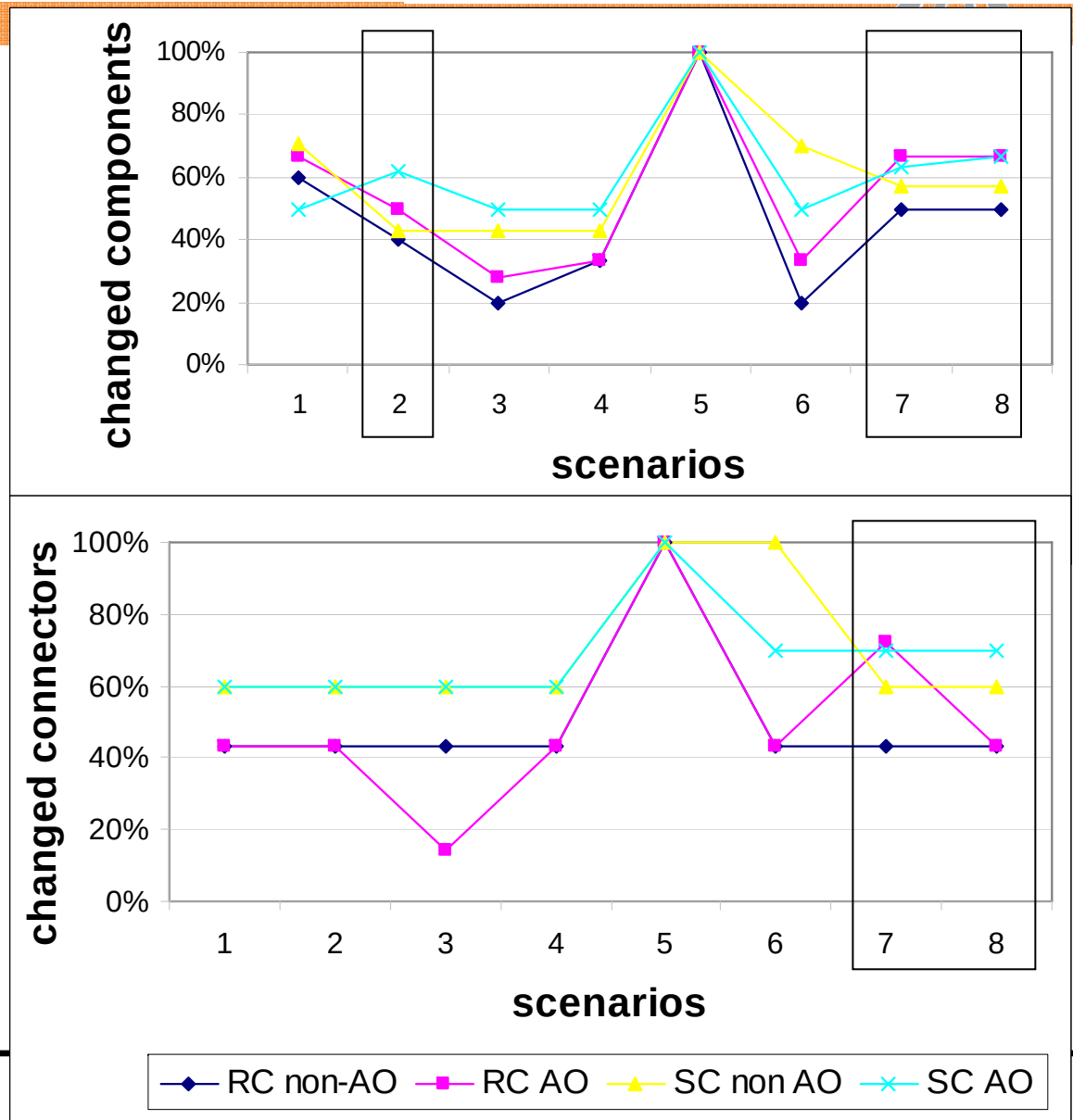
Reactive Coordination
Stigmergic Coordination

Non-AO architectures
tended to be more stable:

*focus of a change was a
non-crosscutting concern*

**Key factor: “density” of
crosscutting in the same
join point**

(independent of styles, types
of changes, CCCs)

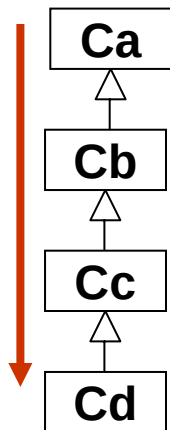


Analysis: Stability of Components/Connectors

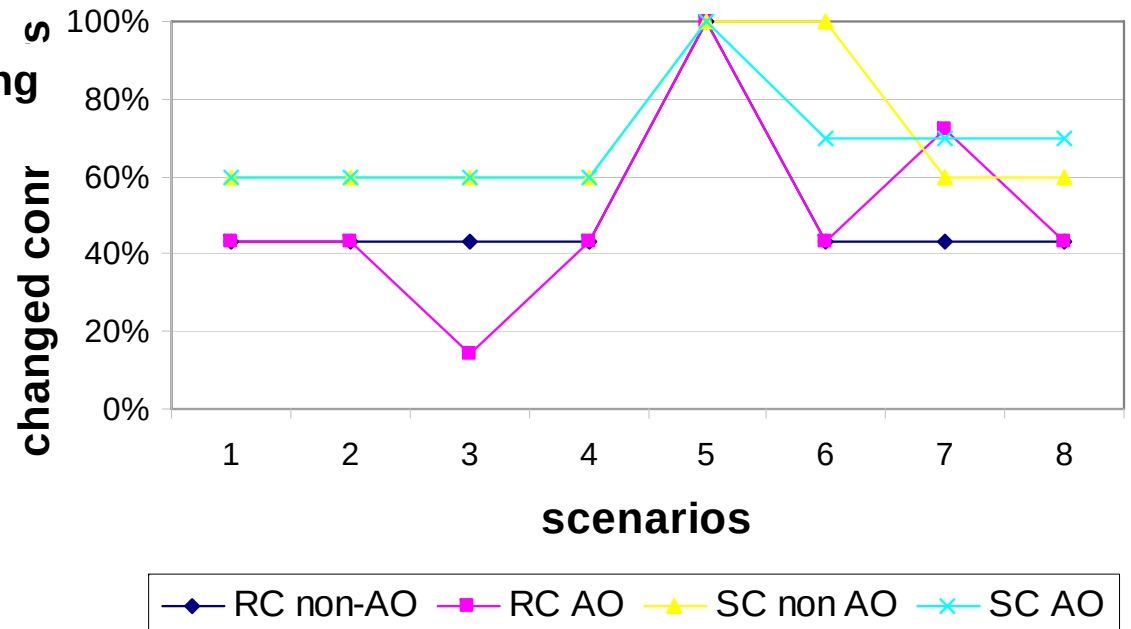
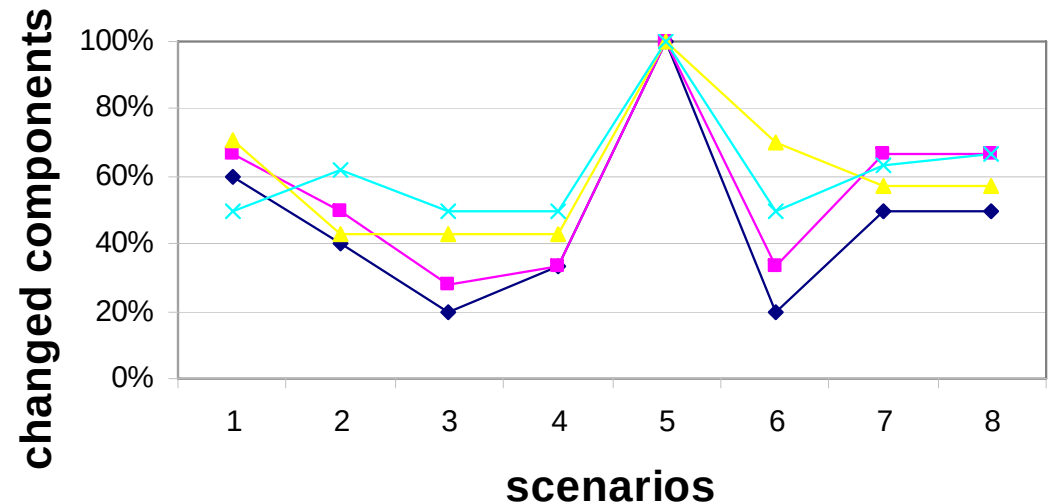
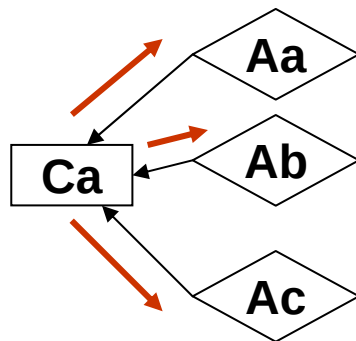
Key factor: “density” of crosscutting in the same join point

- Confirms somehow the findings of our previous implementation-level stability study [Greenwood, 2007]

Large inheritance trees



High Crosscutting density

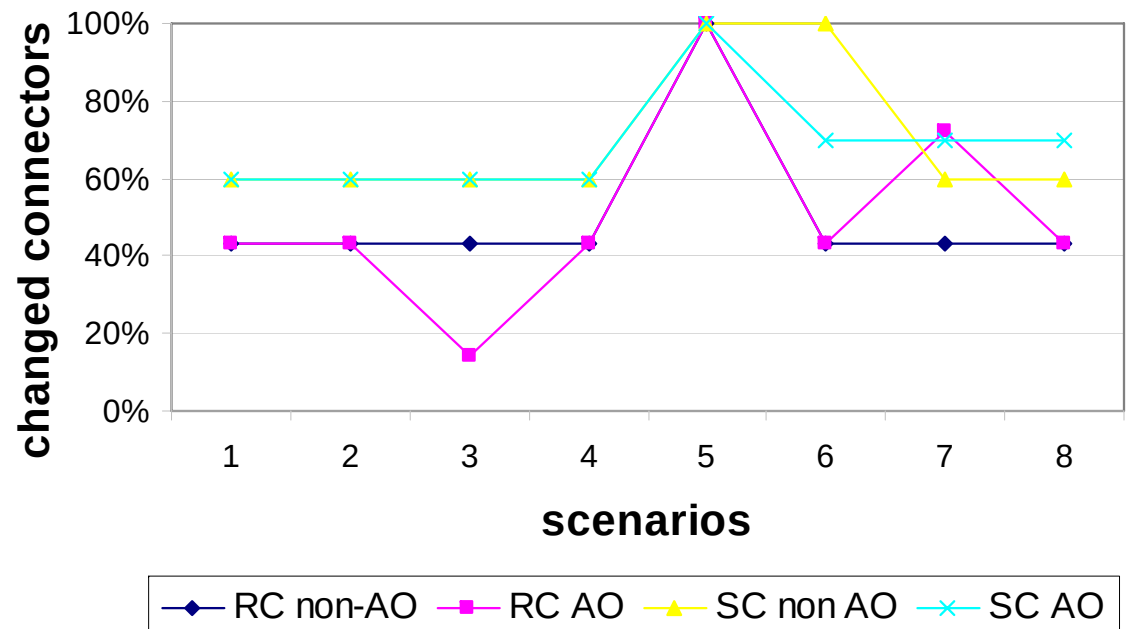
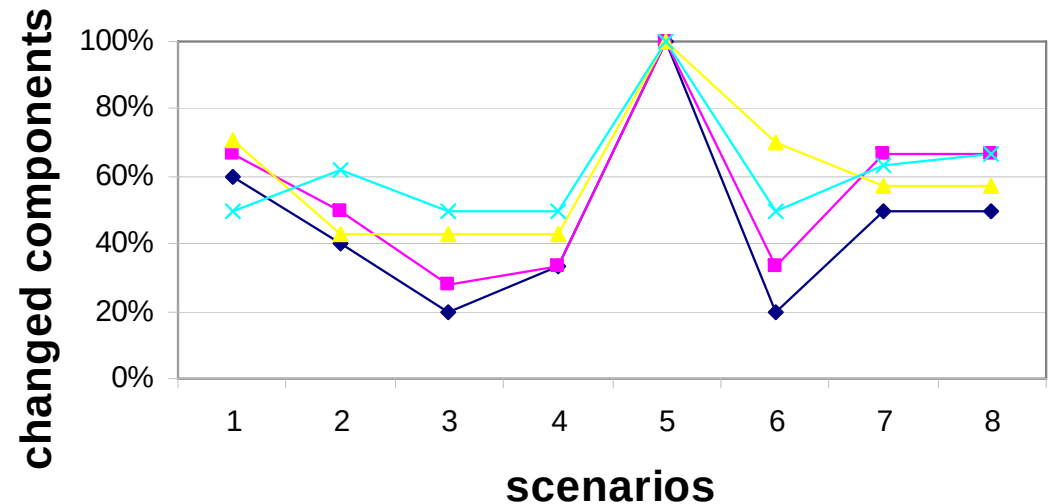
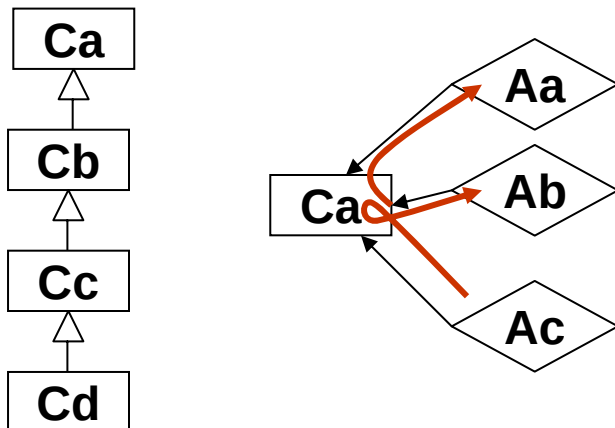


Analysis: Stability of Components/Connectors

Key factor: “density” of crosscutting in the same join point

- Confirms somehow the findings of our previous implementation-level stability study [Greenwood, 2007]

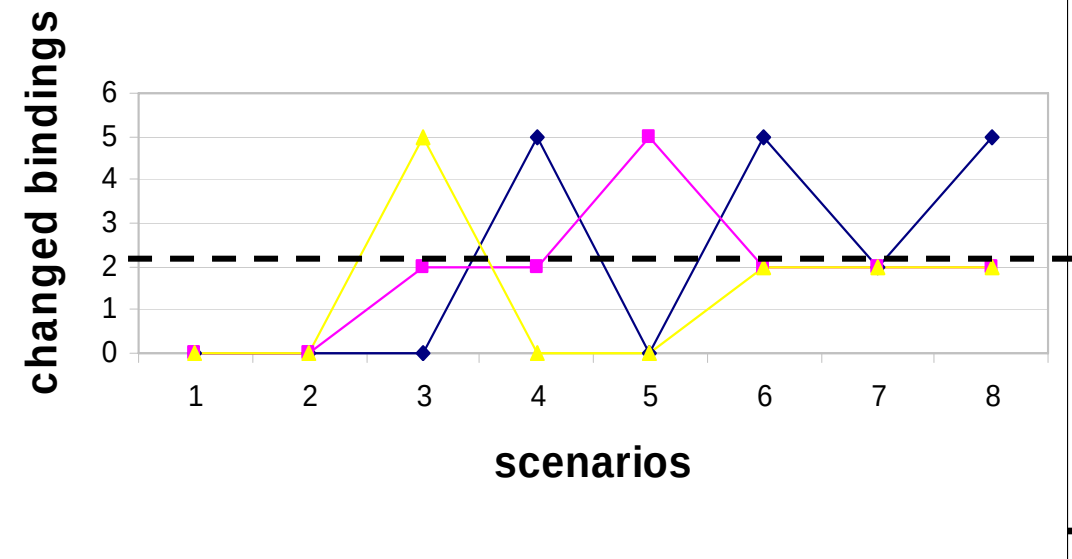
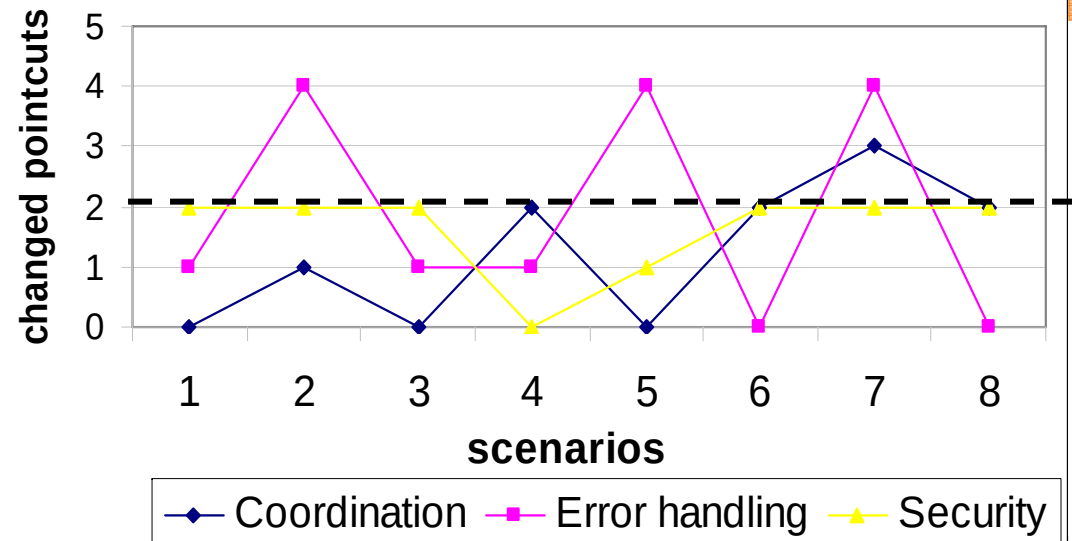
Large inheritance trees



Analysis: Stability of Bindings

Example:
Pointcuts in AO Blackboard
vs.
Bindings in Conventional
Blackboard

- Threshold > 2 changes
- AO – more stable
- Coordination:
AO – 1 ripple effect
non-AO – 3 ripple effects



Conclusions

- “Universal validity” of the conventional wisdom did not hold
- Aspectisation of architecturally-relevant exception handling was a consistent problem in all the architectural styles analysed
 - EJFlow – a design model for explicit definition of end-to-end architectural exception flows [Cacho et al, AOSD 2008]
 - Extended AspectJ (using abc)
 - Extend an ADL, e.g. AspectualACME
- The main outcomes were
 - AO architectures tend to be more stable when a change targets a CCC
 - AO architectures tended to require less invasive changes
 - AO architectures did not scale much when changes were made in non-CCC
 - High density of architectural crosscutting in specific join points should be avoided
- Are AspectJ-like pointcut declarations the most appropriate way?
- We have observed that pointcuts with higher degree of quantification had no impact on the (in)stability of aspect-oriented software architectures



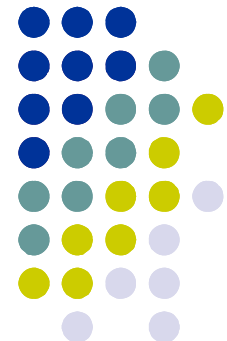
LANCASTER
UNIVERSITY



On the Quantitative Analysis of Architectural Stability in Aspectual Decomposition

**Ambra Molesini, *Alessandro Garcia*,
Christina Chavez and Thais Batista**

ambra.molesini@unibo.it, a.garcia@lancaster.ac.uk
flach@ufba.br, thais@ufrnet.br



Study Constraints, Imperfections and Future Work

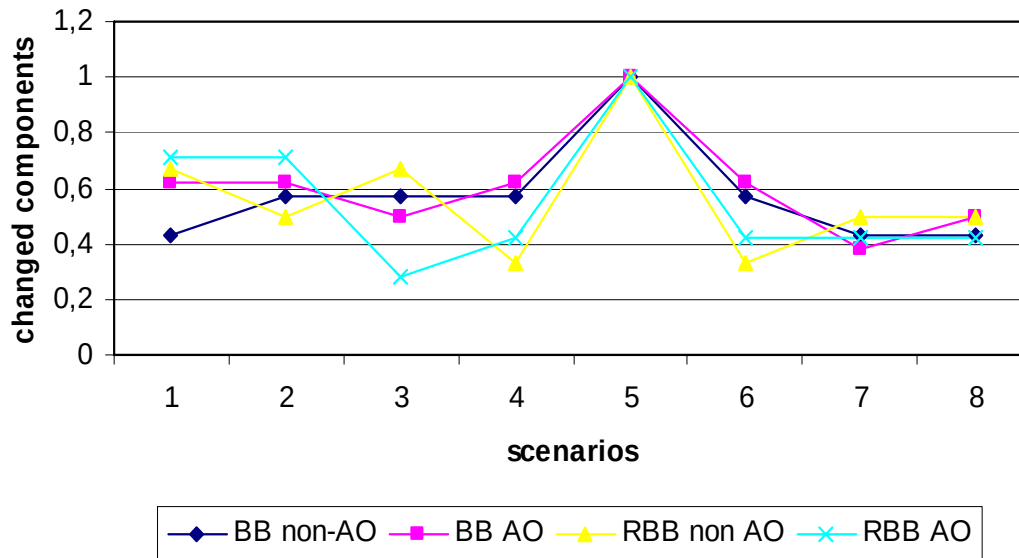


- Our analysis concentrated on:
 - First exploratory study: one target application
 - we cannot rely on statistically-relevant data
 - however...
 - we tried to observe to what extent similar phenomena was observed in other studies (e.g. Greenword, 2007)
 - we have used multiple AO and non-AO designs of the CMS architecture
 - Changes in the architectural specifications were inferred from the base implementation
 - Key aspect “quarks”: pointcuts and aspectual connectors (advice)
 - No exploration of inter-type declarations and merge operators

Other findings

- Certain stylistic compositions might ameliorate crosscutting
 - E.g. security in the non-AO reactive coordination architectures was stable
 - However, not in the non-AO stigmergic coordination architectures

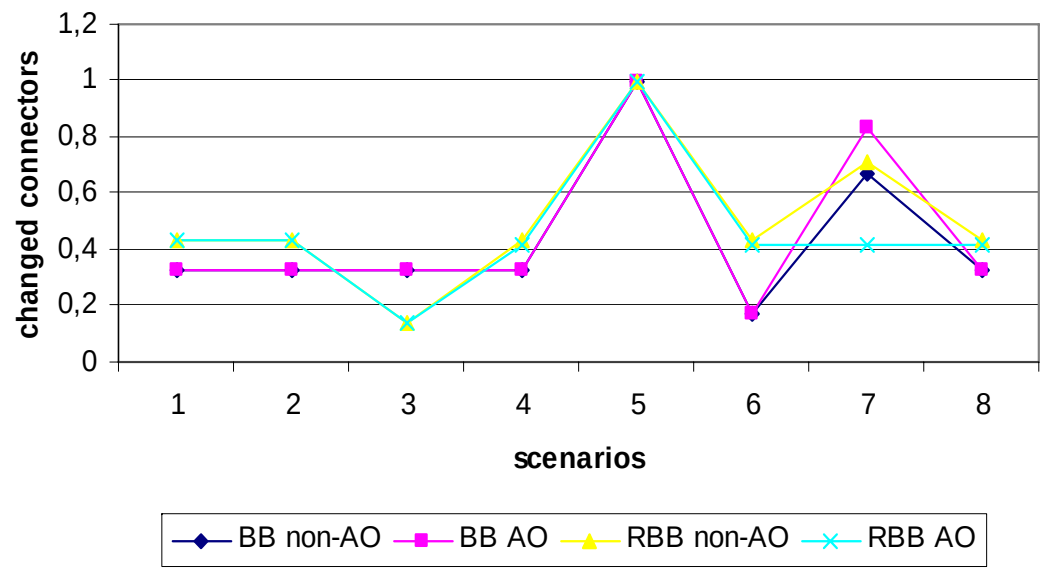
Empirical Results: BB and RBB



The graphics concentrate on the number of components and connectors changed because they have captured significant differences through the scenarios

AO architectures were sensibly more stable when the main focus of a change was a crosscutting concern.

Computing Department



Study Procedures - II



- Crosscutting concerns cut across the boundaries of modular units and tend to be *scattered* over several modules and *tangled up* with other concerns
- The natural consequences of crosscutting are
 - lower cohesion
 - stronger coupling between modular units
 - reduced comprehensibility evolvability and reusability of software
- Crosscutting concerns identified:
 - coordination
 - error handling
 - security

Pointcuts

Pointcut	Join Point
1. change coordination status	update blackboard signal status change
2. read coordination info	inspect blackboard
3. invoke coordination source	select source activate source

Aspect Oriented Blackboard

LANCASTER
UNIVERSITY



```
Family BlackboardFam  
extends SharedDataFam with {  
  Component Type BB  
  extends SharedDataT with {  
    Ports inspect, update : p_provide ... };  
  Component Type KS = {  
    Ports inspectBB, updateBB : p_use ...  
  };  
  Component Type Control = {... };  
  ...  
}
```

```
System CMS : BlackboardFam =  
new BlackboardFam extended with {  
  Component PC Chair: KS;  
  Component Rev1, Rev2, Rev3: KS;  
  ...  
  \* Coordination Crosscutting Concern *\br/>  Component Coordinator = {  
    Port changeCoordStatus: ... ;  
    Port readCoordInfo: ... ;  
    Port invokeCoordSource: ... ; }  
  Connector coord = f  
    crosscuttingRole source; baseRole sink;  
    glue source after sink; }  
  \* Pointcut description - aspectual bindings *\br/>  Attachments {  
    Coordinator.readCoordInfo to coord.source;  
    coord.sink to *.inspectBB;  
    ... } }
```



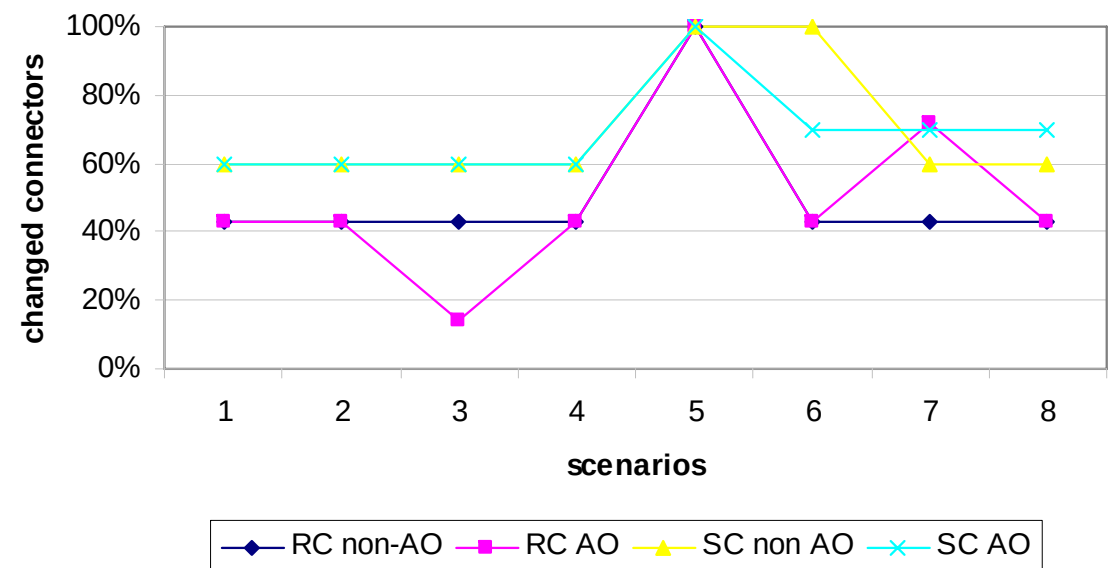
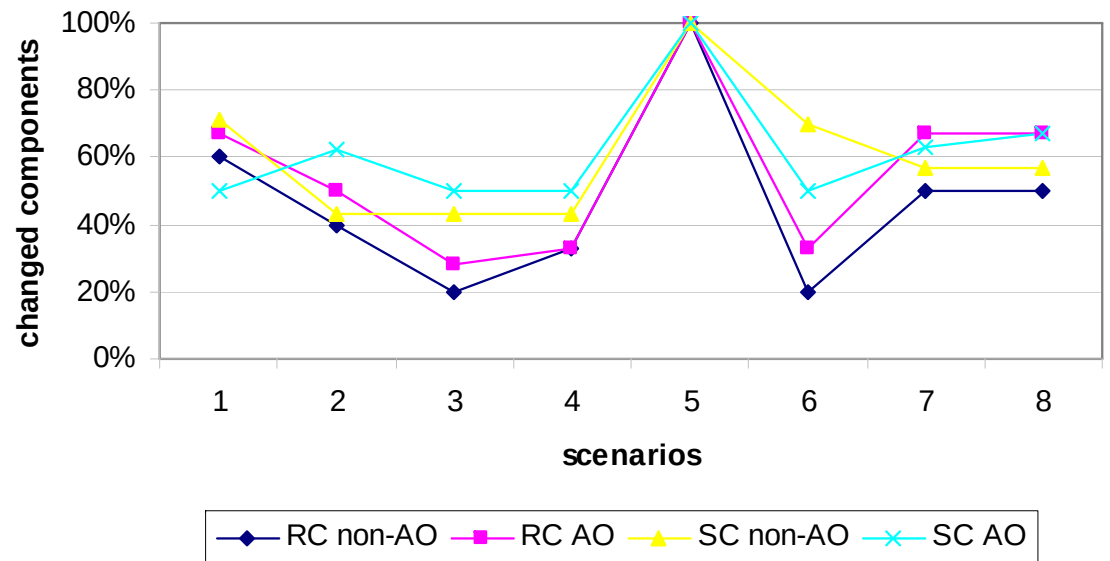
Analysis: RC and SC

Non-AO architectures tended to be more stable when the main focus of a change was a non crosscutting concern

Example: RC and SC

The superiority of non-AO architectures was independent of stylistic choices, join models, and types of changes

Computing Department



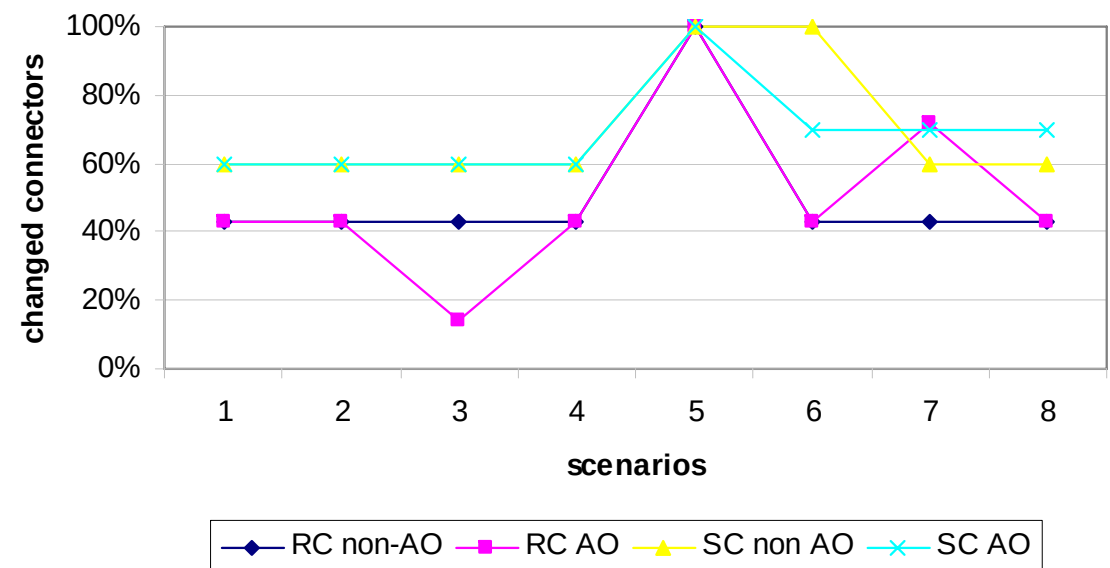
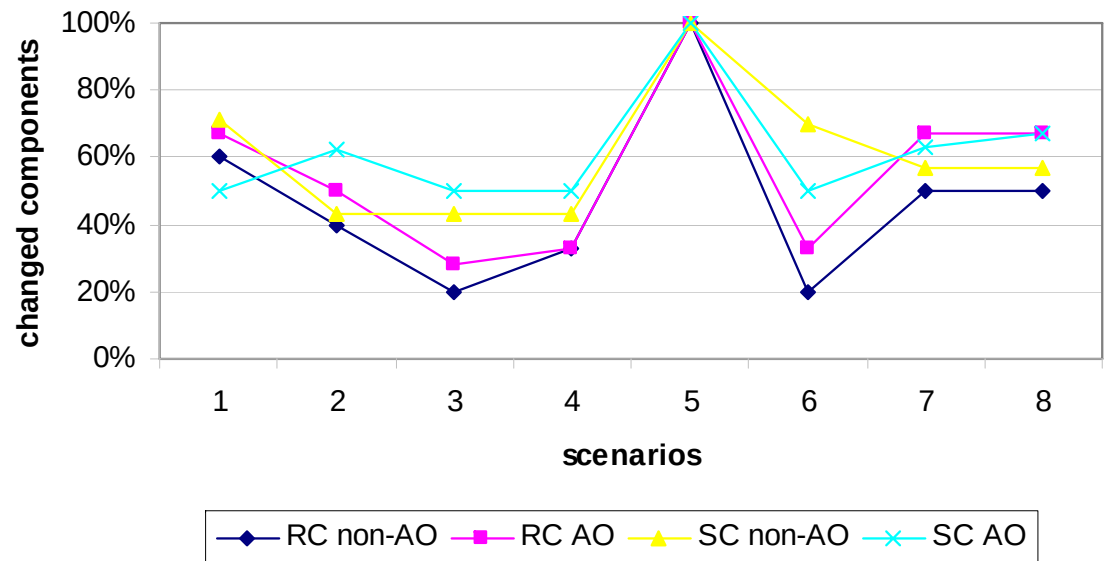
Analysis: RC and SC

Non-AO architectures tended to be more stable when the main focus of a change was a non crosscutting concern

Example: RC and SC

The superiority of non-AO architectures was independent of stylistic choices, join models, and types of changes

Computing Department

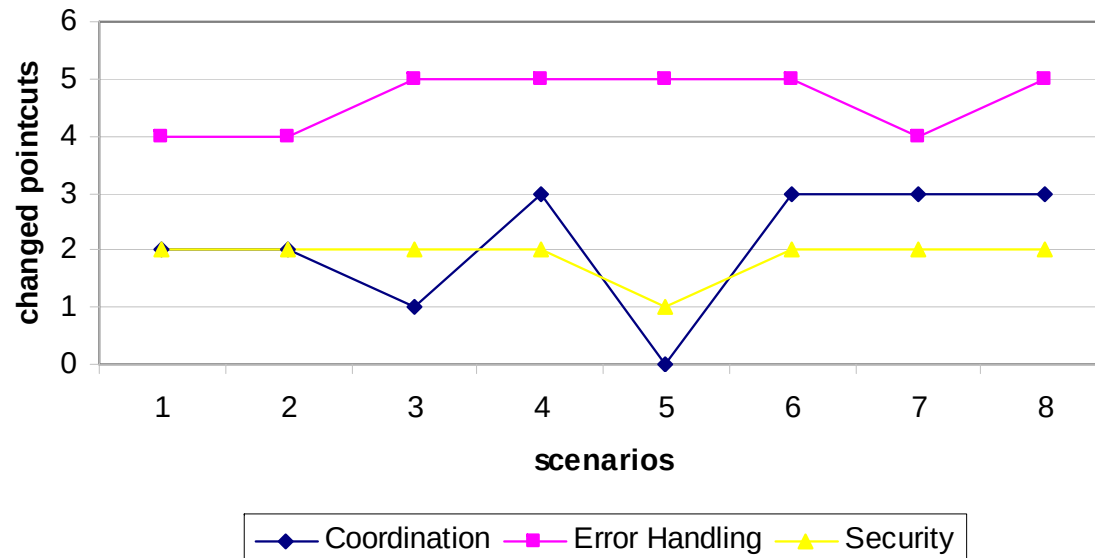


Empirical Results: RBB

LANCASTER
UNIVERSITY

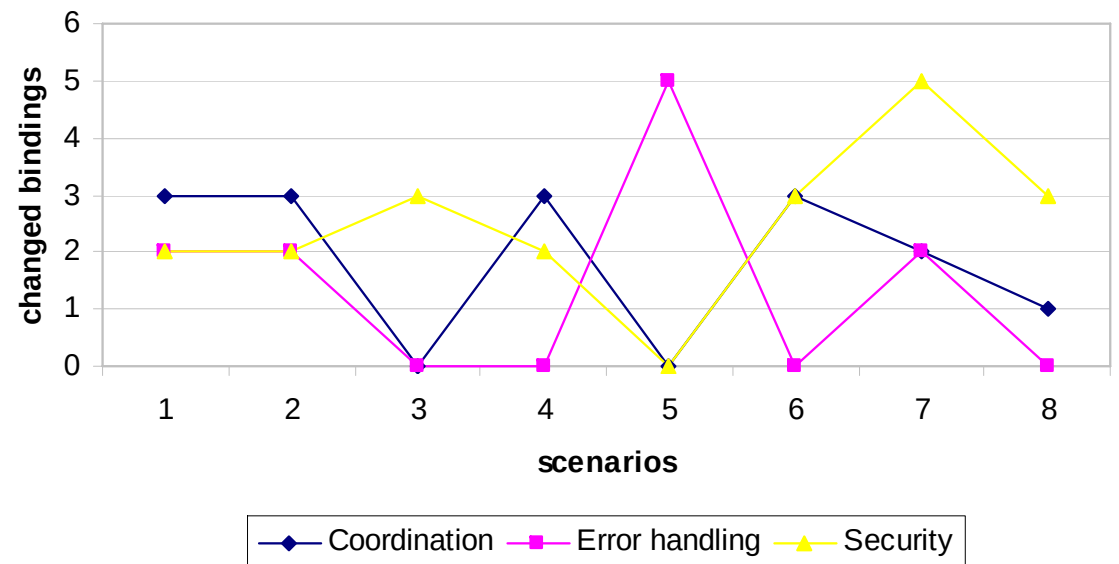


Pointcuts were more stable than conventional bindings.

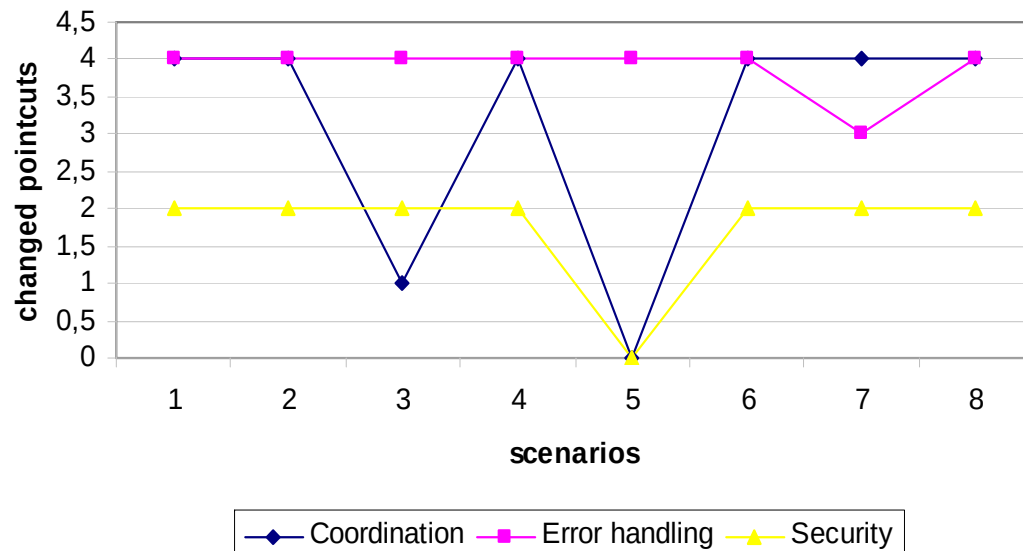


Major instabilities of coordination-specific bindings were observed in three scenarios against only one major coordination pointcut stability.

Computing Department

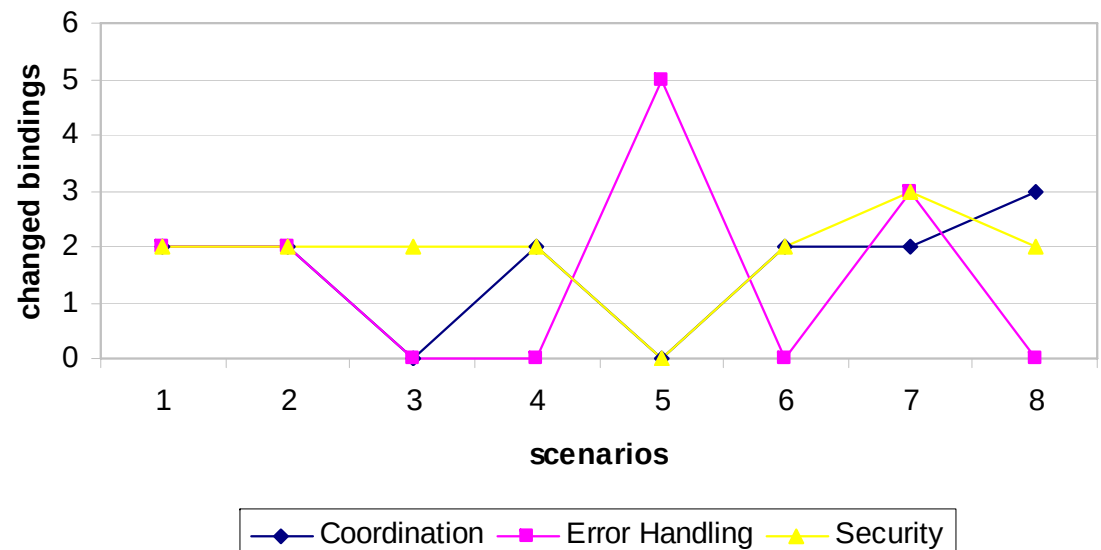


Empirical Results: RC

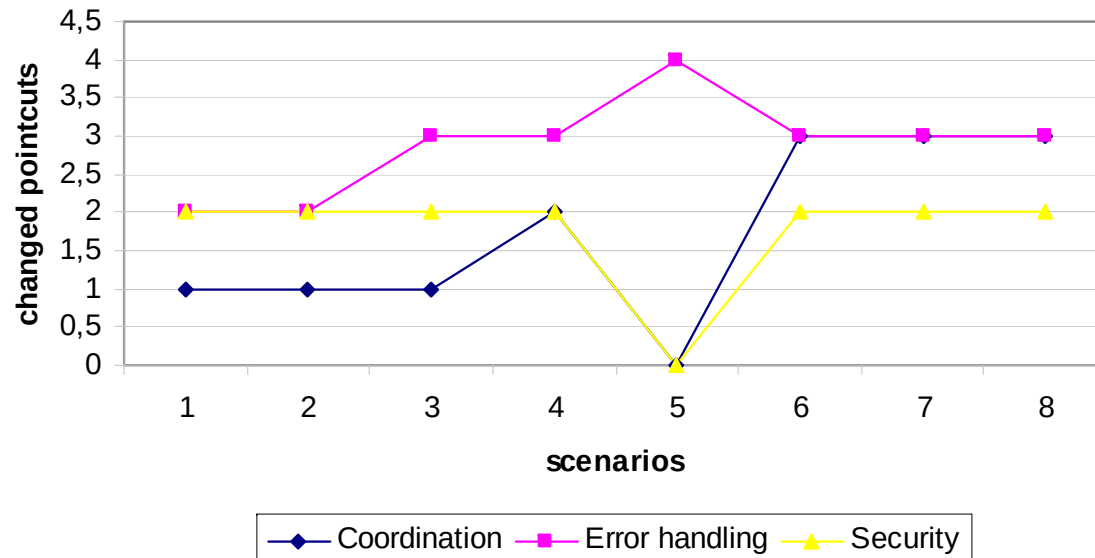


Composition of styles might ameliorate or eliminate the presence of crosscutting in non-AO architectures, sometimes more effectively than AO architectures.

Computing Department

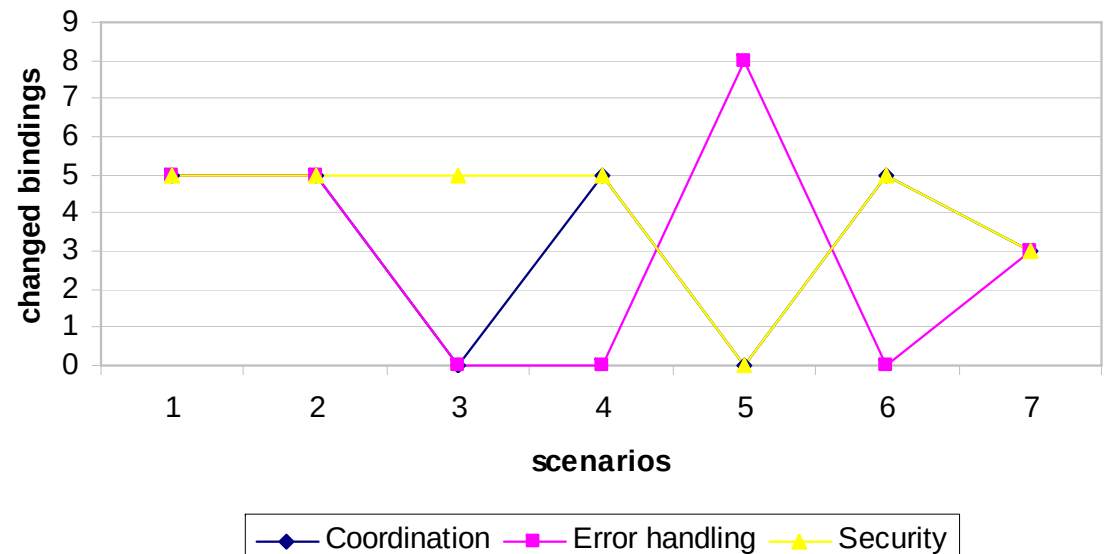


Empirical Results: SC



More instabilities were observed in the exception handling bindings in the AO versions than in the non-AO versions.

Computing Department



Conclusions

LANCASTER
UNIVERSITY



Computing Department



InfoLab21