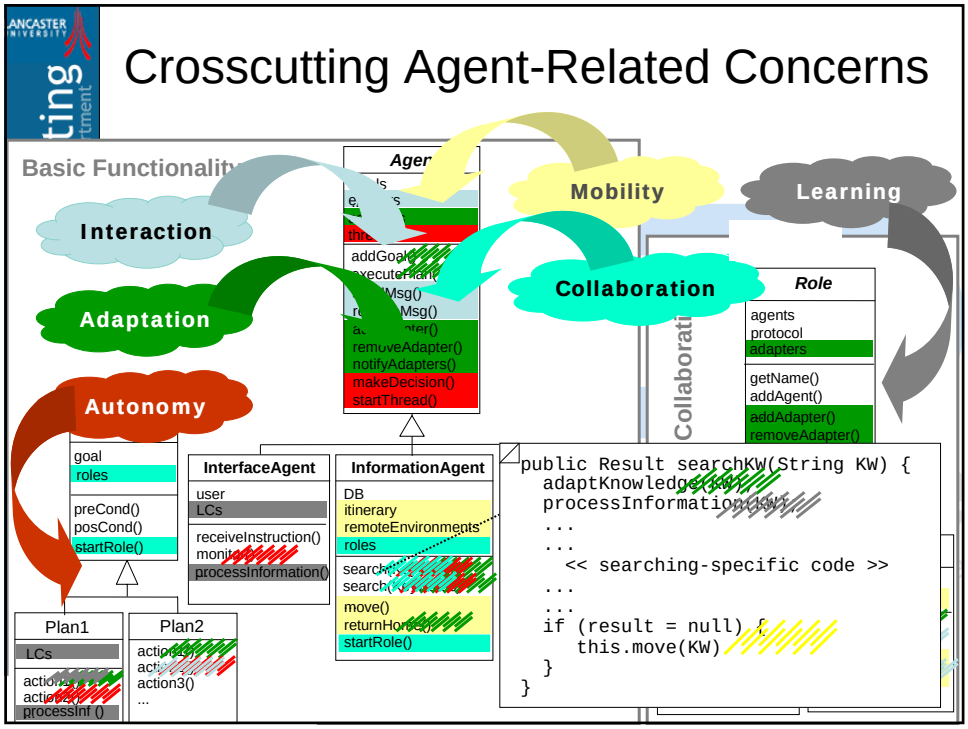
Aspects in Agent-Based Software Engineering: Lessons Learned

Alessandro Garcia
Lancaster University, UK
Uira Kulesza, Claudio Sant'Anna, Carlos Lucena
PUC-Rio, Brazil
Christina Chavez
UFBA, Brazil
garciaa@comp.lancs.ac.uk, [uira,claudios,lucena]@inf.puc-rio.br, flach@im.ufba.br

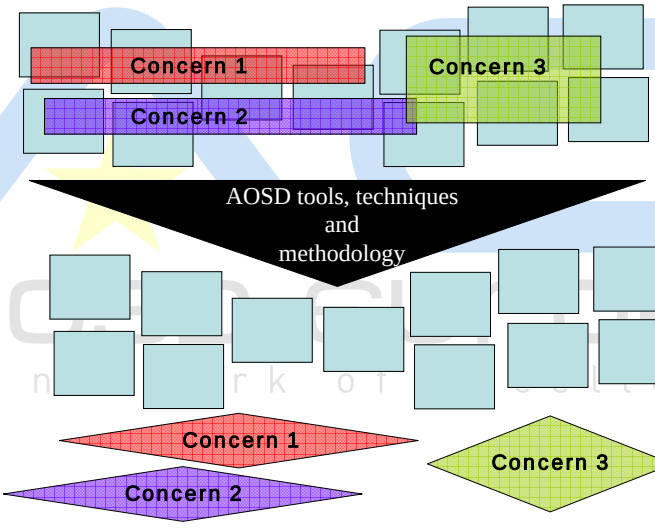
The Problem

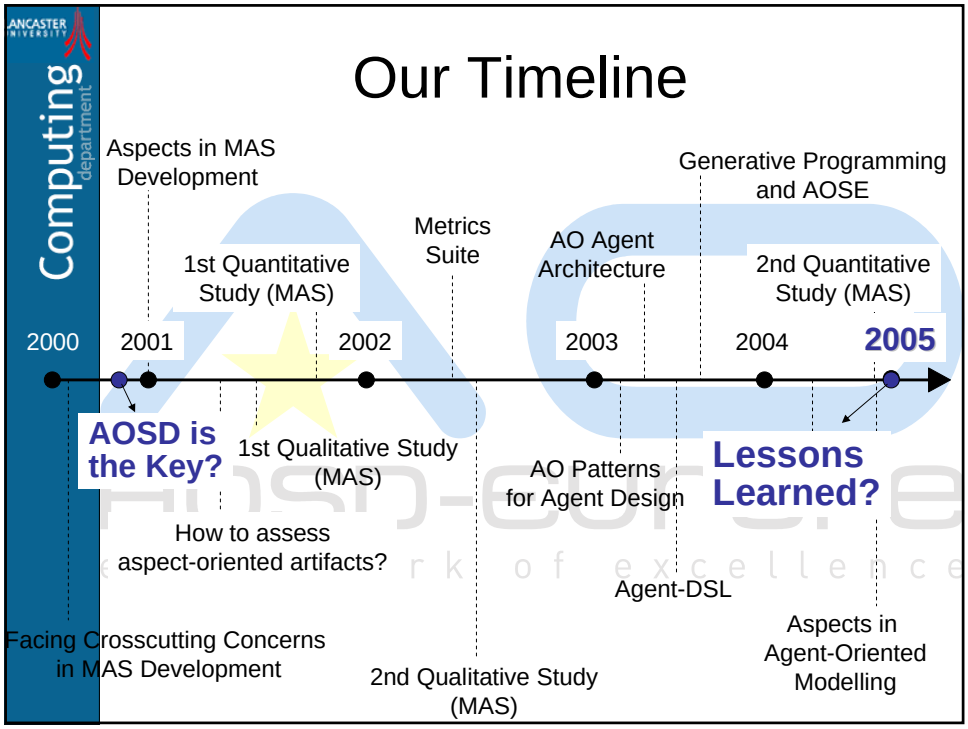
- Crosscutting concerns in agent-based development
  - Learning, code mobility, autonomy…
  - It does not matter which abstractions the software developers are relying on
    - object-oriented programming, AOSE abstractions
    - they are always crosscutting
  - Quality attributes depend largely on the modularity provided by SE abstractions
    - reusability
    - maintainability

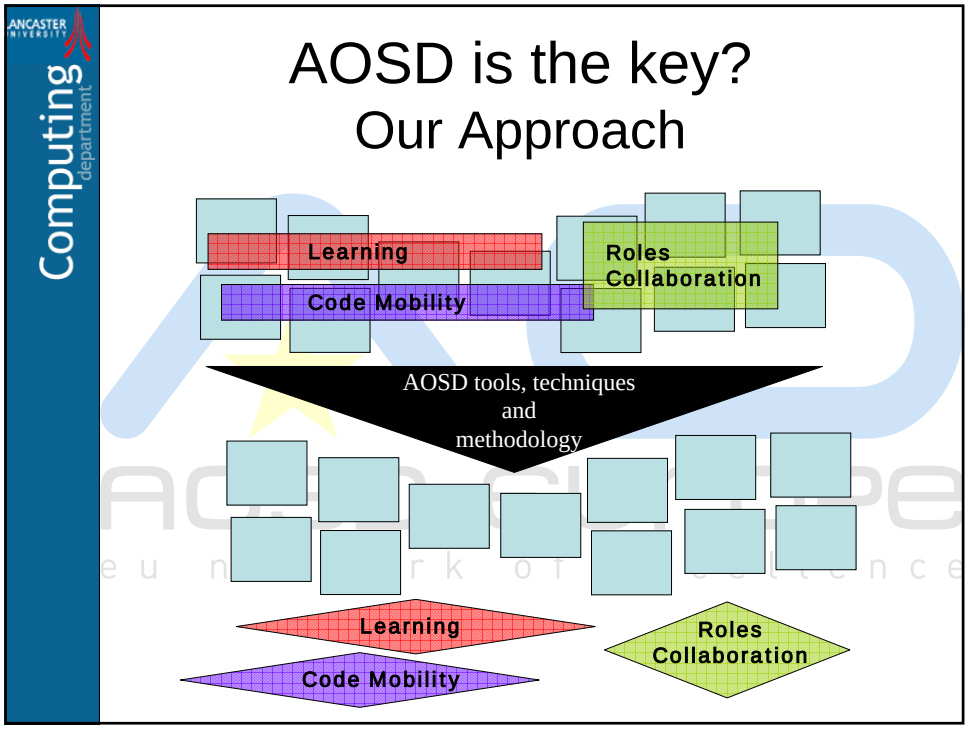
Crosscutting Agent-Related Concerns



Aspect-Oriented Software Development (AOSD)




Computing department
Our Timeline
Aspects in MAS Development
Generative Programming and AOSE
Metrics Suite
AO Agent Architecture
1st Quantitative Study (MAS)
2nd Quantitative Study (MAS)
2000
2001
2002
2003
2004
2005
AOSD is the Key?
1st Qualitative Study (MAS)
AO Patterns for Agent Design
Lessons Learned?
How to assess aspect-oriented artifacts?
Agent-DSL
Facing Crosscutting Concerns in MAS Development
2nd Qualitative Study (MAS)
Aspects in Agent-Oriented Modelling


Computing department
AOSD is the key?
Our Approach
Learning
Roles Collaboration
Code Mobility
AOSD tools, techniques and methodology
Learning
Roles Collaboration
Code Mobility

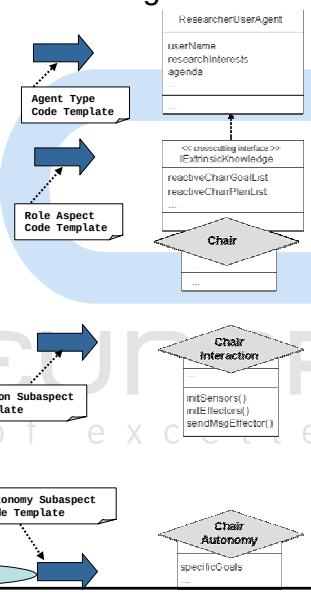
AOSD is the key?

- Seeking the answers…
  - Agent-DSL
  - Architectural approach
  - Design and implementation method
  - Design pattern language
  - AO generative approach
  - AspectT: Implementation Framework
  - Empirical studies
    - Qualitative ones: quality model
    - Quantitative ones: metrics suite

Our Aspect-Oriented Approach

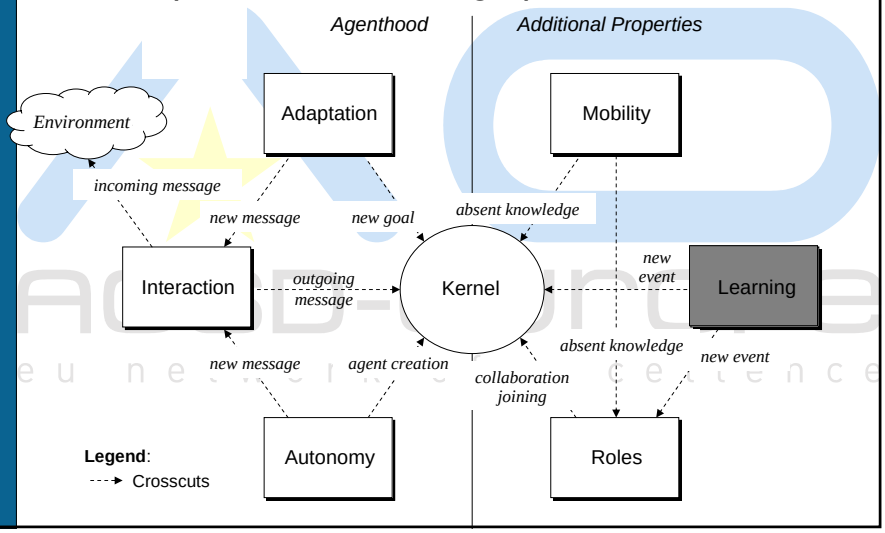
- Agent-DSL: modelling crosscutting features

```
<MAS ... xsi:noNamespaceSchemaLocation= "agent-dsl.xsd">
  <agent>
      <name>ResearcherUserAgent</name>
      <belief> ... </belief>
      <goal> ... </goal>

      <role>
         <name>Chair</name>
         <belief> ... </belief>
         <goal>
            <name>PaperDistributionGoal</name>
            <type>Reactive</type>
         </goal>
         <plan>
            <name>PaperDistributionPlan</name>
            <type>Reactive</type>
            <communication>false</communication>
         </plan>

        <interaction>
            <sensor>
               <name>sensorAgent</name>
               <type>AgentCommunication</type>
               <platform>JADE</platform>
            </sensor>
            <effector>
               <name>effectorAgent</name>
               <type>AgentCommunication</type>
               <platform>JADE</platform>
            </effector>
            <message>
               <id>REQUEST_DISTRIBUTE_PAPER</id>
               <performative>REQUEST</performative>
               <service> SERVICE_DISTRIBUTE_PAPERS </service>
            </message>
         </interaction>

      <autonomy>
         <executionAutonomy>
            <concurrencyStrategy> ThreadPool </c.S.>
         </executionAutonomy>
         <reactiveAutonomy>
            <messageToGoal>
               <message> REQUEST_DISTRIBUTE_PAPER </message>
```



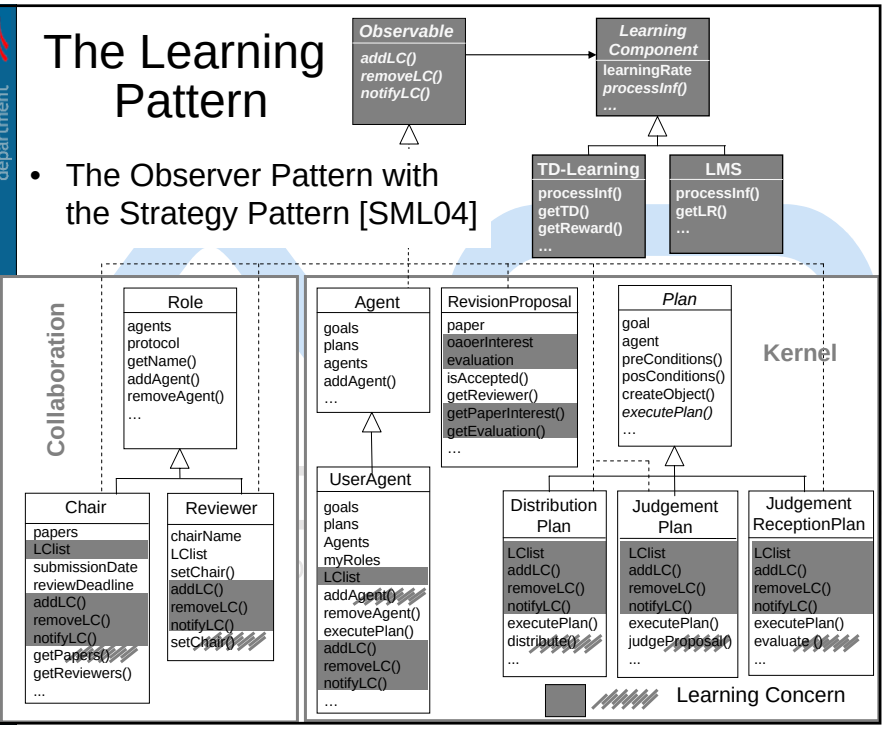
Our Approach

- Detailed Design
  - Aspect-oriented design patterns



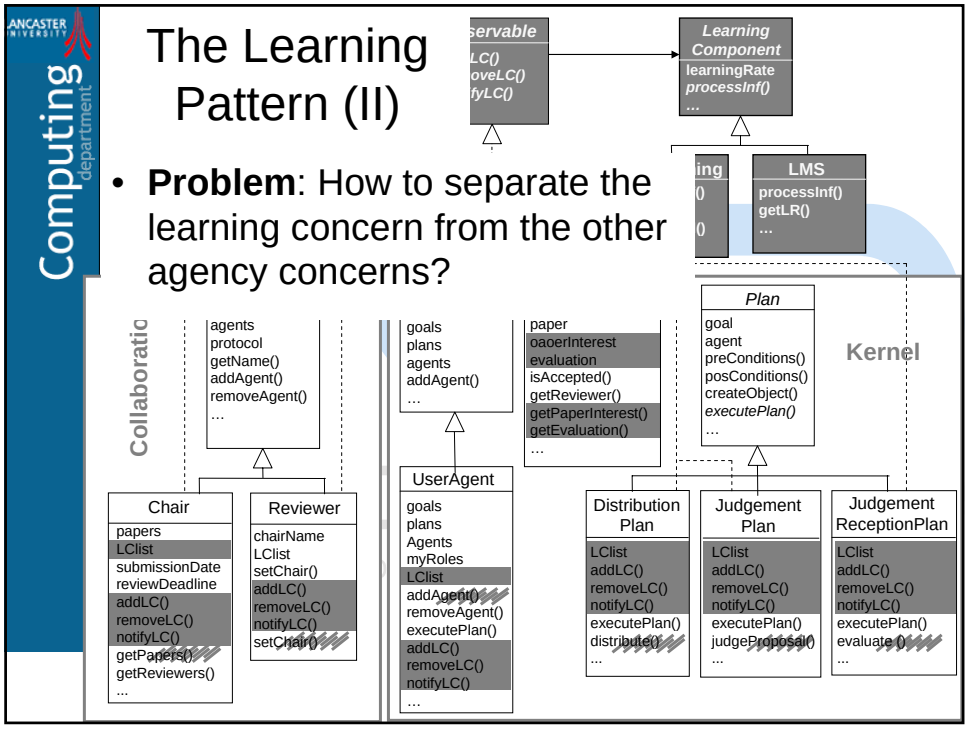
The Learning Pattern

- The Observer Pattern with the Strategy Pattern [SML04]



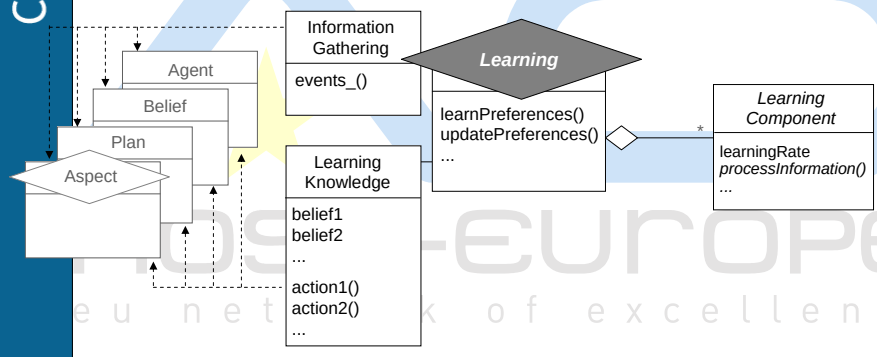
The Learning Pattern (II)

- **Problem**: How to separate the learning concern from the other agency concerns?



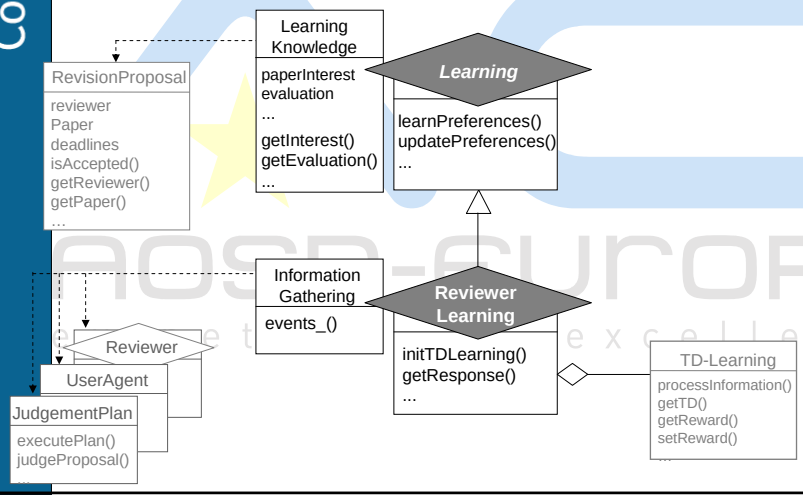
The Learning Pattern (III)

- **Solution**:



The Pattern Language

- **Abstract aspects**
  - **Refining the aspects to different contexts**



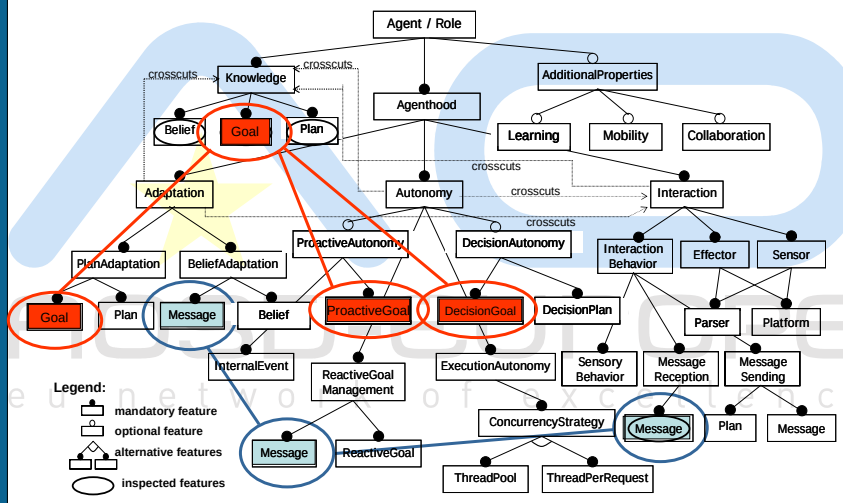
What we have learned?

1. What are the main **motivations** to use AOSD techniques for MAS development?
2. **Successful aspectization** of MAS-related concerns?
3. **Limitations** of existing AOSD techniques to address MAS crosscutting concerns?
4. **Future directions** emerged from our practical exploration of AOSD in AOSE?

Main motivations?

- Crosscutting phenomena manifests early in the software lifecycle
  - domain engineering (agent-oriented product lines)
    - Crosscutting agent features
      - Affect other feature trees

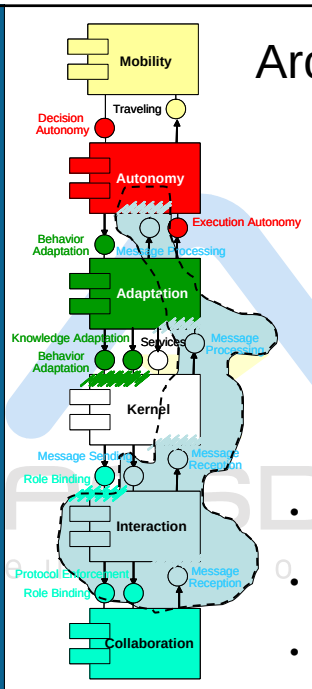
Crosscutting in Domain Modelling



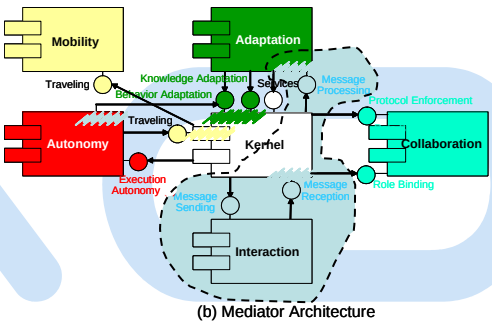
Main motivations?

- Crosscutting phenomena manifests early in the software lifecycle
  - domain engineering (agent-oriented product lines)
    - Crosscutting agent features
      - Affect other feature trees
  - software architecture (ADL and UML-based representations)
    - Crosscutting architectural concerns of a software agent
      - Affect several architectural components and their interfaces

Architectural Crosscutting



(a) Layered Architecture



(b) Mediator Architecture

- Leads to tangling and scattering at the agent architecture description
- Hinders modular reasoning on architectural agent properties
- Causes architectural interface bloat

Main motivations?

- Maintainable and reusable MASs
  - Finer granularity
  - Allow for reuse and variation of those concerns
- Adaptive MAS in open environments
  - Dynamic reconfiguration
    - roles
    - collaboration protocols
  - Context-aware learning strategies
- Full modularization of crosscutting concerns across MAS lifecycle

What we have learned?

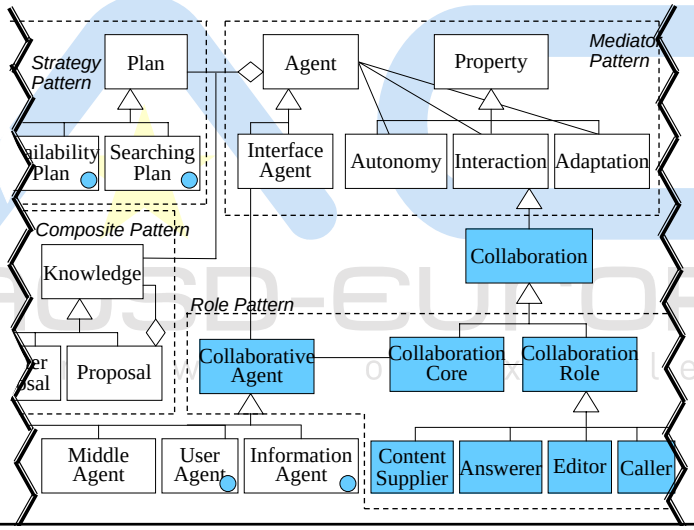
1. What are the main **motivations** to use AOSD techniques for MAS development?
2. **Successful aspectization** of MAS-related concerns?
3. **Limitations** of existing AOSD techniques to address MAS crosscutting concerns?
4. **Future directions** emerged from our practical exploration of AOSD in AOSE?

Successful aspectization?

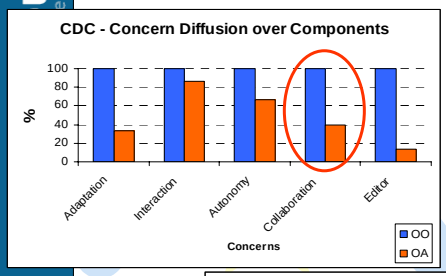
- Detailed design and implementation
  - AspectJ and Hyper/J
  - Separable concerns
    - Code mobility, learning, roles, autonomy
    - Autonomy
      - thread management (execution autonomy)
      - triggering of proactive behaviour (proactive autonomy)
      - triggering of decision-making plans (decision autonomy)
    - Roles
      - Separation of intrinsic and extrinsic knowledge (roles)
      - role binding
      - protocol enforcement

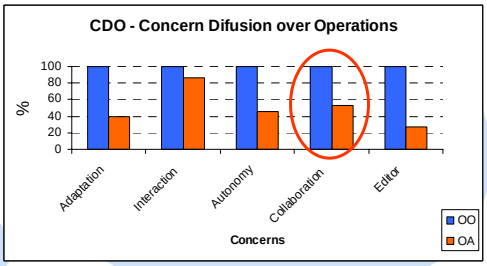
Quantitative Evaluation

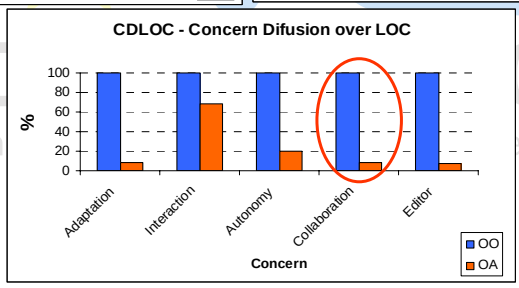
- Viewpoint: Role-related Concerns



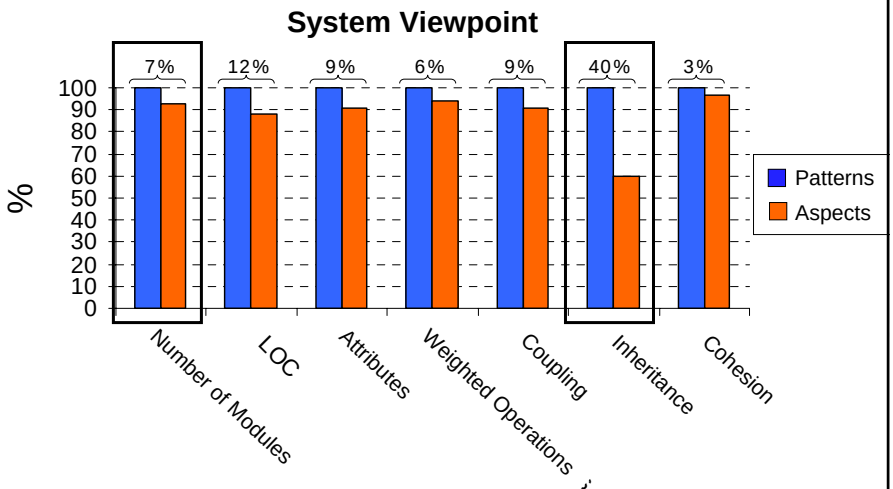
Quantitative Evaluation





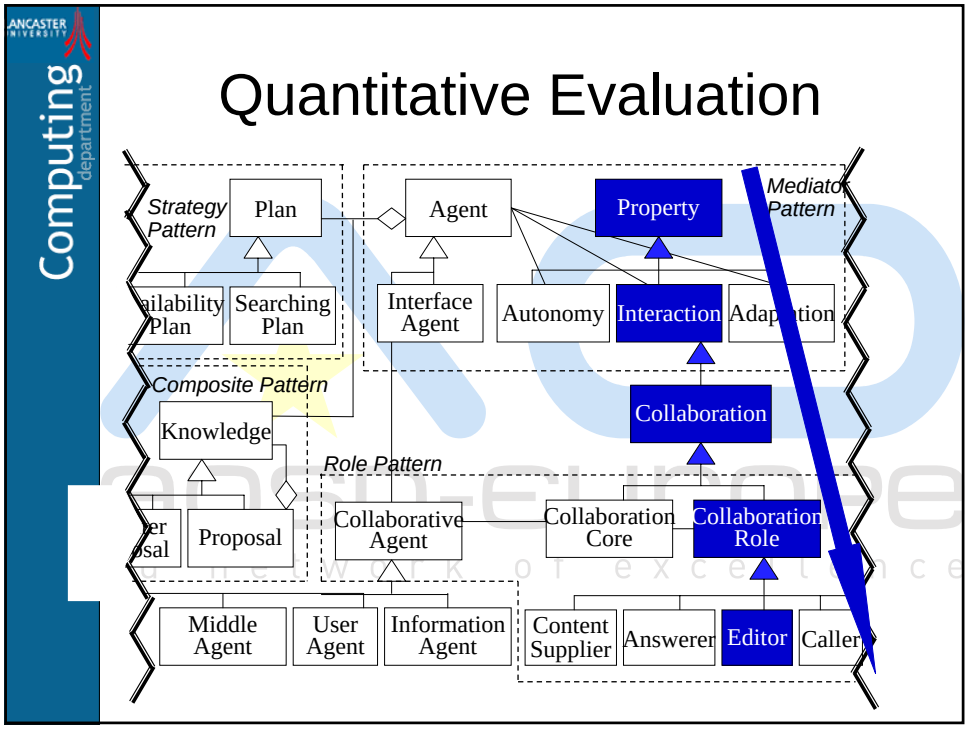


Quantitative Evaluation

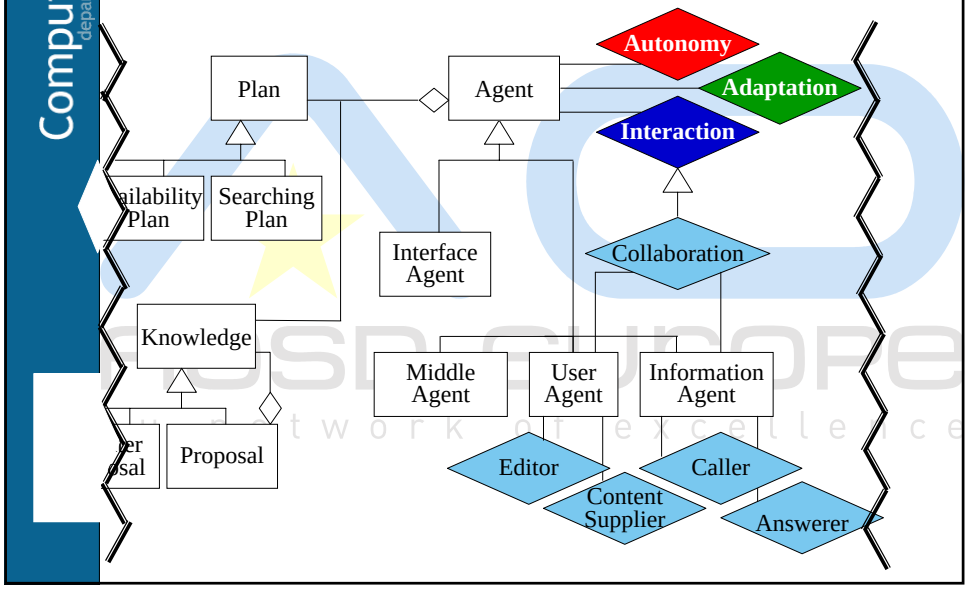


GARCIA, A. et al. **Separation of Concerns in Multi-Agent Systems: An Empirical Study**. In: Software Engineering for Multi-Agent Systems II, Springer, LNCS 2940, March 2004, pp. 49-72.

Quantitative Evaluation



Quantitative Evaluation



Successful aspectization?

- "Inseparable concerns"
  - Interaction
  - message assembling from different plans/roles
    - very coupled to the role or plan context
- the composition rules is the central contribution of AOSD
  - conclusions specific to the AspectJ join point model

Successful aspectization?

- Inter-aspect relationships
  - mobility does not affect only the agent kernel
    - also crosscuts roles, interaction, and learning
    - AspectJ-oriented abstractions forces indirect interaction, which sometimes complicates the design
- Complex solution for simple reactive agents
  - crosscutting concern is not evident
    - e.g. autonomy-related behaviour tends to be localized in few methods

Successful aspectization?

- Aspects as the “glue”
  - agent kernel vs. concern-specific implementation framework/platform (e.g. Aglets/Jade for mobility)
- Incremental process vs. iterative process
  - we mostly had an iterative process
  - lack of obliviousness

What we have learned?

1. What are the main **motivations** to use AOSD techniques for MAS development?
2. **Successful aspectization** of MAS-related concerns?
3. **Limitations** of existing AOSD techniques to address MAS crosscutting concerns?
4. **Future directions** emerged from our practical exploration of AOSD in AOSE?

Limitations?

- Our Agent-DSL requires more powerful composition rules
  - ideally their specification should be separated from the aspects themselves
- Aspects in agent-oriented ADLs
  - to express architectural aspects in MASs
- Tropos/i* somewhat deal with aspects at requirements stage
  - softgoals
  - But… need for enhancing agent-oriented design languages with aspects
    - Gaia, AUML, MAS-ML, …
- Isolate solutions
  - e.g. Kendall et al – aspectization of agent roles

Limitations?

- Detailed design and implementation
  - Inter-aspect conflicts in the composition
    - **roles** – introduction of similar behaviour
    - **mobility** – composition of agent kernel and abstract methods of mobility frameworks
      - e.g. *getName() – JADE*
  - Repetitive and time-consuming tasks
    - **interaction**
      - enumeration of all message senders in the aspects
      - our 1st solution: naming convention
      - our 2nd solution: code generators based on the Agent-DSL descriptions

What we have learned?

1. What are the main **motivations** to use AOSD techniques for MAS development?
2. **Successful aspectization** of MAS-related concerns?
3. **Limitations** of existing AOSD techniques to address MAS crosscutting concerns?
4. **Future directions** emerged from our practical exploration of AOSD in AOSE?

Future Directions

- We have investigated the aspectization of internal agent properties
  - systemic properties: coordination, fault-tolerance, …
- Need for improved traceability
  - Aspects in agent-oriented modelling
    - agent-oriented meta-models
    - modelling languages
  - Code generation

Concluding remarks

- In a stage where we are facing a number of growing AOSE abstractions…
  - Is not the case to try to understand their commonalities?
  - We see aspect-oriented abstractions and composition mechanisms as promising candidates
    - Many of those abstractions are in fact to address crosscutting concerns
      - Autonomy, roles, coordination, mobility…
  - However, we still need to investigate effective composition rules/mechanisms to the context of AOSE

Aspects in Agent-Based Software Engineering: Lessons Learned

Alessandro Garcia
Lancaster University, UK
Uira Kulesza, Claudio Sant'Anna, Carlos Lucena
PUC-Rio, Brazil
Christina Chavez
UFBA, Brazil
garciaa@comp.lancs.ac.uk, [uira,claudios,lucena]@inf.puc-rio.br, flach@im.ufba.br