Agent Oriented Software Engineering

Ambra Molesini

Alma Mater Studiorum – Università di Bologna (Italy)
ambra.molesini@unibo.it

Bologna, Italy, 28th June 2010
Outline

- Part I: General Concepts
- Part II: Agent Oriented Software Engineering
- Part III: AOSE Methodologies
- Part IV: Situational Method Engineering
- Part V: Research directions and conclusions
Outline of Part I

1. Software Engineering
2. Software Process
3. Methodologies
Outline

- Part I: General Concepts
- **Part II: Agent Oriented Software Engineering**
- Part III: AOSE Methodologies
- Part IV: Situational Method Engineering
- Part V: Research directions and conclusions
Outline of Part II

4 Why do we need AOSE?

5 Why agents and MAS?
Outline

- Part I: General Concepts
- Part II: Agent Oriented Software Engineering
- Part III: AOSE Methodologies
- Part IV: Situational Method Engineering
- Part V: Research directions and conclusions
Outline of Part III

6 Meta-model

7 AOSE Methodologies
- ADELFÉ
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
Outline

- Part I: General Concepts
- Part II: Agent Oriented Software Engineering
- Part III: AOSE Methodologies
- **Part IV: Situational Method Engineering**
- Part V: Research directions and conclusions
Outline of Part IV

8 Methodologies Documentation

9 Situational Method Engineering

10 Situational Method Engineering in AOSE
   - SPEM and AOSE processes
   - Method Fragment Representation
   - PRODE: PROcess DEsign for design processes

11 Result Evaluation
Outline

- Part I: General Concepts
- Part II: Agent Oriented Software Engineering
- Part III: AOSE Methodologies
- Part IV: Situational Method Engineering
- Part V: Research directions and conclusions
Outline of Part V

12. Research directions and visions

13. Conclusions

14. List of interesting papers
Part I

General Concepts
Outline

1. Software Engineering

2. Software Process

3. Methodologies
Software Engineering

What is Software Engineering?

Software Engineering is an engineering discipline concerned with theories, methods and tools for professional software development [Sommerville, 2007]
Software Engineering

What is Software Engineering?

Software Engineering is an engineering discipline concerned with theories, methods and tools for professional software development [Sommerville, 2007]

What is the aim of Software Engineering?

Software Engineering is concerned with all aspects of software production from the early stage of system specification to the system maintenance / incremental development after it has gone into use [Sommerville, 2007]
Software Engineering: Concerns

- There is a need to *model* and *engineer* both
  - the development process
    - controllable, well documented, and reproducible ways of producing software
  - the software
    - ensuring a given level of quality (e.g., % of errors and performances)
    - enabling reuse, maintenance, and incremental development

- This requires suitable
  - abstractions
  - tools
Software Engineering Abstractions

- Software deals with *abstract entities*, having a real-world counterpart:
  - numbers, dates, names, persons, documents...
- In what terms should we model them in software?
  - data, functions, objects, agents...
  - i.e., what are the **ABSTRACTIONS** we should use to model software?
- May depend on the available technologies!
  - use OO abstractions for OO programming envs
  - not necessarily: use OO abstractions because they are better, even for COBOL programming envs
Tools

- *Notation tools* represent the outcomes of the software development
  - diagrams, equations, figures...
- *Formal models* prove properties of software prior to the development
  - lambda-calculus, pi-calculus, Petri nets...
- *CASE tools* are based on notations and models to facilitate activities
  - simulators
- *CAPE tools* support process engineering during the construction of new software development processes
Computer science is concerned with theory and fundamentals—*modelling* computational systems.

Software engineering is concerned with the practicalities of developing and delivering useful software—*building* computational systems.

Deep knowledge of computer science is essential for software engineering in the same way that deep knowledge of physics is essential for electric engineers.

Ideally, all of software engineering should be underpinned by theories of computer science... but this is not the case, in practice.

Software engineers must often use *ad hoc* approaches to developing software systems.

Elegant theories of computer science cannot always be applied to real, complex problems that require a software solution [Sommerville, 2007].
System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering.

System engineers are involved in system specification, architectural design, integration and deployment—they are less concerned with the engineering of the system components.

Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system [Sommerville, 2007].
Outline

1. Software Engineering

2. Software Process

3. Methodologies
Development Process

The development process is an ordered set of steps that involve all the activities, constraints and resources required to produce a specific desired output satisfying a set of input requirements.

Typically, a process is composed by different stages/phases put in relation to each other.

Each stage/phase of a process identifies a portion of work definition to be done in the context of the process, the resources to be exploited to that purpose and the constraints to be obeyed in the execution of the phase.

Case by case, the work in a phase can be very small or more demanding.

Phases are usually composed by a set of activities that may, in turn, be conceived in terms of smaller atomic units of work (steps).
The software development process is the coherent set of policies, organisational structures, technologies, procedures and deliverables that are needed to conceive, develop, deploy and maintain a software product.
Software Process: Concepts

The software process exploits a number of contributions and concepts [Fuggetta, 2000]

Software development technology — Technological support used in the process. Certainly, to accomplish software development activities we need tools, infrastructures, and environments.

Software development methods and techniques — Guidelines on how to use technology and accomplish software development activities. The methodological support is essential to exploit technology effectively.

Organisational behavior — The science of organisations and people.

Marketing and economy — Software development is not a self-contained endeavor. As any other product, software must address real customers’ needs in specific market settings.
Generic activities in all software processes are [Sommerville, 2007]:

**Specification** — What the system should do and its development constraints

**Development** — Production of the software system

**Validation** — Checking that the software is what the customer wants

**Evolution** — Changing the software in response to changing demands
Specification

- The process of establishing what services are required and the constraints on the system’s operation and development
- Requirements engineering process
  - application domain study
  - feasibility study
  - requirements elicitation and analysis
  - requirements specification
  - requirements validation
Development

- The process of converting the system specification into an executable system
- Software design
  - analyse the problem in order to create the first logical architecture
  - design a software structure/architecture that realises the specification
- Design activities
  - architectural design
  - abstract specification
  - interface design
  - component design
  - data structure design
  - algorithm design
- Implementation
  - translate the system architecture into an executable program
- The activities of design and implementation are closely related and may be inter-leaved
Validation

- Verification and validation is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
  - Component or unit testing
    - Individual components are tested independently.
    - Components may be functions or objects or coherent groupings of these entities.
  - System testing
    - Testing of the system as a whole.
    - Testing of emergent properties is particularly important.
  - Acceptance testing
    - Testing with customer data to check that the system meets the customer’s needs.
Evolution

- Software is inherently flexible and can change
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new
There is no an ideal process

[Sommerville, 2007]
Many Sorts of Software Processes

- Different types of systems require different development processes [Sommerville, 2007]
  - real time software in aircrafts has to be completely specified before development begins
  - in e-commerce systems, the specification and the program are usually developed together
- Consequently, the generic activities, specified above, may be organised in different ways, and described at different levels of details for different types of software
- The use of an inappropriate software process may reduce the quality or the usefulness of the software product to be developed and/or increased
Software Process Model

- A **Software Process Model** is a simplified representation of a software process, presented from a specific perspective [Sommerville, 2007].
- A process model prescribes which phases a process should be organised around, in which order such phases should be executed, and when interactions and coordination between the work of the different phases should be occur.
- In other words, a process model defines a skeleton, a template, around which to organise and detail an actual process.
Software Process Model: Examples

- Examples of process models are
  - Workflow model — this shows sequence of activities along with their inputs, outputs and dependencies
  - Activity model — this represents the process as a set of activities, each of which carries out some data transformation
  - Role/action model — this depicts the roles of the people involved in the software process and the activities for which they are responsible
Generic Software Process Models

• Generic process models
  
  **Waterfall** — separate and distinct phases of specification and development
  
  **Iterative development** — specification, development and validation are interleaved
  
  **Component-based software engineering** — the system is assembled from existing components
Outline

1. Software Engineering
2. Software Process
3. Methodologies
Methodologies vs. Methods: General Issue

- Disagreement exists regarding the relationship between the terms *method* and *methodology*.
- In common use, methodology is frequently substituted for method; seldom does the opposite occur.
- Some argue this occurs because methodology sounds more scholarly or important than method.
- A footnote to *methodology* in the 2006 American Heritage Dictionary notes that:
  - *the misuse of methodology obscures an important conceptual distinction between the tools of scientific investigation (properly methods) and the principles that determine how such tools are deployed and interpreted (properly methodologies)*
Methodologies vs. Methods in Software Engineering

In Software Engineering the discussion continues...

- some authors argue that a software engineering method is a recipe, a series of steps, to build software, while a methodology is a codified set of recommended practices. In this way, a software engineering method could be part of a methodology
- some authors believe that in a methodology there is an overall philosophical approach to the problem. Using these definitions, Software Engineering is rich in methods, but has fewer methodologies
Method

Method [Cernuzzi et al., 2005]

- A method prescribes a way of performing some kind of activity within a process, in order to properly produce a specific output (i.e., an artefact or a document) starting from a specific input (again, an artefact or a document).

- Any phase of a process, to be successfully applicable, should be complemented by some methodological guidelines (including the identification of the techniques and tools to be used, and the definition of how artifacts have been produced) that could help the involved stakeholders in accomplishing their work according to some defined best practices.
Methodology

Methodology [Ghezzi et al., 2002]

- A methodology is a collection of methods covering and connecting different stages in a process.
- The purpose of a methodology is to prescribe a certain coherent approach to solving a problem in the context of a software process by preselecting and putting in relation a number of methods.
- A methodology has two important components:
  - one that describes the process elements of the approach.
  - one that focuses on the work products and their documentation.
Methodologies vs. Software Process

- Based on the above definitions, and comparing software processes and methodologies, we can find some common elements in their scope [Cernuzzi et al., 2005]
  - both are focusing on what we have to do in the different activities needed to construct a software system
  - however, while the software development process is more centered on the global process including all the stages, their order and time scheduling, the methodology focuses more directly on the specific techniques to be used and artifacts to be produced

- In this sense, we could say that methodologies focus more explicitly on how to perform the activity or tasks in some specific stages of the process, while processes may also cover more general management aspects, e.g., basic questions about who and when, and how much
Example: OO Software Engineering

- **Abstractions:**
  - objects, classes, inheritance, services.

- **Processes:**
  - RUP, OPEN, etc.

- **Methodologies:**
  - object-oriented analysis and design
  - centered around object-oriented abstractions

- **Tools (Modeling Techniques):**
  - UML (standard), E-R, finite state automata, visual languages
Part II

Agent Oriented Software Engineering
Outline

4 Why do we need AOSE?

5 Why agents and MAS?
Why do we need Agent-Oriented Software Engineering?

- Agent-based computing introduces novel abstractions and asks for
  - making clear the set of abstractions
  - adapting methodologies and producing new tools
- Novel, specific agent-oriented software engineering approaches are needed!
What are agents?

There has been some debate on what an agent is, and what could be appropriately called an agent.

Two main viewpoints (centered on different perspectives on autonomy):

- the (strong) Artificial Intelligence viewpoint
  - an agent must be, proactive, intelligent, and it must converse instead of doing client-server computing
- the (weak) Software Engineering Viewpoint
  - an agent is a software component with internal (either reactive or proactive) threads of execution, and that can be engaged in complex and stateful interactions protocols
What are Multiagent Systems?

- Again...
  - the (strong) Artificial Intelligence viewpoint
    - a MAS (multiagent system) is a society of individuals (AI software agents) that interact by exchanging knowledge and by negotiating with each other to achieve either their own interest or some global goal
  - the (weak) Software Engineering Viewpoint
    - a MAS is a software systems made up of multiple independent and encapsulated loci of control (i.e., the agents) interacting with each other in the context of a specific application viewpoint...
SE Viewpoint on Agent-Oriented Computing

- We commit to weak viewpoint because
  - it focuses on the characteristics of agents that have impact on software development
    - concurrency, interaction, multiple loci of control
    - intelligence can be seen as a peculiar form of control independence; conversations as a peculiar form of interaction
  - It is much more general
    - does not exclude the strong AI viewpoint
    - several software systems, even if never conceived as agent-based one, can be indeed characterised in terms of weak multi-agent systems

- Let's better characterise the SE perspective on agents...
SE Implications of Agent Characteristics

- **Autonomy**
  - control encapsulation as a dimension of modularity
  - conceptually simpler to tackle than a single (or multiple inter-dependent) locus of control

- **Situatedness**
  - clear separation of concerns between
    - the active computational parts of the system (the agents)
    - the resources of the environment

- **Interactivity**
  - communication, coordination
  - collaborative or competitive interaction
  - not a single characterising protocol of interaction
  - interaction protocol as an additional SE dimension
SE Implications of Agent Characteristics

- **Sociality**
  - not a single characterising protocol of interaction
  - interaction as an additional SE dimension

- **Openness**
  - controlling self-interested agents, malicious behaviours, and badly programmed agents
  - dynamic re-organisation of software architecture

- **Mobility and Locality**
  - additional dimension of autonomous behaviour
  - improve locality in interactions
MAS Characterisation

Society of Agents (Multiagent Architecture)

High-level Dynamic Interactions between Agents

Interactions with the Environment

Environment
Agent-Oriented Abstractions

- The development of a multi-agent system should fruitfully exploit *abstractions* coherent with the above characterisation:
  - *agents*, autonomous entities, independent loci of control, situated in an environment, interacting with each other.
  - *environment*, the world agents perceive (including resources as well as other agents).
  - *interaction protocols*, as the acts of interactions among agents and between agents and resources of environment.

- In addition, there may be the need of abstracting:
  - the *local context* where an agent lives (e.g., a sub-organisation of agents) to handle mobility & openness.

- Such abstractions translate into concrete entities of the software system.
Agent-Oriented Methodologies

- There is a need for SE methodologies
  - centered around specific agent-oriented abstractions
  - the adoption of OO methodologies would produce mismatches
    - classes, objects, client-servers: little to do with agents!
- Each methodology may introduce further abstractions
  - around which to model software and to organise the software process
    - e.g., roles, organisations, responsibilities, beliefs, desires and intentions...
  - not directly translating into concrete entities of the software system
    - e.g. the concept of role is an aspect of an agent, not an agent
Agent-Oriented Tools

- SE requires tools to
  - represent software
    - e.g., interaction diagrams, E-R diagrams, etc.
  - verify properties
    - e.g., petri nets, formal notations, etc.

- AOSE requires
  - specific agent-oriented tools
    - e.g., UML per se is not suitable to model agent systems and their interactions (object-oriented abstractions not agent-oriented ones)
Outline

4. Why do we need AOSE?

5. Why agents and MAS?
Why Agents and Multi-Agent Systems?

- Other lectures may have already outlined the advantages of (intelligent) agents and of multi-agent systems, and their possible applications
  - autonomy for delegation (do work on our behalf)
  - monitor our environments
  - more efficient interaction and resource management

- We mostly want to show that

agent-based computing, and the abstractions it uses, represent a new and general-purpose software engineering paradigm!

MASs already proved effective in dealing with open, distributed, complex problems that are considered hard challenges for classical software engineering approaches
There is much more to agent-oriented software engineering

- AOSE is not only for “agent systems”
  - most of today’s software systems features are very similar to those of agents and multi-agent systems
  - agent abstractions, methodologies, and AOSE tools suit such software systems
- AOSE is suitable for a wide class of scenarios and applications!
  - agents’ “artificial intelligence” features may be important but are not central
- But of course...
  - AOSE may sometimes appear to be too “high-level” for simple complex systems...
  - there is a gap between the AOSE approach and the available technologies
Agents and Multi-Agent Systems are (virtually) Everywhere

- Examples of components that can be modelled (and observed) in terms of agents:
  - autonomous network processes
  - computing-based sensors
  - PDAs
  - robots

- Example of software systems that can be modelled as multi-agent systems:
  - internet applications
  - P2P systems
  - sensor networks
  - pervasive computing systems
Part III

AOSE Methodologies & Meta-models
In this part we present:

- the meta-model idea
- some AO methodologies:
  - ADELFE [Bernon et al., 2005a], [Capera et al., 2004], [Bernon and Capera, 2008]
  - Gaia [Wooldridge et al., 2000], [Zambonelli et al., 2003], [Cernuzzi et al., 2009]
  - PASSI [Cossentino, 2005], [Cossentino et al., 2008], [Cossentino et al., 2007b]
  - Tropos [Susi et al., 2005], [Bresciani et al., 2004], [Hadar et al., 2010]
  - Prometheus [Padgham and Winikof, 2003], [Padgham and Winikoff, 2005], [DeLoach et al., 2009]
  - SODA [Molesini et al., 2010], [Molesini et al., 2008b], [Molesini et al., 2009a]
  - INGENIAS [Grasia Group, 2009], [Pavòn et al., 2005], [García-Magariño et al., 2009]
Outline

6 Meta-model

7 AOSE Methodologies
- ADELFE
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
Meta-models

**Definition**

Meta-modelling is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for the modelling in a predefined class of problems.

- A meta-model enables checking and verifying the completeness and expressiveness of a methodology by understanding its deep semantics, as well as the relationships among concepts in different languages or methods.
- The process of designing a system consists of instantiating the system meta-model the designers have in their mind in order to fulfill the specific problem requirements [Bernon et al., 2004]
Using Meta-models

- Meta-models are useful for specifying the concepts, rules and relationships used to define a family of related methodologies.

- Although it is possible to describe a methodology without an explicit meta-model, formalising the underpinning ideas of the methodology in question is valuable when checking its consistency or when planning extensions or modifications.

- A good meta-model must address all of the different aspects of methodologies, i.e. the process to follow and the work products to be generated.

- In turn, specifying the work products that must be developed implies defining the basic modelling building blocks from which they are built.

- Meta-models are often used by methodologists to construct or modify methodologies.
Meta-models & Methodologies

- Methodologies are used by software development teams to construct software products in the context of software projects.
- Meta-model, methodology and project constitute, in this approach, three different areas of expertise that, at the same time, correspond to three different levels of abstraction and three different sets of fundamental concepts.
- As the work performed by the development team at the project level is constrained and directed by the methodology in use, the work performed by the methodologist at the methodology level is constrained and directed by the chosen meta-model.
- Traditionally, these relationships between *modelling layers* are seen as instance-of relationships, in which elements in one layer are instances of some element in the layer above.
MAS Meta-model

- MAS meta-models usually include concepts like role, goal, task, plan, communication.
- In the agent world, the meta-model becomes a critical element when trying to create a new methodology because, to date, there are not common denominator.
  - Each methodology has its own concepts and system structure.
Designing a software means instantiating its meta-model
The use of meta-models to underpin object-oriented processes was pioneered in the mid-1990s by the OPEN Consortium [OPEN Working Group, ] leading to the current version of the OPEN Process Framework (OPF).

The Object Management Group (OMG) then issued a request for proposals for what turned into the SPEM (Software Processing Engineering Metamodel) [Object Management Group, 2008]
SPEM (Software Process Engineering Meta-model) [Object Management Group, 2008] is an OMG standard object-oriented meta-model defined as an UML profile and used to describe a concrete software development process or a family of related software development processes.

SPEM is based on the idea that a software development process is a collaboration between active abstract entities called roles which perform operations called activities on concrete and real entities called work products.

Each role interacts or collaborates by exchanging work products and triggering the execution of activities.

The overall goal of a process is to bring a set of work products to a well-defined state.
SPEM level of abstraction

- MetaObject Facility (M3) connected to MOF
- Process Metamodel (M2) connected to UPM, UML
- Process Model (M1) connected to e.g., RUP, SI Method, Open
- Performing process (M0) connected to Process as really enacted on a given project
The goals of SPEM are to:

- support the representation of one specific development process
- support the maintenance of several unrelated processes
- provide process engineers with mechanisms to consistently and effectively manage whole families of related processes promoting process reusability
Standardize representation and manage libraries of reusable Method Content

- Content on agile development
- Content on managing iterative development
- Guidance on serialized java beans

JUnit user guidance
Content on J2EE
Configuration mgmt guidelines

Develop and manage Processes for performing projects

- Process for Custom Application Development with J2EE
- Process for Embedded System Development
- Process for SOA Governance

Process assets patterns
Standard or reference processes
Enactable project plan templates
Corporate guidelines on compliance

Configure a cohesive process framework customized for my project needs

Create project plan templates for Enactment of process in the context of my project
Clear separation between

- **Method Contents** — introduce the concepts to document and manage development processes through natural language description
- **Processes** — defines a process model as a breakdown or decomposition of nested *Activities*, with the related *Roles* and input / output *Work Products*
- **Capability patterns** — reusable best practices for quickly creating new development processes
SPEM: Method Content and Process

Method Framework

Method Content
- Work Product Definition
- Role Definition
- Task Definition
- Category

Process
- Task Use
- Role Use
- Work Product Use
- Activity
- Process
## SPEM Notation

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td><img src="symbol1" alt="Activity Symbol" /></td>
</tr>
<tr>
<td>Category</td>
<td><img src="symbol2" alt="Category Symbol" /></td>
</tr>
<tr>
<td>Composite role and Team</td>
<td><img src="symbol3" alt="Composite role Symbol" /></td>
</tr>
<tr>
<td>Guidance</td>
<td><img src="symbol4" alt="Guidance Symbol" /></td>
</tr>
<tr>
<td>Milestone</td>
<td><img src="symbol5" alt="Milestone Symbol" /></td>
</tr>
<tr>
<td>Process</td>
<td><img src="symbol6" alt="Process Symbol" /></td>
</tr>
<tr>
<td>Process Component</td>
<td><img src="symbol7" alt="Process Component Symbol" /></td>
</tr>
<tr>
<td>Process Pattern</td>
<td><img src="symbol8" alt="Process Pattern Symbol" /></td>
</tr>
<tr>
<td>Role Definition and Use</td>
<td><img src="symbol9" alt="Role Definition Symbol" /></td>
</tr>
<tr>
<td>Task Definition and Use</td>
<td><img src="symbol10" alt="Task Definition Symbol" /></td>
</tr>
<tr>
<td>Tool Definition</td>
<td><img src="symbol11" alt="Tool Definition Symbol" /></td>
</tr>
<tr>
<td>WorkProduct Definition and Use</td>
<td><img src="symbol12" alt="WorkProduct Definition Symbol" /></td>
</tr>
</tbody>
</table>
Outline

6 Meta-model

7 AOSE Methodologies
- ADELFE
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
Outline

6 Meta-model

7 AOSE Methodologies
   • ADELFE
   • Gaia
   • PASSI
   • TROPOS
   • Prometheus
   • SODA
   • INGENIAS
Characteristics of ADELFE

- ADELFE is dedicated to the design of systems that are complex, open and not well-specified (Adaptive Multi-Agent Systems)
- The primary objective of ADELFE method is to cover all the phases of a classical software design
- RUP has been tailored to take into account specificities coming from the design of AMAS
- ADELFE follows the vocabulary of RUP
- Only the requirement, analysis and design work definitions require modifications in order to be adapted to AMAS, other appearing in the RUP remaining the same
- ADELFE is supported by several Tools
Adaptive Multi-Agent Systems Theory

- The openness and dynamics are source of *unexpected* events and an open systems plugged into a dynamic environments has to be able to adapt to these changes, to *self-organise*

- Self-organisation is a means to make the system adapt but also to overcome complexity

- If a system is complex and its algorithm unknown, it is impossible to code its *global function*

- This function has then to emerge at the macro level (system level) from the interaction at the micro level (component level)

- It is sufficient to build a system whose components have cooperative attitude to make it realise an expected function

- Cooperation is the local criterion that enables component to find the right place within the organisation
Adaptive Multi-Agent Systems

- Adaptive Multi-Agent Systems are composed of agents that permanently try to maintain cooperative interactions with other.

- Any *cooperative agent* in AMAS follow a specific lifecycle that consists in:
  - the agent gets perceptions from its environment
  - it autonomously uses them to decide what to do in order to reach its own goal
  - it acts to realise the action on which it has previously decided
The ADELFE Meta-model
The ADELFE Meta-model

- Belongs to no predefined organization (AMAS)
- Has local goal
- Is cooperative
- Detects and removes NCS
The ADELFE Meta-model

An agent owns some skills e.g. specific knowledge that enables it to realize its own partial function.
Aptitudes show the ability of an agent to reason both about knowledge and beliefs it owns.
An agent may possess some characteristics which are its intrinsic or physical properties (for instance, the size of an agent or the number of legs of a robot-like or ant-like agent)
The ADELFE Meta-model

An agent observes some cooperation rules and avoids NCS (Non Cooperative Situations)
The ADELFE Meta-model

An agent has perception of the environment elements

Molesini (UniBo)
AOSE
AOSE 2010
The ADELFE Process
ADELFE: Example

```
perception
  + message ()
  - outcomeAllCell
  - roomInfo: String[*]
  - manages: BookingAgent[*]
  + bookingAgentsBeen ()
interaction
  + getInformation ()
  + inform ()
  + informAboutConstraints ()
  + acceptPartnership ()
  + acceptFreeRoom ()
representation
  - constraint: Constraint[*]
  - bookingStatus: BookingState[*]
  - partnershipStatus: PartnershipStatus[*]
  - partnernshipConstraint: Constraint[*]
  - partner: BookingAgent
skill
  - manageCell ()
  - isProposableAcceptable ()
  - interpretMessage ()
aptitude
  - manageConstraints ()
  - manageReservations ()
  - managePartners ()
  - manageMessages ()
  - isRoomFilling ()
cooperation
  - bookingIncompetence ()
  - bookingConflict ()
  - bookingIsolation ()
  - messageUnproductivity ()
  - partnershipConflict ()
  - partnershipIncompetence ()
action
  - bookRoom ()
  - freeRoom ()
  - signPartnership ()
  - cancelPartnership ()
  - sendMessage ()
```

```
perception
  + message ()
representation
  - bookingAgents: BookingAgent[*]
  - constraint: Constraint[*]
  - represents: Person
  - representPerson: Person
skill
  - interpretMessage ()
aptitude
  - manageBookingAgents ()
  - manageReservations ()
  - manageMessages ()
cooperation
  - incompetence ()
action
  - createBookingAgent ()
  - deleteBookingAgent ()
  - addConstraints ()
  - deleteConstraints ()
  - sendMessage ()
```

```
perception
  + message ()
representation
  - constraint: Constraint[*]
  - represents: Person
skill
  - interpretMessage ()
aptitude
  - manageReservations ()
  - managePartners ()
cooperation
  - incompetence ()
action
  - bookRoom ()
  - freeRoom ()
  - signPartnership ()
  - cancelPartnership ()
  - sendMessage ()
```

```
perception
  + message ()
representation
  - constraint: Constraint[*]
skill
  - interpretMessage ()
aptitude
  - manageReservations ()
cooperation
  - incompetence ()
action
  - bookRoom ()
  - freeRoom ()
```
ADELFE: Example
Outline

6 Meta-model

7 AOSE Methodologies

- ADELFE
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
The Gaia Methodology

- It is the most known AOSE methodology
  - firstly proposed by Jennings and Wooldridge in 1999
  - extended and modified by Zambonelli in 2000
  - final Stable Version in 2003 by Zambonelli, Jennings, Wooldridge
  - many other researchers are working towards further extensions...
The Gaia Methodology

- Starting from the requirements (what one wants a software system to do)
- Guide developers to a well-defined design for the multi-agent system
- Model and dealing with the characteristics of complex and open multi-agent systems
- Easy to implement
The Gaia Methodology

- Exploits organisational abstractions
  - conceive a multi-agent systems as an organisation of individual, each of which playing specific roles in that organisation
  - and interacting accordingly to its role

- Introduces a clear set of abstractions
  - roles, organisational rules, organisational structures
  - useful to understand and model complex and open multi-agent systems

- Abstract from implementation issues
The Gaia Meta-model
The Gaia Meta-model

OrganizationalStructure

Organization
control regime
topology

Service
inputs
outputs
pre-conditions
post-conditions

Environment

Resource
name
description

Action
type

Permission

AgentType

Role

Responsibility

Activity

Communication

Protocol

LivenessProperty

SafetyProperty

Molesini (UniBo)

AOSE 2010
The Gaia Meta-model

- OrganizationalStructure
  - Organization
    - control regime
    - topology
  - Environment
  - Service
    - inputs
    - outputs
    - pre-conditions
    - post-conditions
  - Resource
    - name
    - description
  - Action
    - type
  - Permission
  - Role
    - +member
    - observes
    - plays
    - acts on/interacts with
  - LivenessProperty
  - SafetyProperty
  - Responsibility
  - Activity
  - Protocol
    - name
    - initiator
    - partner
    - inputs
    - outputs
    - description

This diagram illustrates the Gaia Meta-model, which encompasses various meta-models for organizational structures, services, environments, and roles, among others. The relationships and properties are indicated with arrows and annotations, providing a comprehensive view of the model's components and their interactions.
The Gaia Meta-model

OrganizationalStructure

Organization
- control regime
- topology

Service
- inputs
- outputs
- pre-conditions
- post-conditions

Environment

AgentType

Resource
- name
- description

Action
- type

Permission

Role

Responsibility

Activity

Communication

Protocol
- name
- initiator
- partner
- inputs
- outputs
- description

OrganizationRule

SafetyRule

LivenessRule

Service

Resource

Activity

Permission

Role

Responsibility

AgentType

Organization

Environment

Collaborates/Interacts

Observes

Provides

Acts on/Interacts with

Has

+member

Permitted action

0..*
1

1..*
1

0..*
1

0..*
1

+member

0..*
1

+permitted action

1

0..*
1

1

1..*
1

1..*
1

1..*
1

1..*
1

1..*
1

1..*
1
The Gaia Meta-model

OrganizationalStructure

Service
- inputs
- outputs
- pre-conditions
- post-conditions

Environment

Resource
- name
- description

Action
- type

Permission

Role

Activity

Responsibility

Communication

Protocol
- name
- initiator
- partner
- inputs
- outputs
- description

Organization
- control regime
- topology

AgentType

LivenessProperty

SafetyProperty

0..*

1

1

1

0..*

1..*

+member

collaborates

0..*

1

observes

observes

acts on/interacts with

has

1..*

+initiator/participant

1

provides

0..*

1

observes

0..*

1

1..*

0..*

1

1..*

1..*

1..*

0..*

1

1

0..*

1..*
The Gaia Meta-model
The Gaia Meta-model

- OrganizationalStructure
  - control regime
  - topology

- Organization
  - observes

- Service
  - inputs
  - outputs
  - pre-conditions
  - post-conditions

- AgentType
  - +member
  - collaborates

- Resource
  - name
  - description

- Environment
  - 0..*

- Permission
  - *

- Role
  - acts on/interacts with

- Action
  - type

- Responsibility
  - +initiator/participant

- Activity
  - 1..*

- Communication
  - 0..*

- OrganizationRule
  - 0..*

- Protocol
  - name
  - initiator
  - partner
  - inputs
  - outputs
  - description

- LivenessProperty
  - SafetyProperty

- SafetyRule
  - LivenessRule

- Molesini (UniBo)
- AOSE
- AOSE 2010
The Gaia Meta-model
The Gaia Process

Analysis → Architectural Design → Detailed Design
Role Name: ReviewCatcher
Description:
This role is in charge of selecting reviewers and distributing papers among them.

Protocol and Activities:
GetPaper, CheckPaperTopic, CheckRefereeExpertise, CheckRefereeConstraints, AssignPaperReferee,
ReceiveRefereeRefuse, UpdateDBSubmission, UpdateDBReferee

Permissions:
Reads
- paper_submitted
- referee-data
in order to check the topic and authors
in order to check the expertise and constraint (i.e. the referee is one of the authors, or belong to the same organization

Changes
- DB Submission
- DB Referee
assigning a referee to the paper
assigning the paper to the referee incrementing the number of assigned papers

Responsibilities:
Liveness:
ReviewCatcher = (GetPaper, CheckPaperTopic, CheckRefereeExpertise,
CheckRefereeConstraints, AssignPaperReferee, [ReceiveRefereeRefuse | UpdateDBSubmission, UpdateDBReferee])^n

Safety:
∀ paper: number_of_referees ≥ n
Referee ≠ Author
Referee_organization ≠ Author_organization
<table>
<thead>
<tr>
<th>Action</th>
<th>Environment Abstraction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads</td>
<td>Paper Submitted</td>
<td>the Web site receives a paper</td>
</tr>
<tr>
<td>Review</td>
<td>Review Submitted</td>
<td>the Web site receives a review</td>
</tr>
<tr>
<td>Changes</td>
<td>DB Submission</td>
<td>insert in the data base the paper or the review received; one per each track</td>
</tr>
<tr>
<td></td>
<td>DB Reviewer</td>
<td>insert in the data base the personal data of the reviewer, the topic of expertise and the maximum number of papers the referee accepted to review</td>
</tr>
</tbody>
</table>

**Protocol Name:** ReceivePaperAssignement

**Initiator:** ReviewCatcher  
**Partner:** ReviewPartitioner  
**Input:** paperSubmitted

**Description:** The ReviewPartitioner, having checked the area of the paper, assigns the paper to the corresponding ReviewCatcher (the Vice-Chair in charge of that area).

**Output:** The paper is assigned to a specific area and the DB Submission is updated.
Meta-model

AOSE Methodologies
- ADELFE
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
The PASSI Methodology

- PASSI (Process for Agent Societies Specification and Implementation) is a step-by-step requirement-to-code methodology.
- The methodology integrates design models and concepts from both Object Oriented Software Engineering and MAS using UML notation.
- PASSI refers to the most diffuse standards: UML, FIPA, JAVA, Rational Rose.
- PASSI is conceived to be supported by PTK (PASSI Tool Kit) an agent-oriented CASE tool.
The PASSI Methodology

PASSI process supports:

- modelling of requirements is based on use-cases
- ontology that as a central role in the social model
- multiple perspectives: agents are modelled from the social and internal point of view, both structurally and dynamically
- reuse of existing portions of design code; this is performed through a pattern-based approach
- design of real-time systems
- the design process is incremental and iterative

Extends UML with the MAS concepts
The PASSI Meta-model
The PASSI Process
PASSI: Example
PASSI: Example

[Diagram of PASSI process]

Role Name

Agent which plays it

Description

Responsibilities

Molesini (UniBo)
AOSE
AOSE 2010 96 / 237
Outline

6 Meta-model

7 AOSE Methodologies

- ADELFE
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
The Tropos Methodology

- Tropos is an agent-oriented software development methodology founded on two key features
  - (i) the notions of agent, goal, plan and various other knowledge level concepts are fundamental primitives used uniformly throughout the software development process
  - (ii) a crucial role is assigned to requirements analysis and specification when the system-to-be analysed with respect to its intended environment
- Then the developers can capture and analyse the goals of stakeholders
- These goals play a crucial role in defining the requirements for the new system: prescriptive requirements capture the what and the how for the system-to-be
The Tropos Methodology

- Tropos adopts Eric Yu’s i* model which offers actors (agents, roles, or positions), goals, and actor dependencies as primitive concepts for modelling an application during early requirements analysis.
The Tropos Meta-model

Diagram:

- **Actor**
- **Position**
- **Agent**
- **Role**

Relationships:
- **occupy** from **Actor** to **Position**
- **cover** from **Position** to **Role**
- **play** from **Agent** to **Role**

 Cardinalities:
- **1** for **Agent**
- **0..n** for **Role**

The Tropos Meta-model

Diagram:

- Actor
- Position
- Agent
- Role

Relationships:
- Agent plays a Role
- Agent occupies a Position
- Agent covers a Role

Notation:
- 1
- 0..n

Terms:
- occupy
- cover
- play
The Tropos Meta-model
The Tropos Meta-model
The Tropos Meta-model
The Tropos Meta-model

Diagram showing relationships between actors, goals, resources, and plans.
The Tropos Meta-model

Diagram showing relationships between entities such as Actors, Goals, SoftGoals, HardGoals, Resources, Dependencies, and Plans.
Tropos: Example
We introduce the system actor and analyze its dependencies with actors in its environment identifying system’s functional and non-functional requirements.
Late requirements

The goals decomposition, means-end, and contribution analysis are performed on the system’s goals.
Outline

6 Meta-model

7 AOSE Methodologies
- ADELFE
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
The Prometheus Methodology

- Prometheus is a detailed process for specifying, designing, and implementing intelligent agent systems.
- The goal in developing Prometheus is to have a process with defined deliverables which can be taught to industry practitioners and undergraduate students who do not have a background in agents and which they can use to develop intelligent agent systems.
- Prometheus distinguishes itself from other methodologies by supporting the development of intelligent agents:
  - providing *start-to-end* support,
  - having evolved out of practical industrial and pedagogical experience,
  - having been used in both industry and academia, and, above all, in being detailed and complete.
The Prometheus Methodology

- Prometheus is also amenable to tool support and provides scope for cross checking between designs.
- The methodology consists of three phases: system specification, architectural design, and detailed design.
- Although the phases are described in a sequential fashion it is acknowledged that like most Software Engineering methodologies, practice involves revisiting earlier phases as one works out the details.
The Prometheus Overview
There is typically substantial iteration between scenario development and goal hierarchy development until the developer feels that the application is sufficiently described/defined. At this stage goals are grouped into cohesive units and assigned to roles which are intended as relatively small and easily specified chunks of agent
The Prometheus Design Tool – A Conference Management System Case Study

Fig. 7. Goal Overview Diagram
Fig. 8. System Roles Diagram

functionality. The percepts and actions are then also assigned to the roles appropriately to allow the roles to achieve their goals. This is done using the 'System Roles' diagram.
For example, Figure 8 shows that the 'Assignment' role is responsible for the goals to collect preferences (from the reviewers) and assign papers (to the reviewers). To achieve these goals the role needs the input (reviewer info) and reviewer preferences (prefs) and should perform the actions of requesting preferences from reviewers (request prefs) and giving out the paper assignments.

The next stage is the architectural design where we specify the internal composition of the system. The main tasks here are to decide the agent types (as collections of roles) and to define the agent conversations (protocols) that will happen in order to realise the specified goals and scenarios. Decisions regarding grouping of roles into agents are captured in the 'Agent-Role Grouping Diagram'. Figure 9 shows the roles of assigning papers to reviewers (Assignment) and managing the review process (review management) as being part of a Review manager agent. A number of issues must be considered in determining how to group roles into agents, including standard software engineering issues of cohesion and coupling. The relationships of roles to data are also considered in determining role groupings. The Data Coupling and Acquaintance diagrams can assist the designer in visualising these aspects.

Once decisions have been made about how roles are grouped into agents, information can be propagated from the role specifications, to show which percepts and actions are associated with which agents. This information is automatically generated into the 'System Overview Diagram' which, when completed, provides an overview of the internal system architecture. What must be done to complete this overview is to define interactions between the agents (protocols), and to add any shared data. Figure 10 shows the system overview for our conference management system design. Observing the 'Papers manager' agent we can see that it receives papers (percept) from authors and provides an acknowledgment (action) to them. It interacts with the 'Selections manager' agent via the 'selection decision' protocol to be able to send authors...
Outline

6 Meta-model

7 AOSE Methodologies
- ADELFE
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
SODA: Societies in Open and Distributed Agent spaces

SODA ...

- ... is an agent-oriented methodology for the analysis and design of agent-based systems
- ... focuses on inter-agent issues, like the engineering of societies and environment for MAS
- ... adopts agents and artifacts – after the A&A meta-model [Omicini et al., 2006] – as the main building blocks for MAS development
- ... introduces a simple layering principle in order to cope with the complexity of system description
- ... adopts a tabular representation
SODA: Overview

Requirements Analysis

Analysis

References Tables

Requirements Tables

Responsibilities Tables

Domain Tables

Dependencies Tables

Relations Tables

Topologies Tables

Analysis

Design

Architectural Design

Detailed Design

References Tables

Requirements Tables

Responsibilities Tables

Domain Tables

Dependencies Tables

Relations Tables

Topologies Tables

Mapping Tables

Entities Tables

Interaction Tables

Constraints Tables

Topological Tables

Transitions Tables

Agent/Society Design Tables

Environment Design Tables

Interaction Design Tables

Topological Design Tables

Molesini (UniBo)

AOSE

AOSE 2010 113 / 237
The SODA Meta-model
The SODA Meta-model
The SODA Meta-model
The SODA Process

Requirements Analysis

Analysis

Is the problem well specified?

Layering

yes

no

Architectural Design

Is the system well specified?

yes

no

Layering

Detailed Design

Are there problems in the system?

no

The SODA Layering

- new layer?
  - no: Select Layer
  - yes:
    - increases detail: In-zoom
    - increases abstraction: Out-zoom
    - Projection
### SODA: Example

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManageStartUp</td>
<td>creating call for papers and defining the rules of the organisation</td>
</tr>
<tr>
<td>ManageSubmission</td>
<td>managing user registration and paper submissions</td>
</tr>
<tr>
<td>ManagePartitioning</td>
<td>partitioning papers based on the conference structure</td>
</tr>
<tr>
<td>ManageAssignment</td>
<td>managing the assignment process according to the organisation rules</td>
</tr>
<tr>
<td>ManageReview</td>
<td>managing the review process and sending reviews to authors</td>
</tr>
</tbody>
</table>
### SODA: Example

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>management user</td>
<td>managing user information</td>
</tr>
<tr>
<td>management review</td>
<td>managing review information</td>
</tr>
<tr>
<td>management paper</td>
<td>managing paper information</td>
</tr>
<tr>
<td>management assignment</td>
<td>managing assignment information</td>
</tr>
<tr>
<td>management partitioning</td>
<td>managing partitioning information</td>
</tr>
<tr>
<td>management process</td>
<td>managing start-up information</td>
</tr>
<tr>
<td>webSite</td>
<td>web interface of the conference</td>
</tr>
<tr>
<td>Rule</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Deadline-Rule</td>
<td>paper can be sent only if current time precedes the deadline</td>
</tr>
<tr>
<td>User-Rule</td>
<td>get user is possible if the requested user is the requester or the requester is the PC-chair</td>
</tr>
<tr>
<td>Author-Rule</td>
<td>author can access and modify only his public paper information</td>
</tr>
<tr>
<td>Match-Rule</td>
<td>papers can be partitioned according key words</td>
</tr>
<tr>
<td>AutRev-Rule</td>
<td>the PC-member cannot be one of the paper authors</td>
</tr>
<tr>
<td>Review-Rule</td>
<td>the PC-member cannot access private information about his papers</td>
</tr>
</tbody>
</table>
Outline

6 Meta-model

7 AOSE Methodologies
- ADELFE
- Gaia
- PASSI
- TROPOS
- Prometheus
- SODA
- INGENIAS
The INGENIAS Methodology

- The INGENIAS methodology covers the analysis and design of MAS and it is intended for general use.
- The methodology is supported by the INGENIAS Development Kit (IDK), which contains a graphical editor for MAS specifications.
- Besides, the INGENIAS Agent Framework (IAF), integrated in the IDK, has been proposed for enabling a full model-driven development and transforming automatically specifications into code in the Java Agent Development Framework.
- The software development process proposed by the methodology is based on RUP [Kruchten, 2003].
- The methodology distributes the tasks of analysis and design in three consecutive phases: Inception, Elaboration and Construction.
- Each phase may have several iterations (where iteration means a complete cycle of development).
The INGENIAS Methodology

- INGENIAS follows the Model Driven Development (MDD), so it is based on the definition of a set of meta-models that describe the elements that form a MAS from several viewpoints.
- The specification of a MAS is structured in five viewpoints:
  1. the definition, control and management of each agent mental state
  2. the agent interactions
  3. the MAS organisation
  4. the environment
  5. the tasks and goals assigned to each agent
The INGENIAS Meta-model

- **Organization**
  - Define its tasks and relates them with the goals using...
  - Is defined in

- **Agent**
  - Express its goals
  - Defines its behavior using...
  - Perceives from
  - Belongs to
  - It is defined in

- **Tasks and Goals Metamodel**
- **Organization Metamodel**
- **Agent Metamodel**
- **Interaction Metamodel**
- **Environment Metamodel**
The INGENIAS Process

Inception  Elaboration  Construction
INGENIAS: Example
INGENIAS: Example
Part IV

Methodologies Documentation & Situational Method Engineering
Outline

8 Methodologies Documentation

9 Situational Method Engineering

10 Situational Method Engineering in AOSE
   - SPEM and AOSE processes
   - Method Fragment Representation
   - PRODE: PROcess DEsign for design processes

11 Result Evaluation
As said before, in the software engineering field, there is common agreement in that there is not a unique methodology or process, which fits all the application domains.

This means that the methodology or process must be adapted to the particular characteristics of the domain for which the new software is developed.

There are two major ways for adapting methodologies:

- **Tailoring**: particularization or customization of a pre-existing processes.
- **Situational Method Engineering (SME)**: process is assembled from pre-existent components, called fragments, according to user’s needs (see next section).

The research on SME has become crucial in AOSE since a variety of special-purpose agent-oriented methodologies have been defined in the past years to discipline and support the MASs development.
Each of the AO methodologies proposed until now presents specific meta-model, notation, and process. These characteristics are fundamental for a correct comprehension of a methodology and should be documented in a proper way for supporting the creation of new ad-hoc AOSE methodologies.

SME is strictly related to the documentation of the existing methodologies. The successfully construction of a new process is based on the correct integration of different fragments that should be well formalised.

The methodologies’ documentation should be done in a standard way in order to facilitate: the user’s comprehension and the adoption of automatic tools able to interpret the fragment documentation.
Methodologies Documentation

- The IEEE FIPA Design Process Documentation and Fragmentation (DPDF) working group [DPDF, 2009] has recently proposed a template for documenting AO methodologies.

- This template
  - has been conceived without considering any particular process or methodology → all processes can be documented using it
  - is neutral regarding the MAS meta-model and/or the modelling notation adopted in describing the process
  - has a simple structure resembling a tree, so documentation is made in a natural and progressive way:
    - addressing in first place the general description and meta-model definition which constitute the root elements of the process
    - detailing in a second step the branches which are the phases
  - is easy to use for a software engineer as it relies on few previous assumptions
  - suggests as notation the use of the OMG’s standard SPEM [Object Management Group, 2008] with few extensions [Seidita et al., 2008] (see next section)
Template structure

1. Introduction
   1.1. The (process name) Process lifecycle
   1.2. The (process name) Metamodel
      1.2.1. Definition of MAS metamodel elements
2. Phases of the (process name) Process
   2.1. (First) Phase
      2.1.1. Process roles
      2.1.2. Activity Details
      2.1.3. Work Products
   2.2. (Second) Phase
      2.2.1. Process roles
      2.2.2. Activity Details
      2.2.3. Work Products
   ... (further phases) ...
3. Work Product Dependencies
Methodologies Documentation: benefits

- The template helps
  - in easily catching/understanding/studying the methodology: it seems evident the facility of studying another methodology when the new one uses an approach we already know
  - in reusing parts
  - in identifying similarities and differences in the methodologies
Methodologies Documentation: examples

- Examples...
Outline

8 Methodologies Documentation

9 Situational Method Engineering

10 Situational Method Engineering in AOSE
   - SPEM and AOSE processes
   - Method Fragment Representation
   - PRODE: PROcess DEsign for design processes

11 Result Evaluation
Methodologies

As for software development, individual methodologies are often created with specific purposes in mind [Henderson-Sellers, 2005]
  - particular domains
  - particular segments of the lifecycle

Users often make the assumption that a methodology in not in fact constrained but, rather, is universally applicable

This can easily lead to methodology failure, and to the total rejection of methodological thinking by software development organisation

The creation of a single universally applicable methodology is an unattainable goal

We should ask ourselves how could we create a methodological environment in which the various demands of different software developers might be satisfied altogether
Method Engineering [Brinkkemper, 1996]

Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems.
All the concepts and ideas used in the Method Engineering are also applicable to our definitions of methodology and method. Method Engineering tries to model methodological processes and products by isolating conceptual method fragments. These fragments act as methodological "building blocks".
Method & Methodology

All the concepts and ideas used in the Method Engineering are also applicable to our definitions of methodology and method.

Method Engineering tries to model methodological processes and products by isolating conceptual *method fragments*.

These fragments act as methodological “building blocks”.
Method Engineering: Motivations

Adaptability – to specific projects, companies, needs & new development settings

Reuse – of best practices, theories & tools
Method Engineering: Motivations

- Adaptability – to specific projects, companies, needs & new development settings
- Reuse – of best practices, theories & tools
Method Engineering: Concerns

- Similarly as software engineering is concerned with all aspects of software production, so is method engineering dealing with all engineering activities related to methods, techniques and tools.
- The term method engineering is not new but it was already introduced in mechanical engineering to describe the construction of working methods in factories.
- Even if the work of Brinkkemper is dated, most of the open research issues he presented are not well addressed yet:
  - meta-modelling techniques
  - tool interoperability
  - situational method(ology)
  - comparative review of method(ologie)s and tools
A situational method is an information systems development method tuned to the situation of the project at hand.

Engineering a situational method requires standardised building blocks and guide-lines, so-called meta-methods, to assemble these building blocks.

Critical to the support of engineering situational methods is the provision of *standardised method building blocks* that are stored and retrievable from a so-called method base.

Furthermore, a *configuration process* should be set up that guides the assembly of these building blocks into a situational method.

The building blocks, called *method fragments*, are defined as *coherent pieces of information system development methods*.
Configuration Process [Brinkkemper, 1996]
Situational Method Engineering I

- Every project is different, so it is essential in the method configuration process to characterise the project according to a list of contingency factors.
- This project characterisation is an input to the selection process, where method fragments from the method base are retrieved.
- Experienced method engineers may also work the other way round, i.e. start with the selection of method fragments and validate this choice against the project characterisation.
- The unrelated method fragments are then assembled into a situational method.
- As the consistency and completeness of the method may require additional method fragments, the selection and validation processes could be repeated.
- Finally, the situational method is forwarded to the systems developers in the project.
Situational Method Engineering II

- As the project may not be definitely clear at the start, a further elaboration of the situational method can be performed during the course of the project.
- Similarly drastic changes in the project require to change the situational method by the removal of inappropriate fragments followed by the insertion of suitable ones.
And Now?

Two important questions
  ▶ how to represent method fragments?
  ▶ how to assembly method fragments?

To assemble method fragments into a meaningful method, we need a procedure and representation to model method fragments and impose some constraints or rules on method assembly processes.
Outline

8 Methodologies Documentation

9 Situational Method Engineering

10 Situational Method Engineering in AOSE
   - SPERM and AOSE processes
   - Method Fragment Representation
   - PRODE: PROcess DEsign for design processes

11 Result Evaluation
The development methodology is built by the developer by assembling pieces of the process (method fragments) from a method base.

The method base is composed of contributions coming from existing methodologies and other novel and specifically conceived fragments.

This is the approach used within the FIPA Technical Committee Methodology (2003-2005).
The normal agent development process

- Design Methodology
- System Designer
- CASE Tools
- System Specifications
- Agents
- Solve
- Problem

Diagram:

1. Design Methodology is adopted by a System Designer.
2. The System Designer designs the agents to solve a problem.
3. The agents are specified using CASE tools.
4. The system specifications are produced.
Situational Method Engineering

Perceives

Method Engineer -> Defines -> Design Methodology

Uses

Fragments Repository

CAME Tools

Instantiates

System Designer -> Designs -> Agents

Solve

Problem

Uses

CASE Tools

Produce

System Specifications

Specify

Molesini (UniBo)

AOSE

AOSE 2010

149 / 237
Adopting Situational Method Engineering

What do I need?

- a collection of method fragments
- some guidelines about how to assemble fragments
- a CAME (Computer Aided Method Engineering) tool
- an evaluation framework (is my new methodology really good?)
CAME Tool

- This tool is based on the method meta-model and it is responsible for method fragment specification, i.e. their product and process parts definition.
- Method fragment specification can be done “from scratch”, by assembly or by modification.
- In the first case product and process models of the fragments are defined by instantiating the method meta-model used by the tool.
- In the second case fragments are assembled in order to satisfy some specific situation.
- In the third case fragments are obtained by modification of other fragments in order to better satisfy the method goal.
- Depending to the method meta-model, the CAME tool should offer graphical modelling facilities and special features.
The new process production
The new process production

All methodologies are expressed in a standard notation (we adopt SPEM by OMG)

Existing Methodologies

New Method Fragments

Method Fragments Extraction

Method Base

MAS Meta-Model

CAME tool

CASE tool

Specific Methodology

Specific problem

MAS running on agent platforms

Deployment

MAS Model

All methodologies are expressed in a standard notation (we adopt SPEM by OMG)
The new process production

Existing Methodologies

Method Fragments Extraction

Fragments are identified and described according to the previous discussed definition

New Method Fragments

Method Base

CAME tool

CASE tool

Specific Methodology

Specific problem

MAS running on agent platforms

Molesini (UniBo)

AOSE 2010 152 / 237
The new process production

New fragments are defined if necessary

Existing Methodologies

Method Fragments Extraction

New Method Fragments

Method Base

MAS Meta-Model

CAME tool

CASE tool

MAS running on agent platforms

Deployment

Specific Methodology

Specific problem

Molesini (UniBo)
The new process production

A method fragments repository is composed with all existing fragments

Existing Methodologies

Method Fragments Extraction

New Method Fragments

Method Base

MAS Meta-Model

CAME tool

CASE tool

Specific Methodology

Specific problem

MAS Model

MAS running on agent platforms

Deployment
The new process production

Existing Methodologies

New Method Fragments

Method Fragments Extraction

Method Base

MAS Meta-Model

MAS running on agent platforms

CAME tool

CASE tool

The desired MAS-Meta-Model is composed according to problem specific needs (for instance including or not self-organizing agents)

Deployment

MAS Model

Specific Methodology

Specific problem

The desired MAS-Meta-Model is composed according to problem specific needs (for instance including or not self-organizing agents)
The new process production

A CAME (Computer Aided Method Engineering) tool assists in the selection of fragments and composition of design process.
The new process production

A new and problem specific methodology is built

Existing Methodologies

New Method Fragments

Method Fragments Extraction

Method Base

CAME tool

CASE tool

MAS Model

MAS running on agent platforms

Deployment

Specific Methodology

Specific problem
The new process production

A CASE (Computer Aided Software Engineering) tool is used to effectively design the multi-agent system.

Existing Methodologies

Method Fragments Extraction

New Method Fragments

Method Base

MAS Meta-Model

CAME tool

CASE tool

Specific Methodology

Specific problem

MAS running on agent platforms

Deployment

MAS Model

A CASE (Computer Aided Software Engineering) tool is used to effectively design the multi-agent system.
The new process production

- Existing Methodologies
- New Method Fragments

Method Fragments Extraction

Method Base

MAS Meta-Model

CAME tool

CASE tool

Specific Methodology

Specific problem

The multi-agent system has been coded, tested and is ready to be deployed

MAS running on agent platforms

The multi-agent system has been coded, tested and is ready to be deployed

Deployment

MAS Model

The multi-agent system has been coded, tested and is ready to be deployed

Molesini (UniBo)
So we need:

- A meta-model for modelling and design an AOSE process
- A specific description of an AOSE fragment
- A way for assembly AOSE fragments
The process description

- Three are the main elements of a design process
  - activity
  - process role
  - work product
- AOSE processes are also affected by
  - MAS Meta-model (MMM) Element
- SPEM does not support the MMM Elements
Extending SPEM Specifications [Seidita et al., 2009a]

- MMM is the starting point for the construction of a new design process
  - each part (one or more elements) of this meta-model can be instantiated in one (or more) fragment(s)
- Each fragment refers to one (or more) MMM element(s)
  - refers = instantiates/relates/quotes/refines
- The MMM element is the constituent part of a Work Product
- The MMM is not part of the SPEM meta-model
  - it is the element which leads us in modifying and extending SPEM diagram
Extending SPEM Specifications [Seidita et al., 2009a]

- The need for establishing which is the real action a process role performs on a MMM element when he is carrying out a specific activity

- The set of actions:
  - **define** – it is performed when a MMM element is introduced for the first time and its features are defined in a portion of process (hence in a fragment)
  - **relate** – when a relationship is created (defined) among two or more MMM elements previously defined in another portion of process
  - **quote** – a MMM element or a relationship is quoted in a specific work product
  - **refine** – a MMM element attribute is defined or a value is identified for it

- We also find useful to specify the work product kind by referring to an explicit set of WP kinds
Proposed icons
Method fragment meta-model

- The FIPA Methodology Technical Committee in 2003-2005 proposed the following definition of method fragment [Cossentino et al., 2007a]
What is a Method Fragment

A fragment is a portion of the development process, composed as follows:

- **A portion of process** (what is to be done, in what order), defined with a **SPEM** diagram
- **One or more deliverables** (like (A)UML/UML diagrams, text documents and so on)
- **Some preconditions** (they are a kind of constraint because it is not possible to start the process specified in the fragment without the required input data or without verifying the required guard condition)
- **A list of concepts** (related to the MAS meta-model) to be defined (designed) or refined during the specified process fragment
- **Guideline(s)** that illustrates how to apply the fragment and best practices related to that
- **A glossary of terms** used in the fragment (in order to avoid misunderstandings if the fragment is reused in a context that is different from the original one)
- **Other information** (composition guidelines, platform to be used, application area and dependency relationships useful to assemble fragments) complete this definition.
The PRODE approach for Agent-Oriented Method Engineering [Seidita et al., 2009b]
The PRODE Process Representation
PRODE divided in three main areas of research

1) A collection of process fragments
2) Guidelines for fragment assembling
PRODE divided in three main areas of research

3) A CAPE (Computer Aided Process Engineering) tool
The CAPE tool: Metameth

- **Metameth** is an (open-source) agent-oriented tool we built to support our experiments in methodologies composition and their application in real projects.

- **Metameth** is:
  - a CAPE tool: since it supports the definition of the design process life-cycle and the positioning of the different method fragments in the intended place
  - a CAME tool: since it allows the definition of different method fragments
  - a CASE tool: since it supports a distributed design process, it offers several (by now UML) graphical editors and an expert system for verifying the resulting system
Metameth tool architecture

PROCESS DEFINITION

MAS Metamodel Editor (PROTEGE')

Rule Editor

WorkFlow Editor (JAVE)

XPDL

PROCESS EXECUTION

Expert System (JESS)

Knowledge Base

WorkFlow Engine (SHARK)

Activity Agent (JADE)

UML Editor (ECLIPSE)

Method Engineer

Designer

Molesini (UniBo)

AOSE

AOSE 2010 164 / 237
Outline

8 Methodologies Documentation

9 Situational Method Engineering

10 Situational Method Engineering in AOSE
   - SPEM and AOSE processes
   - Method Fragment Representation
   - PRODE: PROcess DEsign for design processes

11 Result Evaluation
Results Evaluation: an open problem?

Results Evaluation is crucial also in process improvement/reengineering.
AO Design Process Evaluation

Details on AO processes evaluation [Numi Tran and Low, 2005]

Structure of the evaluation framework

<table>
<thead>
<tr>
<th></th>
<th>GAIA</th>
<th>TROPOS</th>
<th>MAS-COMMONAKADS</th>
<th>PROMETHEUS</th>
<th>PASSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development lifecycle</td>
<td>Iterative within each phase but sequential between phases</td>
<td>Iterative and incremental</td>
<td>Cyclic risk-driven process</td>
<td>Iterative across all phases (except for coding and deployment)</td>
<td></td>
</tr>
<tr>
<td>Coverage of the lifecycle</td>
<td>Analysis and Design</td>
<td>Analysis and Design</td>
<td>Analysis and Design</td>
<td>Analysis and Design</td>
<td>Analysis, Design and Implementation</td>
</tr>
<tr>
<td>Development perspective</td>
<td>Top-down</td>
<td>Top-down</td>
<td>Hybrid</td>
<td>Bottom-up</td>
<td>Bottom-up</td>
</tr>
<tr>
<td>Application domain</td>
<td>Independent (business process management, GIS, traffic simulation)</td>
<td>Independent (e-business systems, knowledge management, health IS)</td>
<td>Independent (Flight reservation, automatic control)</td>
<td>Independent (Robotics manufacturing, online bookstore)</td>
<td>Independent (distributed robotics applications, online bookstore)</td>
</tr>
<tr>
<td>Size of MAS</td>
<td>$\leq$ 100 agent classes</td>
<td>Not specified</td>
<td>Not specified, but possibly any size</td>
<td>Any size</td>
<td>Not specified</td>
</tr>
<tr>
<td>Agent nature</td>
<td>Heterogeneous</td>
<td>BDI-like agents</td>
<td>Heterogeneous</td>
<td>BDI-like agents</td>
<td>Heterogeneous</td>
</tr>
<tr>
<td>Support for verification and validation</td>
<td>No</td>
<td>Yes</td>
<td>Mentioned but no explicit steps/guidelines provided</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Details on AO processes evaluation

- From:

- Evaluation is based on:
  - concepts and properties (autonomy, proactiveness, . . .)
  - notations and modeling techniques (accessibility, expressiveness)
  - process (development context, Lifecycle coverage)
  - pragmatics (required expertise, scalability, . . .)
Details on AO processes evaluation

- From:

- Based on a questionnaire
- Reused and extended in AL3-AOSE TFG3

Details on AO processes evaluation

- The Capability Maturity Model Integration (CMMI) [SEI, 2006a]
  - the overall goal of CMMI is to provide a framework that can share consistent process improvement best practices and approaches, but can be flexible enough to address the rapidly changing needs of the community

- SCAMPI (Standard CMMI Assessment Method for Process Improvement)[SEI, 2006b] it is a schema for process evaluation in five steps: activation, diagnosis, definition, action, learning
Details on AO processes evaluation: CMMI discrete levels

- Levels are used in CMMI to describe an evolutionary path recommended for an organisation that wants to improve the processes.
- The maturity level of an organization provides a way to predict an organization’s performance in a given discipline or set of disciplines.
- A maturity level is a defined evolutionary plateau for organizational process improvement.
### Details on AO processes evaluation: CMMI discrete levels

<table>
<thead>
<tr>
<th>Maturity Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Initial</td>
<td>processes are usually ad hoc and chaotic</td>
</tr>
<tr>
<td>2-Managed</td>
<td>processes are planned and executed in accordance with policy</td>
</tr>
<tr>
<td>3-Defined</td>
<td>processes are well characterized and understood, and are described in standards, procedures, tools, and methods</td>
</tr>
<tr>
<td>4-Quantitatively managed</td>
<td>the organization and projects establish quantitative objectives for quality and process performance and use them as criteria in managing processes</td>
</tr>
<tr>
<td>5-Optimizing</td>
<td>an organization continually improves its processes based on a quantitative understanding of the common causes of variation inherent in processes</td>
</tr>
</tbody>
</table>

AOSE processes are (at most) at level 3!! *(only a few of them)*
Open issues

- SME is perceived to be a difficult discipline
  - this is only partially true. All new design processes creator performed (usually in a disordered way) the steps proposed and studied by SME
  - a greater diffusion of AO-SME can have positive effects on the development of new AO design processes (specifically in new areas like self-org)

- Major problems with AO-SME
  - AO processes deals with MAS metamodels and they are an open issue in the agent community
  - lack of standards (ISO specification vs FIPA proposal)
    - lack of standard repository of fragments
  - lack of stable (commercial quality) CAPE/CAME tools
  - design process evaluation is still an open issue in both AO and OO software engineering
Part V

Research directions and conclusions
Outline

12 Research directions and visions

13 Conclusions

14 List of interesting papers
Mainstream AOSE Researches

- **Methodology**
  - dozens of methodologies proposed so far
  - mostly “pencil and papers” exercises with no confrontation with real world problems...

- **Meta-methodologies**
  - interesting and worth to be explored, but...
  - these would require much more research coordination and more feedback from real-world experiences

- **Models & Notations**
  - of great help to clarify agent-oriented abstractions
  - no specific standard still exists

- **Infrastructures**
  - very interesting models but...
  - (the lack of) a pure agent-oriented language slows down the implementation phase
Is This Enough?

- Let’s ask ourselves a simple basic question:
  - what does it mean engineering a MAS?
  - what is the actual subject of the engineering work?

- What is a MAS in a world of:
  - world-wide social and computational networks
  - pervasive computing environments
  - sensor networks and embedded computing

- There is not a single answer:
  - it depends on the observation level

- In the physical world and in micro-electronics
  - [Zambonelli and Omicini, 2004]
    - micro level of observation: dominated by quantum phenomena (and to be studied/engineered accordingly)
    - macro level of observation: dominated by classical physics
    - meso level of observation: quantum and classical phenomena both appears (and have to be taken into account)
AOSE Observation Levels

- **Micro scale**
  - small-medium-size MASs
  - control over each component (limited complexity single stakeholder)
  - this is the (only) focus of mainstream AOSE

- **Macro scale**
  - very large scale distributed MASs
  - no control over single components (decentralisation, multiple stakeholders)
  - the kingdom of “self-organization” people

- **Meso scale**
  - micro scale components deployed in a macro scale scenario
  - my own system influence and is influenced by the whole

- **Very rarely a fully fledged study can be limited to a single level of observation**
  - most MASs (even small scale) are open
  - deployed in some sort of macro scale system
  - dynamically evolving together with the system
Micro-Level Challenges (1)

- Assessing AOSE Advantages
  - AO has clear advantages. What about AOSE?
  - methodologies, methodologies, methodologies... ;-(
    - qualitative work
  - we need to show that AOSE
    - helps saving money and human resource
    - leads to higher quality software products
    - quantitative comparison of AOSE vs. non-AOSE complex software development

- Pay Attention to the Software Process
  - most methodologies assume a “waterfall” model
    - either implicitly or explicitly
    - with no counterpart in industrial software development
  - Need for:
    - agile processes
    - agent-specific flexible processes
  - Can meta-models be of help in that direction?
Micro-Level Challenges (2)

- Agent-specific notations
  - AUML is ok to spread acceptance but...
    - is it really suited for MASs?
    - and for complex systems in general?
    - even the mainstream SE community doubts about that...
  - do more suitable notations exist?
    - agent-specific ones to be invented
    - other non-UML approaches
  - Cf. Sturm et al. 2003: OPM/MAS
  - AML by Whitestein [Cervenka et al., 2005]
  - A proposal of unified notation by L. Padgham, M. Winikoff, S. De Loach, M. Cossentino [Padgham et al., 2009]

- Intelligence engineering
  - selling AI has always been difficult
    - lack of engineering flavor...
  - agents can help with that
    - embodied, modular, intelligence
    - observable rationality
  - our role should be that of:
    - exploiting scientific results from the AI-oriented MAS community
    - turn them into usable engineered products
Macro-Level Challenges (1)

- The macro level deals with complex collective behaviour in large scale MASs
  - some say this is not AOSE...
    - scientific activity
    - observing and reproducing biology
  - but it must become an engineering activity
    - challenging indeed

- Universality in MASs
  - can general laws underlying the behaviour of complex MASs be identified?
    - as they are starting being identified in the “complex systems” research community
    - phase transitions, edges of chaos, etc.
  - letting us study and engineer complex MAS
    - abstracting from the specific characteristics of agents (from ants to rational BDI agents)
    - abstracting from the specific content of their interactions
Macro-Level Challenges (2)

- **Measuring Complex MASs**
  - how can we characterise the behavior of large-scale MASs?
    - when we cannot characterise the behavior of single components
  - macro-level measures must be identified
    - to concisely express properties of a system
    - Cf. Entropy, Macro-properties of complex network, etc
  - and tools must be provided to actually measure systems
  - but measuring must be finalised

- **Controlling Complex MASs**
  - given a measurable property of a MAS
  - software engineers must be able to direct the evolution of a system, i.e., to tune the value of the measurable property
    - in a fully decentralised way
    - and with the possibility of enforcing control over a limited portion of the MAS
  - software engineering will become strictly related to control systems engineering

- **Emergent behaviors, physics, biology, etc**
  - Cf. The activity of the “SELF ORGANIZATION” Agentlink group
Meso-Level Challenges (1)

- It is a problem of deployment
  - engineering issues related to deployment of a MAS (typically engineered at a micro level of observation).
  - ... into a large scale system (to be studied at a macro-level of observation)

- Impact Analysis
  - how will my system behave when it will deployed in an existing open possibly large scale networked system?
  - how I will influence the existing system?
  - micro-scale aspects:
    - tolerance to unpredictable environmental dynamics on my system
    - internal handlings
  - macro-scale aspect:
    - can my “small” MAS change the overall behaviour of the global system?
    - “butterfly effect”?
Meso-Level Challenges (2)

- Identifying the Boundaries
  - how can I clearly identify what is part of my system and what is not?
  - I should identify
    - potential inter-agent and environmental interactions
    - shape the environment (i.e., via agentification)
    - engineer the interactions across the environment
  - in sum: engineering the boundaries of the system

- Trust
  - I can (provably) trust a “small” system of rational agents
  - I can (probabilistically) trust a very large-scale MASs
  - what I can actually say about the small system deployed in the large-scale one
    - how can I measure the “degree of trust”?

- Infrastructures for Open Systems
  - are configurable context-dependent coordination infrastructure the correct answer?
  - are normative approaches the correct ones?
    - we know what we gain but we do not know what we lose
  - Cf. Incentives in social and P2P networks
Research directions and visions: conclusions

- There is not a single AOSE
  - depends on the scale of observation…

- The micro scale
  - overwhelmed by research
  - often neglecting very basic questions…

- The macro scale
  - some would say this is not AOSE
  - but it must become indeed…

- The meso scale
  - fascinating…
  - very difficult to be tackled with engineering approaches…

- What else?
  - there is so much to engineer around…
  - emotional agents, mixed human-agent organisations, interactions with the physical world…
Outline

12 Research directions and visions

13 Conclusions

14 List of interesting papers
Reflections

In this lecture we have spoken about Software Engineering and Agent Oriented Software Engineering.

Some reflections are necessary:

- what are the aspects related to Engineering?
- what are the aspects related to Software Engineering?
- what are the aspects related to the paradigms adopted?
What are the Aspects Related to Engineering?

- Following a clear and disciplined development process
- Adopting a design methodology
- Creating an appropriate (mathematical?) model of a problem that allows to analyse it
- Testing potential solutions
- Evaluating the different design choices and choosing the solution that best meets requirements
- Using: prototypes, scale models, simulations, destructive tests, nondestructive tests, and stress tests
What are the Aspects Related to Software Engineering?

- Customization to the specific *kind of product*: Software
  - specific software development processes tied to the *software lifecycle*
  - specific methodologies
  - specific kinds of model tied to the concept of *software product*
  - testing potential solutions
  - using of specific techniques for: prototypes, scale models, simulations, tests, and stress tests
What are the Aspects Related to the paradigm?

- The building blocks for creating the models
- The level of *thinking / abstraction*
- Functions, objects, agents lead to different ways of *thinking* both the problems and the solutions
  - the paradigm adopted leads to different levels of *model complexity*: complicated problems are well captured by objects and agents, while functions could lead to have very very complex models for representing the problem
  - in the same way the models of the solution are heavily influenced by the paradigm
Outline

12 Research directions and visions

13 Conclusions

14 List of interesting papers
Introduction to Agents and Multiagent Systems

(this is a very PARTIAL list, lots of very interesting refs are not reported here)

- D. Chess, C. Harrison, A. Kershenbaum, *Mobile Agents: are They a Good Idea?* [Chess et al., 1996]
- N. R. Jennings, *An Agent-Based Approach for Building Complex Software System* [Jennings, 2001]
Introduction to AOSE

- N.R. Jennings, *On Agent-Based Software Engineering* [Jennings, 2000]
Relevant References on AOSE

(this is a very PARTIAL list, lots of very interesting refs are not reported here)

- **Books on AOSE**
  - M. Luck, R. Ashri, M. D’Inverno, *Agent-Based Software Development* [Luck et al., 2004]
  - B. Henderson-Sellers and P. Giorgini, *Agent-Oriented Methodologies* [Henderson-Sellers and Giorgini, 2005]

- **Surveys and other papers on AOSE**
  - F. Zambonelli, A. Omicini, *Challenges and Research Directions in Agent-Oriented Software Engineering* [Zambonelli and Omicini, 2004],
  - C. Bernon, M. Cossentino, J. Pavòn *An Overview of Current Trends in European AOSE Research* [Bernon et al., 2005c],
  - C. Bernon, M. Cossentino, J. Pavòn, *Agent-oriented software engineering* [BERNON et al., 2006]
References on others AO Methodologies

- ASPECS: [Cossentino et al., 2010]
- MaSE: [DeLoach et al., 2001], [DeLoach and Kumar, 2005]
- O-MaSE: [DeLoach, 2008], [DeLoach, 2006]
- MESSAGE: [Caire et al., 2002], [Caire et al., 2004], [Garijo et al., 2005]
- CommonKDS [Iglesias and Garijo, 2005]
- AOR [Taveret and Wagner, 2005]
- OPM-MAS [Sturm et al., 2003]
References on Meta-models

- M. Cossentino, S. Gaglio, L. Sabatucci, V. Seidita, *The PASSI and Agile PASSI MAS Meta-models Compared with a Unifying Proposal* [Cossentino et al., 2005]
- A. Molesini, E. Denti, A. Omicini, *MAS Meta-models on Test: UML vs. OPM in the SODA Case Study* [Molesini et al., 2005]
- A. Molesini, E. Denti, A. Omicini, *From AO Methodologies to MAS Infrastructures: The SODA Case Study* [Molesini et al., 2008a]
- INGENIAS Home Page [Grasia Group, 2009]
References on Processes

- B. Henderson-Sellers, C. Gonzalez-Perez. *A comparison of four process metamodels and the creation of a new generic standard* [Henderson-Sellers and Gonzalez-Perez, 2005]
- A. Molesini, N. Nardini, E. Denti, A. Omicini, *SPEM on Test: the SODA Case Study* [Nardini et al., 2008],
- A. Molesini, N. Nardini, E. Denti, A. Omicini, *Advancing Object-Oriented Standards Toward Agent-Oriented Methodologies: SPEM 2.0 on SODA* [Molesini et al., 2008b],
References on Method Engineering

- S. Brinkkemper, *Method engineering: engineering the information systems development methods and tools* [Brinkkemper, 1996]
- B. Henderson-Sellers, J. Debenham, *Towards open methodological support for agent-oriented systems development* [Henderson-Sellers and Debenham, 2003]
- M. Cossentino, S. Gaglio, A. Garro, V. Seidita. *Method Fragments for agent design methodologies: from standardization to research* [Cossentino et al., 2007a]
References on MAS Infrastructures

Communication (FIPA-based) Infrastructures

- S. Poslad, P. Buckle, and R. Hadingham, *The FIPA-OS Agent Platform: Open Source for Open Standard* [Poslad et al., ]
- JACK Intelligent Agents [Busetta et al., ]

Coordination Infrastructures

- G. Cabri, L. Leonardi, F. Zambonelli, *Engineering Mobile Agent Applications via Context-Dependent Coordination* [Cabri et al., 2002]
- M. Viroli, M. Casadei, A. Omicini, *A Framework for Modelling and Implementing Self-Organising Coordination* [Viroli et al., 2009]
- A. Ricci, M. Piunti, M. Viroli, A. Omicini, *Environment Programming in CArtAgO* [Ricci et al., 2009]
Open Research Directions & Visions

- V. Parunak, S. Bruekner, *Entropy and Self-Organization in Agent Systems* [Van Dyke Parunak and Brueckner, 2001]
- M. Mamei, F. Zambonelli, L. Leonardi, *A Physically Grounded Approach to Coordinate Movements in a Team* [Leonardi et al., 2002]


Bibliography II


Bibliography IV

Tropos: An agent-oriented software development methodology.
*Autonomous Agent and Multi-Agent Systems*, 3:203–236.

Method engineering: engineering of information systems development methods and tools.

Meta-modelling based assembly techniques for situational method engineering.


*Engineering Applications of Artificial Intelligence, 18*(2):205–222.

Bibliography VIII


Bibliography XI


Engineering organization-based multiagent systems.

Developing a multiagent conference management system using the O-MaSE process framework.
8th International Workshop (AOSE 2007), Honolulu, HI, USA, May 14, 2007, Revised Selected Papers.

Multi-agent systems engineering: An overview and case study.
Using three aose toolkits to develop a sample design. 

DPDF (2009).
IEEE FIPA Design Process Documentation and Fragmentation 
Homepage. 
http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/.

Software process: a roadmap. 
Bibliography XV


An empirical study of requirements model understanding: Use case vs. tropos models.

Creating a comprehensive agent-oriented methodology: Using method engineering and the OPEN metamodel.
In [Henderson-Sellers and Giorgini, 2005], chapter XIII, pages 236–397.

*In Procs. First International Conference on Agent-Based Technologies and Systems*, pages 14–24, Canada.


Bibliography XVIII

A Survey of Agent-Oriented Methodologies.

Agent-oriented methodology MAS-CommonKADS.
In [Henderson-Sellers and Giorgini, 2005], chapter III, pages 46–78.

On agent-based software engineering.
*Artificial Intelligence, 117(2):277–296.*

An agent-based approach for building complex software systems.
*Communication ACM, 44(4):35–41.*


From AO methodologies to MAS infrastructures: The SODA case study.
In Artikis, A., O’Hare, G., Stathis, K., and Vouros, G., editors, 
Engineering Societies in the Agents World VIII, volume 4995 of LNCS, 
pages 300–317. Springer.
8th International Workshop (ESAW’07), 22–24 October 2007, Athens, 
Greece. Revised Selected Papers.


Bibliography XXV

http://www.omg.org/spec/SPEM/2.0/PDF.


OPEN Working Group.
Open home page.


Bibliography XXX

An overview of agent-oriented programming.
pages 271–290.

evaluation of information modeling methods – a review.
in HICSS ’98: Proceedings of the Thirty-First Annual Hawaii
International Conference on System Sciences-Volume 5, page 314,
Washington, DC, USA. IEEE Computer Society.

Addison-Wesley.


Bibliography XXXIII


Agent Oriented Software Engineering

Ambra Molesini

Alma Mater Studiorum – Università di Bologna (Italy)
ambra.molesini@unibo.it

Bologna, Italy, 28th June 2010