Documentation and Fragmentation of Agent Oriented Methodologies and Processes

Ambra Molesini\textsuperscript{1} \quad Massimo Cossentino\textsuperscript{2}

\textsuperscript{1}Alma Mater Studiorum – Università di Bologna (Italy)
ambra.molesini@unibo.it

\textsuperscript{2}Italian National Research Council - ICAR Institute - Palermo (Italy)
cossentino@pa.icar.cnr.it

12th European Agent Systems Summer School
Saint-Etienne, France
August 2010
Part I

General Concepts
Outline

1. Software Engineering, Processes and Methodologies

2. Why do we need AOSE?
## What is Software Engineering?

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

Development Process [Cernuzzi et al., 2005]

- **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).
Software Process [Fuggetta, 2000]

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.
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.
There is no an ideal process

[Sommerville, 2007]
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
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
Outline

1. Software Engineering, Processes and Methodologies

2. Why do we need AOSE?
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...
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...
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 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, organizations, 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)
Part II

Meta-model
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.
Outline

3 MAS Meta-model

4 Process Meta-model
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 in the agent oriented context, 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 ADELFE Meta-model

The diagram illustrates the ADELFE Meta-model, which includes concepts such as Environment, Element, Classifier, AMAS, Representation, Cooperative Agent, Incomprehension, Ambiguity, Incompetence, Unproductiveness, Concurrency, Conflict, and Uselessness.
The Gaia Meta-model
The PASSI Meta-model
The Tropos Meta-model

Diagram:

- Actor
- Position
- Agent
- Role

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

Cardinalities:
- 1
- 0..n
The Tropos Meta-model
The SODA Meta-model
Outline

3 MAS Meta-model

4 Process Meta-model
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]

- Here, for space reason we present only SPEM
SPEM

- 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

M0: Performing process
- Process as really enacted on a given project

M1: Process Model
- e.g., RUP, SI Method, Open

M2: Process Metamodel
- UPM, UML

M3: MetaObject Facility
- MOF

SPEM: Process Metamodel
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
A software development process is seen as a collaboration between abstract active entities called **process roles** that perform operations called **activities** on concrete, tangible entities called **work products**.
Roles, Activities & WorkProducts

- A software development process is seen as a collaboration between abstract active entities called **process roles** that perform operations called **activities** on concrete, tangible entities called **work products**.

An Activity defines basic units of work within a Process as well as a Process itself.
Roles, Activities & WorkProducts

- A software development process is seen as a collaboration between abstract active entities called *process roles* that perform operations called *activities* on concrete, tangible entities called *work products*.

![Diagram showing a Role Use relationship and process parameters and activities.](image)
Roles, Activities & WorkProducts

- A software development process is seen as a collaboration between abstract active entities called **process roles** that perform operations called **activities** on concrete, tangible entities called **work products**
## SPEM Notation

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>![Activity Symbol]</td>
</tr>
<tr>
<td>Category</td>
<td>![Category Symbol]</td>
</tr>
<tr>
<td>Composite role and Team</td>
<td>![Composite Role Symbol]</td>
</tr>
<tr>
<td>Guidance</td>
<td>![Guidance Symbol]</td>
</tr>
<tr>
<td>Milestone</td>
<td>![Milestone Symbol]</td>
</tr>
<tr>
<td>Process</td>
<td>![Process Symbol]</td>
</tr>
<tr>
<td>Process Component</td>
<td>![Process Component Symbol]</td>
</tr>
<tr>
<td>Process Pattern</td>
<td>![Process Pattern Symbol]</td>
</tr>
<tr>
<td>Role Definition and Use</td>
<td>![Role Definition Symbol]</td>
</tr>
<tr>
<td>Task Definition and Use</td>
<td>![Task Definition Symbol]</td>
</tr>
<tr>
<td>Tool Definition</td>
<td>![Tool Definition Symbol]</td>
</tr>
<tr>
<td>WorkProduct Definition and Use</td>
<td>![WorkProduct Definition Symbol]</td>
</tr>
</tbody>
</table>
SP EM: Activity Details Diagram

From OMG SPEM 2.0 Specifications
SPEM: Work Product Dependency Diagram

From OMG SPEM 2.0 Specifications
From OMG SPEM 2.0 Specifications
Part III

Process Documentation
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., 2009a]
1. Introduction
   1.1. The (process name) Process lifecycle
   1.2. The (process name) Metamodel
      1.2.1. Definition of MAS metamodel elements
   1.3. Guidelines and Techniques
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

- Examples can be found in:
  http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs/
Part IV

Situational Method Engineering
Outline

5 Method Engineering in traditional SE

6 Method Engineering in AOSE
- SPEM and AOSE processes
- Method Fragment Representation
- PRODE: PROcess DEsign for design processes
  - Fragment collection
  - Guidelines for Fragment Assembly
  - Supporting Tools
- Method Fragment extraction and Repository creation
- Result Evaluation
Method Engineering

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

- 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.

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]

Diagram:
- **Project environment**
  - project factors
- **Characterisation of project**
  - characterisation
  - validation
- **Selection of method fragments**
  - selected method fragments
  - requests for new method fragments
- **Assembly of method fragments**
  - situational method
  - requests for adaptations
- **Project performance**
- **Methods Administration**
  - methods
tools
  - method fragment additions/updates
  - method fragments
- **Method Base**
  - experience accumulation
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

5 Method Engineering in traditional SE

6 Method Engineering in AOSE
   - SPEM and AOSE processes
   - Method Fragment Representation
   - PRODE: PROcess DEsign for design processes
     - Fragment collection
     - Guidelines for Fragment Assembly
     - Supporting Tools
   - Method Fragment extraction and Repository creation
   - Result Evaluation
Agent Oriented Situational Method Engineering

- 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 same approach is currently under study by the IEEE FIPA Design Process Documentation and Fragmentation Working Group.
Agent-Oriented Situational Method Engineering

Perceives

Method Engineer

Defines

Design Methodology

Is adopted by

System Designer

Designs

Agents

Solve

Problem

Uses

Fragments Repository

Uses

CAME Tools

Instantiates

CASE Tools

Uses

Produce

System Specifications

Molesini/Cossentino (UniBo/ICAR-CNR)
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?)

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
Outline

5 Method Engineering in traditional SE

6 Method Engineering in AOSE
   - SPEM and AOSE processes
   - Method Fragment Representation
   - PRODE: PROcess DEsign for design processes
     - Fragment collection
     - Guidelines for Fragment Assembly
     - Supporting Tools
   - Method Fragment extraction and Repository creation
   - Result Evaluation
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., 2009b]

- MMM may be seen as 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
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
Extending SPEM Specifications [Seidita et al., 2009b]

- It is also useful to specify the work product kind by referring to an explicit set of WP kinds. The resulting extension to SPEM is depicted below.

![Diagram of SPEM specifications extension](image)
Proposed icons
The dependency diagram
Example: PASSI process activity diagram
Outline

5 Method Engineering in traditional SE

6 Method Engineering in AOSE
- SPEM and AOSE processes
- Method Fragment Representation
  - PRODE: PROcess DEsign for design processes
    - Fragment collection
    - Guidelines for Fragment Assembly
    - Supporting Tools
- Method Fragment extraction and Repository creation
- Result Evaluation
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.
A standardization activity is ongoing within IEEE FIPA about the definition of Method Fragment (I prefer *Process Fragment*).

The Design Process Documentation and Fragmentation Working Group is responsible for that.

Website: [http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/](http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/)
Outline

Method Engineering in traditional SE

Method Engineering in AOSE
- SPEM and AOSE processes
- Method Fragment Representation
  - PRODE: PROcess DEsign for design processes
    - Fragment collection
    - Guidelines for Fragment Assembly
    - Supporting Tools
- Method Fragment extraction and Repository creation
- Result Evaluation
The Prode approach for Agent-Oriented Method Engineering [Seidita et al., 2010]
The PRODE Process Representation
The PRODE Process Representation

[Diagram of PRODE Process Representation with nodes and relationships, including:
- Process
- Development Context
- Process Life-cycle
- Problem Type
- Orders
- MAS Model
- Notation
- Design tools
- Activity
- Work Product
- MAS Meta-Model
- MAS Meta-Model Relationships
- MAS Meta-Model Entity
- Specific_for
- Adopts
- Designs
- Refers_to
- Supported_by
- Composed_of
- Instance_of
- Instance_of
- Is_Responsible_for
- Structured by
- Involves
- Supports
- Adopts
- Edits
- Produces
- Performs]
PRODE proposes an Extension to the FIPA Proposed Method Fragment Definition

- A method Fragment can be explored from four points of view [Cossentino et al., 2007b]:
  - Process
    - the process related aspect of the fragment: workflow, activity and work product
  - Storing
    - it concerns with the storage of the fragment in the method base and its retrieval
  - Reuse
    - it concerns with the reuse feature of the fragment and lists the elements helpful in reusing the fragment during the composition of a new design process
  - Implementation
    - the implementation of the main elements of the process view

- Method fragment construction is Work Product oriented, a method fragment must deliver a product.
Process Analysis and Design in PRODE

Process Analysis

Core Metamodel Creation

Process Requirements Analysis

Process Lifecycle Definition

Process Design

Metamodel Elements Prioritization

Reusable Fragments Selection

New Fragments Creation

Fragments Assembly

Process Completed?

Metamodel Extension

False

True

New Design Process Enactment

Process Deployment

Process Evaluation

System Design
### Example: PRODE Analysis

<table>
<thead>
<tr>
<th>ASPECS Process Requirement</th>
<th>Strategy</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development of very large MASs for hierarchically decomposable problems</td>
<td>Adoption of holonic decomposition of problems</td>
<td></td>
</tr>
<tr>
<td>Reuse of experiences done with PASSI</td>
<td>Support for functional requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Early identification of agents on the basis of requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transformational approach</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An ontology should be used to model agent’s knowledge</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FIPA-compliance at least at the communication level</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input of the process: text scenarios</td>
<td></td>
</tr>
</tbody>
</table>

- A detailed description of ASPECS in [Cossentino et al., 2010]
Process Analysis and Design in PRODE
Example: Core meta-model creation

<table>
<thead>
<tr>
<th>ASPECS Process Requirement</th>
<th>Strategy</th>
<th>Consequence MMME from PASSI</th>
<th>Consequence MMME from CRIO</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development of very large MASs for hierarchically decomposable problems</td>
<td>Adoption of holonic decomposition of problems</td>
<td>Capacity, Organization, Role, Interaction, Holon</td>
<td>Organizations, not agents should be the center of the process</td>
<td></td>
</tr>
<tr>
<td>Reuse of experiences done with PASSI</td>
<td>Support for functional requirements</td>
<td>Scenario, (Functional) Requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Early identification of agents on the basis of requirements</td>
<td>Link agent-requirement</td>
<td></td>
<td>Agents should be replaced by organizations</td>
<td></td>
</tr>
<tr>
<td>Transformational approach</td>
<td>An ontology should be used to model agent’s knowledge</td>
<td>Ontology (including Concepts, Actions, Predicates)</td>
<td>3 domains in the MMM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FIPA-compliance at least at the communication level</td>
<td>Communication, Message, Interaction Protocol, Ontology, Role</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input of the process: text scenarios</td>
<td></td>
<td>Text Scenario is an input of the process</td>
<td></td>
</tr>
</tbody>
</table>
ASPECS is a design process for building holonic multi-agent systems recently developed at UTBM.
Process Analysis and Design in PRODE

Process Analysis

Core Metamodel Creation

Process Requirements Analysis

Process Lifecycle Definition

Process Design

Metamodel Elements Prioritization

Reusable Fragments Selection

New Fragments Creation

Fragments Assembly

Process Completed?

Metamodel Extension False True

New Design Process Enactment

Process Deployment

Process Evaluation

System Design
What is prioritization??

- The problem we face is:
  - What are the first fragments we should introduce in the new process?
The algorithm

Main issues:

- we assume each process fragment instantiates, relates, refines or quotes MAS Meta-Model Elements (MMMEs)
- we created an algorithm for assigning a priority to the realisation of some MMMEs:
  - elements that are “leaves” of the meta-model graph are realised at first
  - other elements follow according to the number of their relationships
- The output is a priority list of fragments
1. Select a metamodel domain (consider the resulting metamodel as a graph with nodes (MMMEs) and edges (relationships))

2. Define List elements1 as a list of MMMEs that can be defined by reusing fragments from the repository, and the associated priority p: List elements1 (MMME, p), p=1;

3. Define List elements2 as a list of MMMEs that cannot be defined by reusing fragments from the repository;

4. Define List elements3 as a list of elements that are not in the core MMM;

5. While the core MMM is not empty
   a) Select the leaves Li (i=1, . . . , n) that: (i) can be instantiated by fragments of the repository and (ii) have less relationships with other elements
      1. Insert Li (i=1, . . . , n) in List elements1;
      2. Remove elements Li (i=1, . . . , n) from the core MMM;
      3. p = p+1;

6. While the core MMM is not empty
   a) Select the leaves Li (i=1, . . . , m) that can not be instantiated by fragments of the repository;
      1. Insert Li (i=1, . . . , m) in List elements2;
      2. Remove Li (i=1, . . . , m) from the core MMM;
7. For each element \( E_{1_i} \) of List_elements1 select an instantiating fragment from the repository (verify the correspondence among fragment rationale and the process requirements/strategies)
   a) If one fragment corresponds to process requirements and strategies then:
      I. insert the fragment in the new process composition diagram
      II. analyze inputs \( I_i \) (\( i=0, \ldots, n \)) and outputs \( O_j \) (\( j=0, \ldots, m \)) of the fragment
         A. If some \( I_i \) or \( O_j \) does not belong to the core MMM then add it to List_elements3; mark the fragment as “To be modified”
         B. remove \( E_{1_i} \) from List_elements1;
      III. For each element \( E_{2_i} \) in List_elements2 analyze if there is a similarity with the elements defined in this fragment
         A. if yes delete \( E_{2_i} \) from List_elements2 and \( I_i/O_i \) from List_elements3
   b) else (if no fragment correspond to requirements and strategies) then
      I. remove \( E_{1_i} \) from List_elements1 and insert it in List_elements2
8. For each $E_{2i}$ (i=0..m) in List_elements2
   a) Define a new fragment for instantiating $E_{2i}$
   b) Insert the fragment in the new process composition diagram
   c) Remove $E_{2i}$ from List_elements2
9. For each $E_{3i}$ (i=0..m) in List_elements3
   a) Introduce elements $E_{3i}$ (i=0..q) from List_elements3 in the core MMM
   b) Repeat from 2. (consider only the new elements)
10. If the process is not completed (i.e. not all design activities from requirements elicitation to coding, testing and deployment have been defined)
    a) Repeat from 1.
Process Analysis and Design in PRODE

[Diagram showing the process analysis and design workflow in PRODE]
Example: the first two fragments in Building the ASPECS Process

- Priority list for MMME that can be instantiated by fragments included in the repository:
  - 1) Capacity, 2) Funct. Requirement, Ontology Elements (concept, predicate, action), Scenario, 3) Role . . .

- Priority list for MMME that can \textbf{NOT} be instantiated by fragments included in the repository:
  - 1) Organization, 2) Interaction, . . .
Process Analysis and Design in PRODE
Example: Aspecs process component diagram
Meta-model Extension

- The Core MAS Metamodel is the starting point for selecting the right fragments from the repository and for assembling them in the new process

- MAS Metamodel extensions come from:
  - the need of incorporating MMMEs referred in selected fragments
  - new process requirements
  - not all design activities from requirements elicitation to coding, testing and deployment have been defined

- Three different situations may arise:
  - different MAS meta-models contribute to the new one with parts that are totally disjointed
  - different MAS meta-models contribute to the new one with parts that overlap and...

  ★ ... overlapping elements have the same definitions bounded to elements with different names or on the contrary

  ★ ... overlapping elements have the same name but different definitions
Metameth

- **Metameth**\(^1\) 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

- **MAS Metamodel Editor (PROTEGE*)**
- **Rule Editor**
- **WorkFlow Editor (JAVE)**
- **XPDL**
- **Expert System (JESS)**
- **Knowledge Base**
- **WorkFlow Engine (SHARK)**
- **Activity Agent (JADE)**
- **UML Editor (ECLIPSE)**

**PROCESS DEFINITION**

**PROCESS EXECUTION**

**Method Engineer**

**Designer**
Supporting design activities

The operations that can be supported by a tool during the design process:

- GUI Action – the tool interacts with the user (using a GUI) in order to support him in some operations
- WP Composition – the tool creates/updates a work product on the basis of the already introduced design information
- Rule Check – semantic and syntactic check of the work product (warning, alerting and suggestions)

Metameth is composed of a society of agents interacting with users:

- a controller agent – responsible for the execution of process
- a community of Activity agents – interacting with designer
- a ProcessModel agent – is responsible of managing the design information
- an editor agent – manages the diagram editor
The rules

- The Process Model agent is responsible of the activation of Jess rules
- Classification according to five categories:

  - Validation
  - Semantic interpretation
  - Auto-composition
  - Update
  - Import

Rule Check

WP Composition
The expert system

- The Metameth expert system is based on JESS
- Rules are expressed in first order logic
- Ontology is designed using Protegè
- Services offered by the expert system:
  - syntax checks: it verifies the abidance to modelling language rules
  - semantic checks: it verifies the abidance to the MAS meta-model (e.g. a role cannot aggregate another one)
  - semantic understanding of diagrams: elements of notations are mapped to their corresponding MAS meta-model element (a use-case is mapped to a requirement)
  - automatic composition of diagrams: some diagrams can be partially composed by accessing information of previous design phases
The Metameth GUI

- Metameth includes several tools (some are taken from the open source community). Among them:
  - a workflow editor used to specify the process and an engine to execute that: JaWe (Java Workflow Editor), Shark
  - a UML modeling tool (IBM Rational System Developer)
  - (already cited) Jess for realizing the expert system

\[\text{An open source tool made by Enhydra: } \text{http://www.together.at/prod/workflow/twe}\]
Outline

5 Method Engineering in traditional SE

6 Method Engineering in AOSE
- SPEM and AOSE processes
- Method Fragment Representation
- PRODE: PROcess DEsign for design processes
  - Fragment collection
  - Guidelines for Fragment Assembly
  - Supporting Tools
- Method Fragment extraction and Repository creation
- Result Evaluation
Method fragment extraction

- The repository is a data base where method fragments are stored in terms of (usually text) documents.
- Fragments extraction is Work Product- and MMM Element-oriented.
- A fragment is identified as a portion of process that produces a significant work product (a diagram or other kind of WP).
  - Fragments can also be composed: Phase fragment, Composed fragment, Atomic fragment.
The categorisation [Seidita et al., 2006]

- The aim is to unify different elements (from different approaches) under a unique definition
  - a set of common phases of software engineering design processes
  - the principal process role performing these phases
  - a set of work product kind

- The repository allows the classification of fragments according to a set of categories based on the most important meta-model elements
  - Phase
  - Process Role
  - Work Product
  - MMM Element

- All the processes we studied were created by different research groups and deal with different design philosophies
- Different processes have significant differences in names and definitions of the design process elements
  - sixteen different process roles
  - seventeen phases
  - several work products and MAS Meta-model elements
Phases

- Any kind of design process can be decomposed in phases
- High level of abstraction for phases resulting form the studied processes
- Some of them are specific for agent based design process

- Requirements
- Analysis
- Design
- Implementation
- Testing
- Deployment
- Coding
Process Roles

- Identification of an high level process role for each phase
- Detailing process roles basing on studied processes

- System Analyst
- Domain Analyst
- User
- Agent Analyst
- Agent Designer
- User Interface Designer
- Programmer
- Test Designer
- Test Developer
Taxonomy: Work product

- **Work Product Kind**
  - **Graphical**
    - Behavioural
  - **Textual**
    - Structural
    - Structured
    - Free
  - **Composite**
The need for a taxonomy

- Three kinds of MAS Meta-model elements
  - problem domain → all aspects of users problem description including environment representation
  - agency Domain → agent based concepts useful to define a solution
  - solution Domain → the structure of the code solution
A new version of the repository is under development. It will be available soon at: http://www.pa.icar.cnr.it/passi/FragmentRepository/index.html
Outline

5 Method Engineering in traditional SE

6 Method Engineering in AOSE
   - SPEM and AOSE processes
   - Method Fragment Representation
   - PRODE: PROcess DEsign for design processes
     - Fragment collection
     - Guidelines for Fragment Assembly
     - Supporting Tools
   - Method Fragment extraction and Repository creation
   - Result Evaluation
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-COMMOKADS</th>
<th>PROMETHEUS</th>
<th>PASSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development lifecycle</td>
<td>Iterative within each phase but</td>
<td>Iterative and incremental</td>
<td>Cyclic risk-driven process</td>
<td>Iterative across all phases</td>
<td>Iterative across and within all</td>
</tr>
<tr>
<td></td>
<td>sequential between phases</td>
<td></td>
<td></td>
<td></td>
<td>phases (except for coding and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>deployment)</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</td>
<td>Analysis, Design and Implementation</td>
</tr>
<tr>
<td></td>
<td>Top-down</td>
<td>Hybrid</td>
<td>Bottom-up</td>
<td>Design</td>
<td></td>
</tr>
<tr>
<td>Development perspective</td>
<td>Independent (business process</td>
<td>Independent (e-business systems,</td>
<td>Independent (Flight reservation,</td>
<td>Independent (holonic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>management, GIS, traffic</td>
<td>knowledge management, health IS)</td>
<td>automatic control)</td>
<td>manufacturing, online bookstore)</td>
<td></td>
</tr>
<tr>
<td>Application domain</td>
<td>In 100 agent classes</td>
<td>Not specified</td>
<td>Not specified, but possibly any</td>
<td>Any size</td>
<td></td>
</tr>
<tr>
<td>Size of MAS</td>
<td></td>
<td></td>
<td>size</td>
<td></td>
<td></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</td>
<td>No</td>
<td>Yes</td>
<td>Mentioned but no explicit steps/</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>and validation</td>
<td></td>
<td></td>
<td>guidelines provided</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Molesini/Cossentino (UniBo/ICAR-CNR)
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\(^a\)

\(^a\) See AL3 AOSE TFG 1-3 Final Report at: http://www.pa.icar.cnr.it/cossentino/al3tf3/

<table>
<thead>
<tr>
<th>Concept/Property</th>
<th>Adelie</th>
<th>Gaia</th>
<th>Ingenias</th>
<th>OPF</th>
<th>PASSI</th>
<th>Prometheus</th>
<th>TROPOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomy</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H/H/M</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>Mental attitudes</td>
<td>L</td>
<td>N</td>
<td>H</td>
<td>H</td>
<td>L/L/M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Proactivity</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H/M/H</td>
<td>H</td>
<td>N</td>
</tr>
<tr>
<td>Reactiveness</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H/H</td>
<td>H</td>
<td>N</td>
</tr>
<tr>
<td>Concurrency</td>
<td>H</td>
<td>M</td>
<td>H</td>
<td>L</td>
<td>H/H/M</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>Teamwork and roles</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H/M/H</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>Cooperation model</td>
<td>AMAS th. Teamwork</td>
<td>ALL</td>
<td>ALL</td>
<td>Task del./Teamwork</td>
<td>none</td>
<td>Negotiation/Task del.</td>
<td></td>
</tr>
<tr>
<td>Protocols support</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H/M/H</td>
<td>H</td>
<td>N</td>
</tr>
<tr>
<td>Communication modes</td>
<td>ALL</td>
<td>Async meas.</td>
<td>ALL</td>
<td>ALL</td>
<td>Direct</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Communication language</td>
<td>ALL</td>
<td>ACL like</td>
<td>ALL</td>
<td>ALL</td>
<td>Speech acts</td>
<td>messages</td>
<td></td>
</tr>
<tr>
<td>Situatedness</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H/M/M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Environment type</td>
<td>All episodic</td>
<td>Dynamic Continuous</td>
<td>All discrete</td>
<td>ALL</td>
<td>ALL</td>
<td>ALL</td>
<td>Inacc., Non episodic, Dynam.</td>
</tr>
</tbody>
</table>
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.
<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
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
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?

- in the next slides a few papers will be listed. They include a list of AOSE survey that report other points of view on this discipline.
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 about 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 Design Methodologies

- Adelfe: [Bernon et al., 2005a]
- ASPECS: [Cossentino et al., 2010]
- Gaia: [Wooldridge et al., 2000], Gaia2: [Zambonelli et al., 2003]
- Ingenias: [Pavòn et al., 2005]
- MaSE: [DeLoach et al., 2001], O-MaSE: [DeLoach, 2008], [DeLoach, 2006]
- PASSI: [Cossentino, 2005], Agile PASSI: [Chella et al., 2006], PASSIM: [Cossentino et al., 2008], GoalPASSI: [Cossentino et al., 2007c]
- SODA: [Molesini et al., 2009a], [Molesini et al., 2009c], [Molesini et al., 2009b]
- Tropos: [Bresciani et al., 2004]
- Prometheus: [Padgham and Winikof, 2003], [Padgham and Winikoff, 2004]
- MESSAGE: [Caire et al., 2002], [Caire et al., 2004], [Garijo et al., 2005]
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 [Grasina 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]
Bibliography

Developing multi-agent systems with a fipa-compliant agent framework.


Engineering adaptive multi-agent systems: the adelfe methodology.

A study of some multi-agent meta-models.
*Agent-Oriented Software Engineering V: 5th International* . . .

A study of some multi-agent meta-models.

An overview of current trends in european aose research.

Agent-oriented software engineering.


Agile passi: An agile process for designing agents.
*Journal of Computer Systems Science and Engineering.*

Mobile agents: Are they a good idea?

Multiagent system engineering: The coordination viewpoint.

From requirements to code with the PASSI methodology.
In [Henderson-Sellers and Giorgini, 2005], chapter IV, pages 79–106.

*International Journal on Agent Oriented Software Engineering (IJAOSE).*

Method fragments for agent design methodologies: from standardisation to research.

Method fragments for agent design methodologies: from standardisation to research.
The passi and agile passi mas meta-models compared with a unifying proposal.

Adapting passi to support a goal oriented approach: a situational method engineering experiment.
In Proc. of the Fifth European workshop on Multi-Agent Systems (EUMAS’07).

A Holonic Metamodel for Agent-Oriented Analysis and Design.
LECTURE NOTES IN COMPUTER SCIENCE, 4659:237.

Aspecs: an agent-oriented software process for engineering complex systems.

Programming multi-agent systems.
Knowledge Engineering Review.

Multiagent Systems Engineering.

Engineering organization-based multiagent systems.

Developing a multiagent conference management system using the O-MaSE process framework.
volume 4951 of LNCS, pages 168–181. Springer.
8th International Workshop (AOSE 2007), Honolulu, HI, USA, May 14, 2007, Revised Selected Papers.
DPDF (2009).
IEEE FIPA Design Process Documentation and Fragmentation Homepage.
http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/.

Software process: a roadmap.

The MESSAGE methodology for agent-oriented analysis and design.
In [Henderson-Sellers and Giorgini, 2005], chapter VIII, pages 203–235.

Fundamental of Software Engineering.

Methodologies and Software Engineering for Agent Systems, chapter A survey of agent-oriented software engineering research.
Kluwer.

Ingenias meta model.

Towards open methodological support for agent-oriented system development.
In Procs. First International Conference on Agent-Based Technologies and Systems, pages 14–24, Canada.

Agent Oriented Methodologies.
Idea Group Publishing, Hershey, PA, USA.
A comparison of four process metamodels and the creation of a new generic standard.

A Survey of Agent-Oriented Methodologies.

On agent-based software engineering.

An agent-based approach for building complex software systems.

Autonomous agents for business process management.

*Agent-Based Software Development.*
Artech House.

PhD thesis, Dottorato in Ingegneria Elettronica, Informatica e delle Telecomunicazioni, Bologna, Italy.

MAS meta-models on test: UML vs. OPM in the SODA case study.
From AO methodologies to MAS infrastructures: The SODA case study.
8th International Workshop (ESAW’07), 22–24 October 2007, Athens, Greece. Revised Selected Papers.

Agent-based conference management: A case study in SODA.
International Journal of Agent-Oriented Software Engineering.

An agent-oriented software engineering approach to home intelligence.

RBAC-MAS & SODA: Experimenting RBAC in AOSE.

Advancing object-oriented standards toward agent-oriented methodologies: SPEM 2.0 on SODA.

Situated process engineering for integrating processes from methodologies to infrastructures.
SPEM on test: the SODA case study.
Special Track on Software Engineering.

The knowledge level.
Artificial Intelligence, 18(1):87–127.

Comparison of ten agent-oriented methodologies.
In [Henderson-Sellers and Giorgini, 2005], chapter XII, pages 341–367.

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

OPEN Working Group.
Open home page.

Prometheus: A methodology for developing intelligent agents.

Developing intelligent agent systems: a practical guide.
John Wiley and Sons.
“go to the ant”: Engineering principles from natural agent systems.

The INGENIAS methodology and tools.
In [Henderson-Sellers and Giorgini, 2005], chapter IX, pages 236–276.

Poslad, S., Buckle, P., and Hadingham, R.
The fipa-os agent platform: Open source for open standard.

Environment programming in CArTAgO.

SEI (2006a).
CMMI for development version 1.2.

SEI (2006b).

A repository of fragments for agent systems design.
In *Proc. Of the Workshop on Objects and Agents (WOA06)*, pages 130–137.

Using and extending the spem specifications to represent agent oriented methodologies.


Why interaction is more powerful than algorithms.

Reasoning about rational agents.
MIT Press, Cambridge, MA, USA.

Intelligent agents: Theory and practice.

The gaia methodology for agent-oriented analysis and design.

Software engineering with agents: Pitfalls and pratfalls.
IEEE Internet Computing, 3(3):20–27.

Organizational abstractions for the analysis and design of multi-agent system.
In First international workshop, AOSE 2000 on Agent-oriented software engineering, pages 235–251, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Developing multiagent systems: The Gaia methodology.
ACM Transactions on Software Engineering and Methodology (TOSEM), 12(3):317–370.

Challenges and research directions in agent-oriented software engineering.
Special Issue: Challenges for Agent-Based Computing.
……. THANKS!!!
Documentation and Fragmentation of Agent Oriented Methodologies and Processes

Ambra Molesini\textsuperscript{1} \hspace{1cm} Massimo Cossentino\textsuperscript{2}

\textsuperscript{1}Alma Mater Studiorum – Università di Bologna (Italy)
ambra.molesini@unibo.it

\textsuperscript{2}Italian National Research Council - ICAR Institute - Palermo (Italy)
cossentino@pa.icar.cnr.it

12th European Agent Systems Summer School
Saint-Etienne, France
August 2010