Documentation and Fragmentation of Agent Oriented Methodologies and Processes

Ambra Molesini[1] Massimo Cossentino[2]

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

[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

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

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

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

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.

The Ideal Software Process

The Ideal Software Process?

There is no an ideal process

[Sommerville, 2007]

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

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*

Outline

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

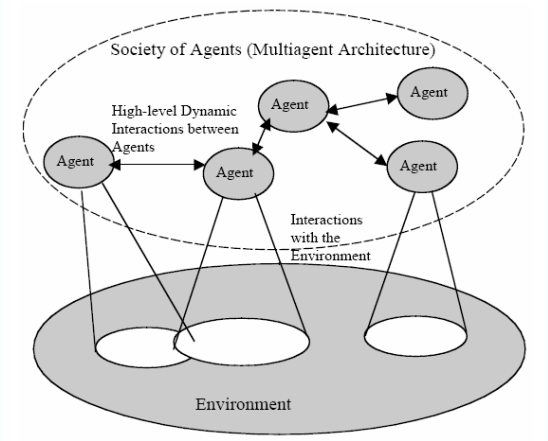
What are Multiagent Systems?

- 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...

MAS Characterisation



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 & opennes
- 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

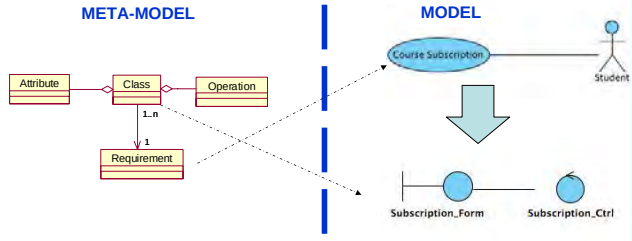
Outline

MAS Meta-model

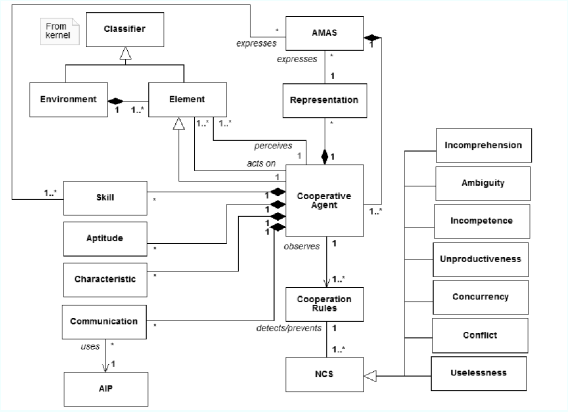
- MAS meta-models usually include concepts like role, goal, task, plan, communication
- In the agent world the meta-model becomes a critical element when trying to create a new methodology because in the agent oriented context, to date, there are not common denominator
  - each methodology has its own concepts and system structure

Software Design: the role of system meta-model

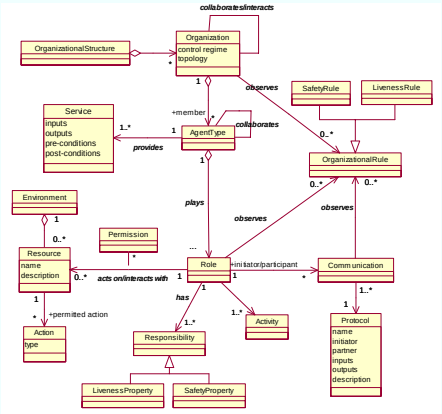
- Designing a software means instantiating its meta-model



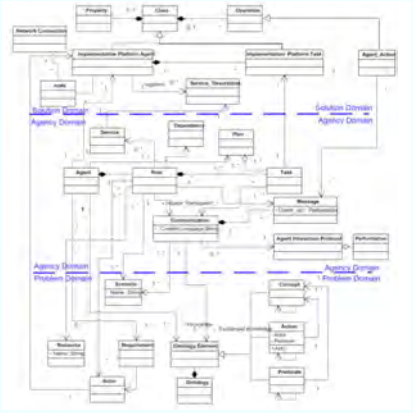
The ADELFE Meta-model



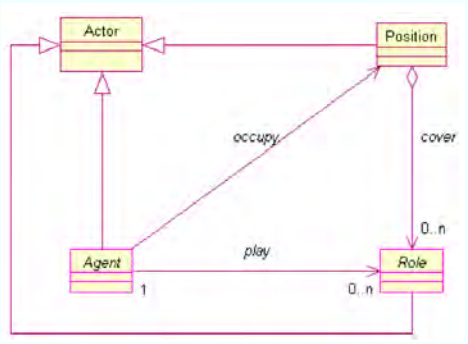
The Gaia Meta-model



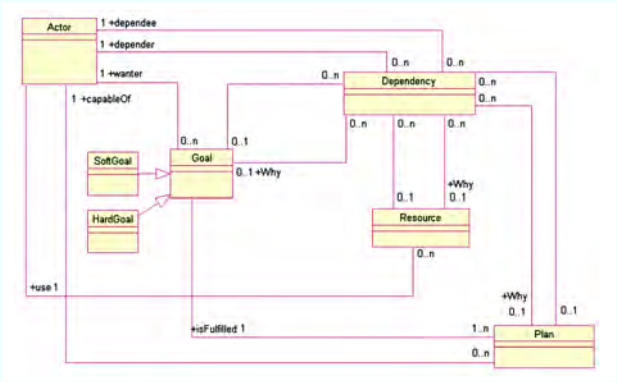
The PASSI Meta-model



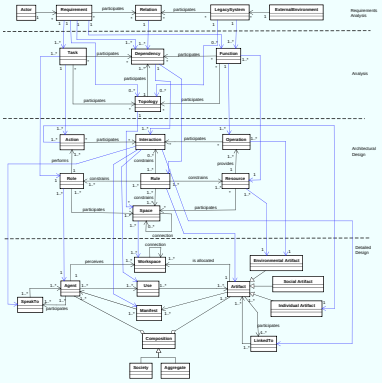
The Tropos Meta-model



The Tropos Meta-model



The SODA Meta-model



Outline

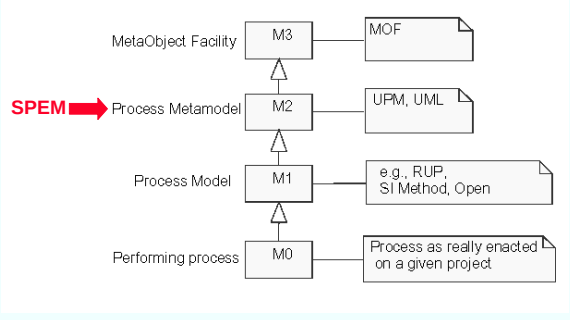
Meta-model

- 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

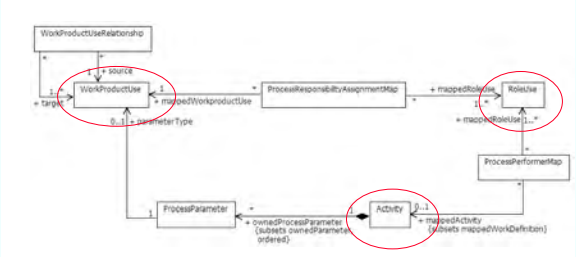


SPEM

- 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

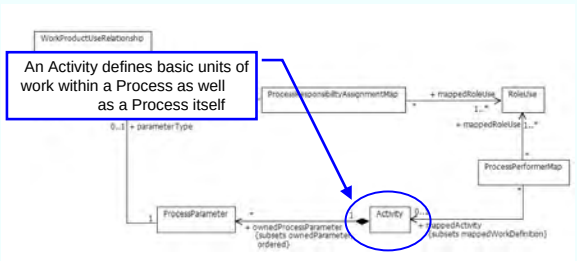
Roles, Activities & WorkProducts

- 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



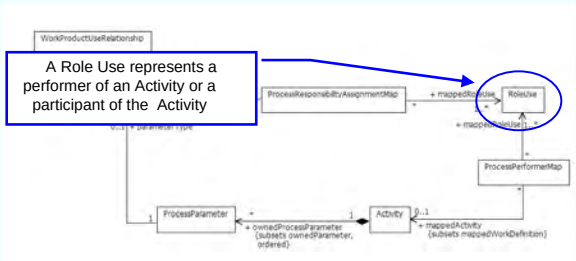
Roles, Activities & WorkProducts

- 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



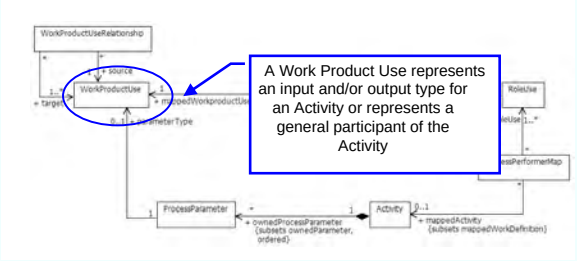
Roles, Activities & WorkProducts

- 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



Roles, Activities & WorkProducts

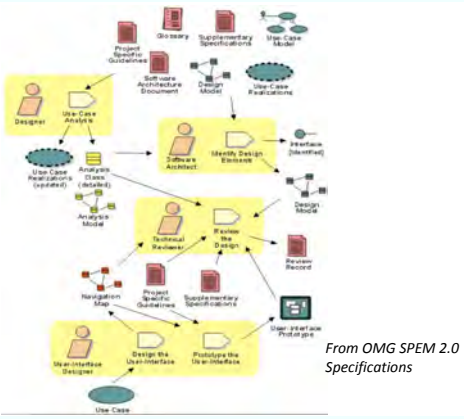
- A software development process is seen as a collaboration between abstract active entities called process roles that perform operations called activities on concrete, tangible entities called work products



SPEM Notation

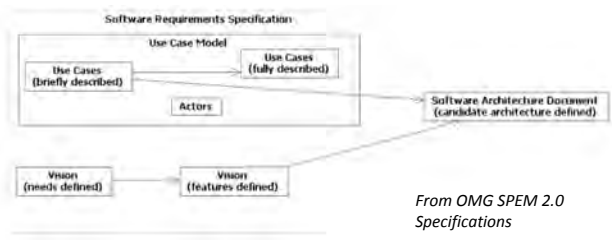
Stereotype	Symbol
Activity	
Category	
Composite role and Team	
Guidance	
Milestone	
Process	
Process Component	
Process Pattern	
Role Definition and Use	
Task Definition and Use	
Tool Definition	
WorkProduct Definition and Use	

SPEM: Activity Details Diagram



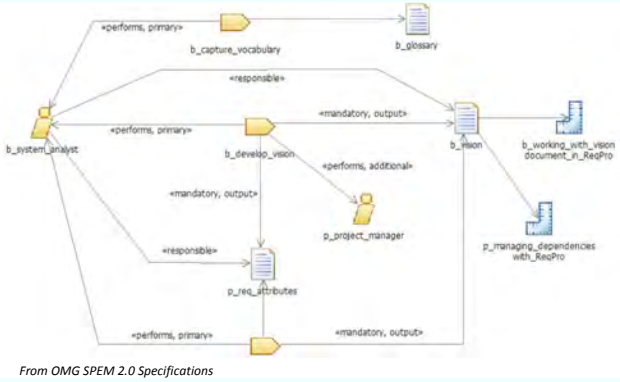
From OMG SPEM 2.0 Specifications

SPEM: Work Product Dependency Diagram



From OMG SPEM 2.0 Specifications

SPEM: Class Diagram



From OMG SPEM 2.0 Specifications

Part III

Process Documentation

AOSE & Processes

- 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

AOSE & Processes

- 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
  - 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
- 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]

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
 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. . .

Part IV

Situational Method Engineering

Outline

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

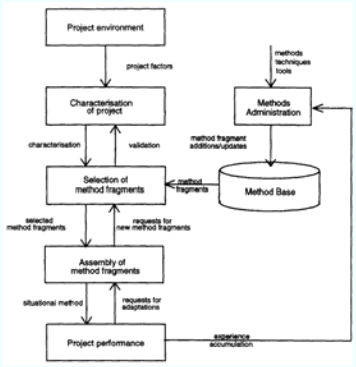
Method Engineering: Concerns

- 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

Situational Methodologies

- 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]



And Now?

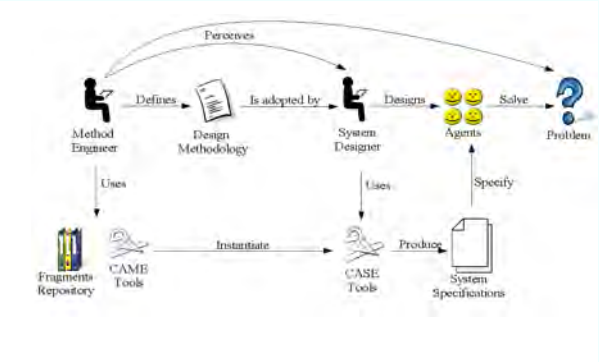
- Two important questions
  - How to represent method fragments?
  - How to assembly method fragments?
- To assemble method fragments into a meaningful method, we need a procedure and representation to model method fragments and impose some constraints or rules on method assembly processes

Outline

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



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

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

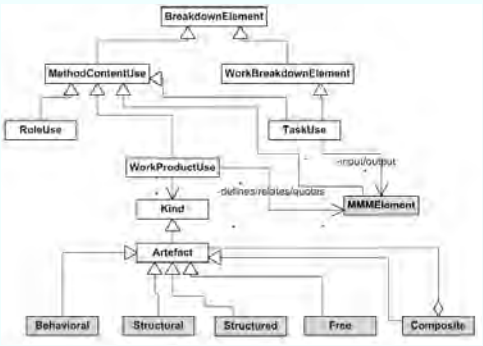
Extending SPEM Specifications [Seidita et al., 2009a]

- 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

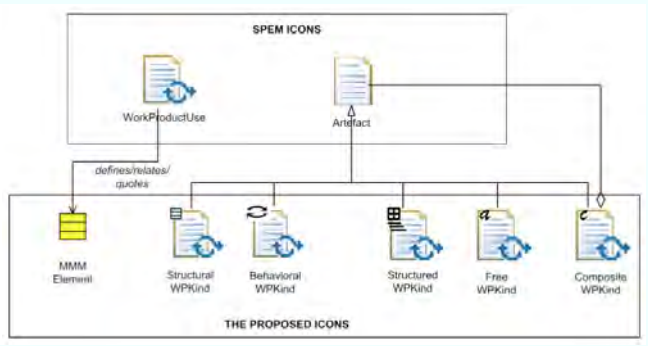
Extending SPEM Specifications [Seidita et al., 2009a]

- 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

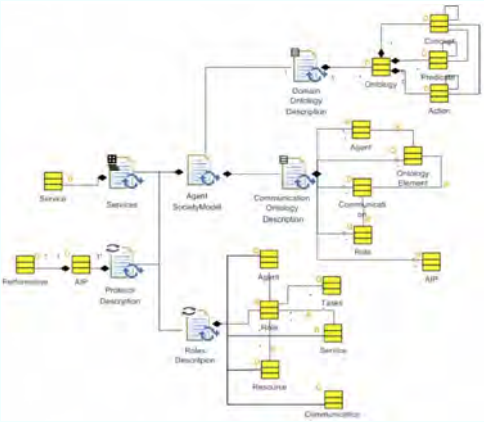
Extending SPEM Specifications [Seidita et al., 2009a]



Proposed icons



The dependency diagram

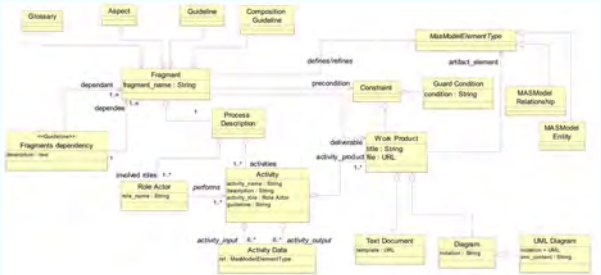


Example: PASSI process activity diagram

Outline

Method fragment meta-model

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



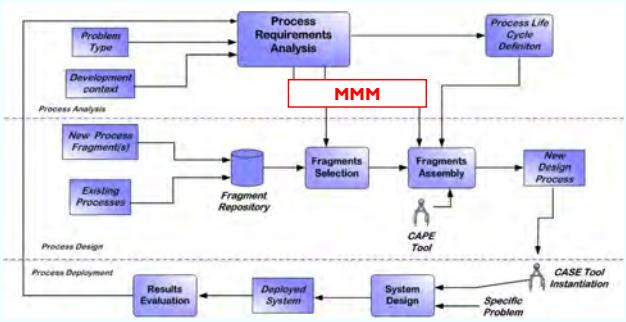
What is a Method Fragment

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

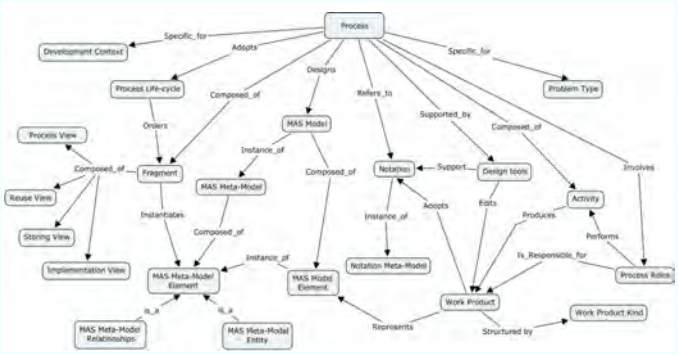
- A portion of process (what is to be done, in what order), defined with a SPEM diagram
- One or more deliverables (like (A)UML/UML diagrams, text documents and so on)
- Some preconditions (they are a kind of constraint because it is not possible to start the process specified in the fragment without 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.

Outline

The Prode approach for Agent-Oriented Method Engineering [Seidita et al., 2009b]



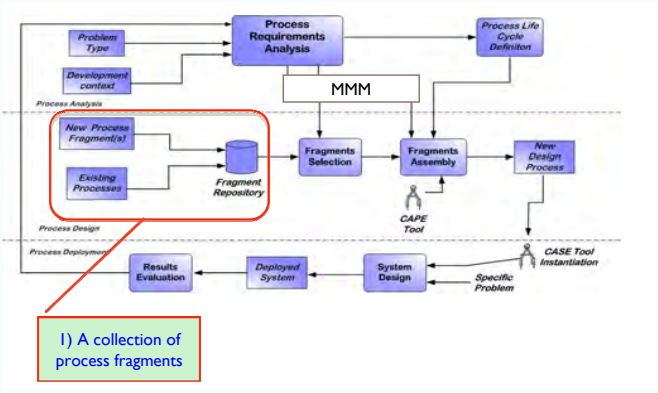
The PRODE Process Representation



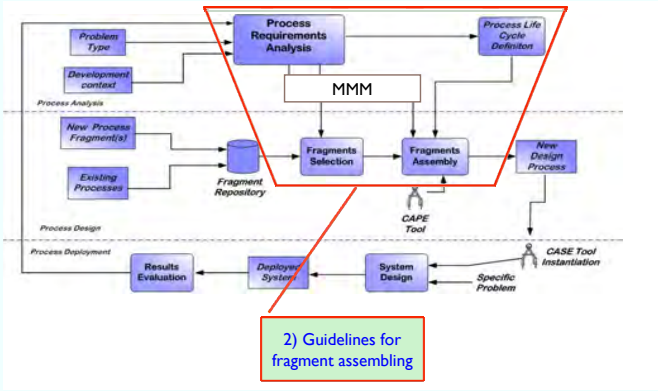
Applying the Proposed Method Fragment Definition

- 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.

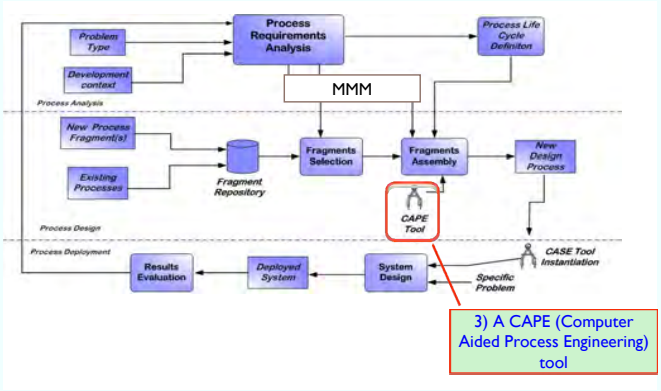
PRODE divided in three main areas of research



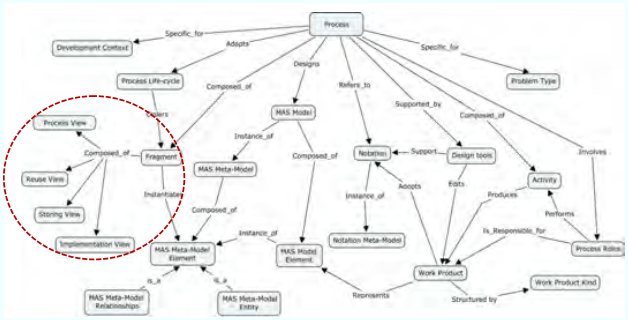
PRODE divided in three main areas of research



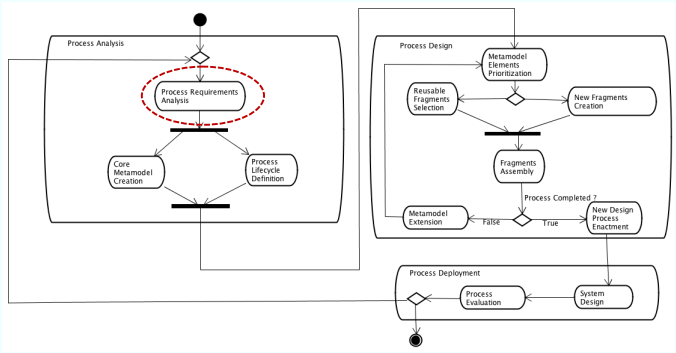
PRODE divided in three main areas of research



The PRODE Process Representation



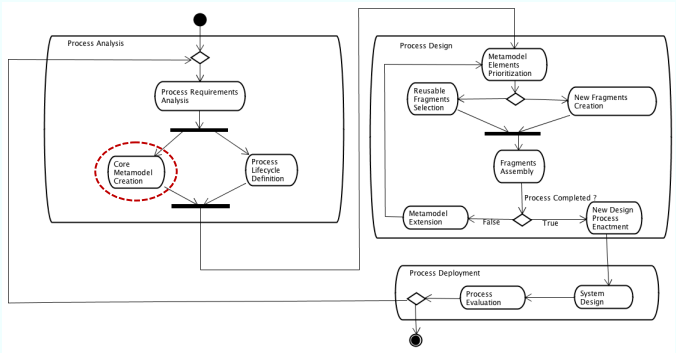
Process Analysis and Design in PRODE



Example: PRODE Analysis

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

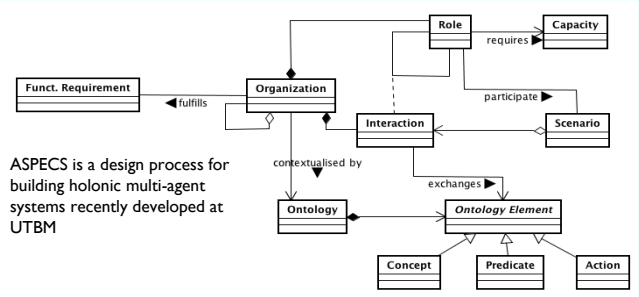
Process Analysis and Design in PRODE



Example: Core meta-model creation

ASPECS Process Requirement	Strategy	Consequence		
		MMME from PASSI	MMME from CRIO	Other
Development of very large MASs for hierarchically decomposable problems	Adoption of holonic decomposition of problems		Capacity, Organization, Role, Interaction, Holon	Organizations, not agents should be the center of the process
Reuse of experiences done with PASSI	Support for functional requirements	Scenario, (Functional) Requirement		
	Early identification of agents on the basis of requirements	Link agent-requirement		Agents should be replaced by organizations
	Transformational approach			3 domains in the MMM
	An ontology should be used to model agent's knowledge	Ontology (including Concepts, Actions, Predicates)		
	FIPA-compliance at least at the communication level	Communication, Message, Interaction Protocol, Ontology, Role		
	Input of the process: text scenarios			Text Scenario is an input of the process

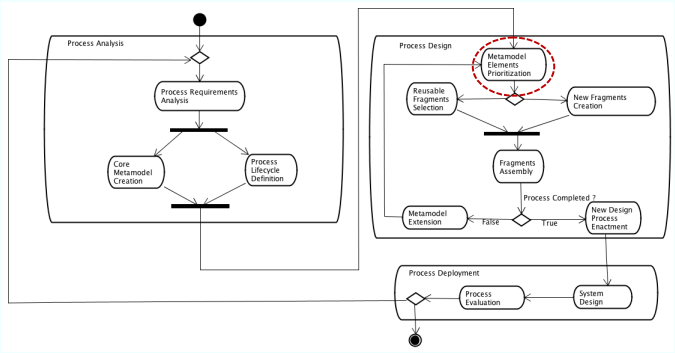
Example: ASPECS core meta-model



ASPECS is a design process for building holonic multi-agent systems recently developed at UTBM

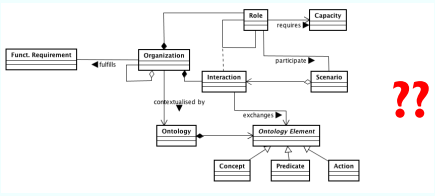
- A detailed description of ASPECS in [Cossentino et al., 2010]

Process Analysis and Design in PRODE



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

The Prioritization Algorithm (1 of 3) [Seidita et al., 2009b]

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;

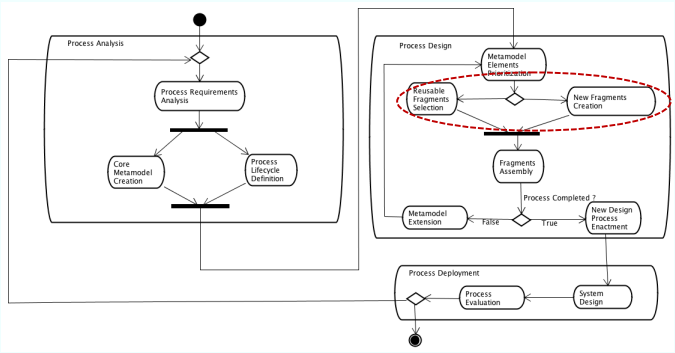
The Prioritization Algorithm (2 of 3)

7. For each element $E1_i$ 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,. . . ,n) and outputs O_j (j=0,. . . ,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 $E1_i$ from List elements1;
      III. For each element $E2_i$ in List_elements2 analyze if there is a similarity with the elements defined in this fragment
         A. if yes delete $E2_i$ from List_elements2 and I_i/O_i from List_elements3
   b) else (if no fragment correspond to requirements and strategies) then
      I. remove $E1_i$ from List_elements1 and insert it in List_elements2

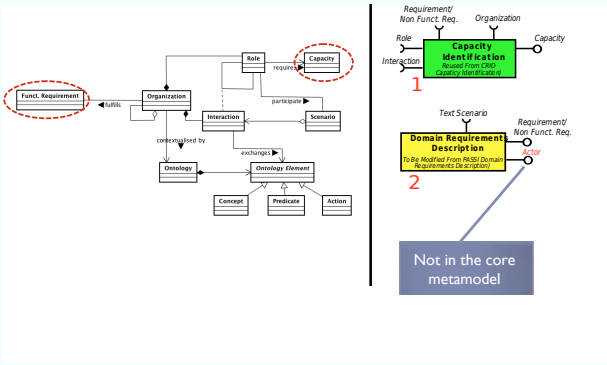
The Prioritization Algorithm (3 of 3)

8. For each $E2_i$ (i=0..m) in List_elements2
   a) Define a new fragment for instantiating $E2_i$
   b) Insert the fragment in the new process composition diagram
   c) Remove $E2_i$ from List_elements2
9. For each $E3_i$ (i=0..m) in List_elements3
   a) Introduce elements $E3_i$ (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.

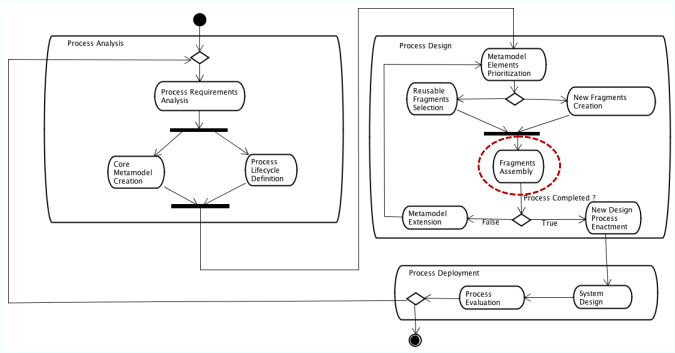
Process Analysis and Design in PRODE



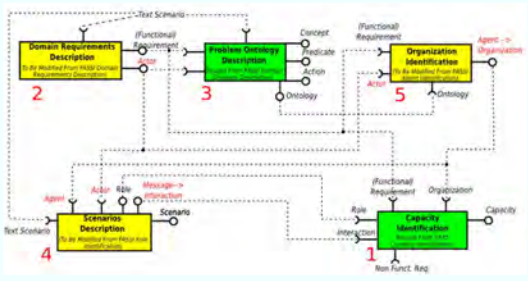
Example: the first two fragments in Building the ASPECS Process



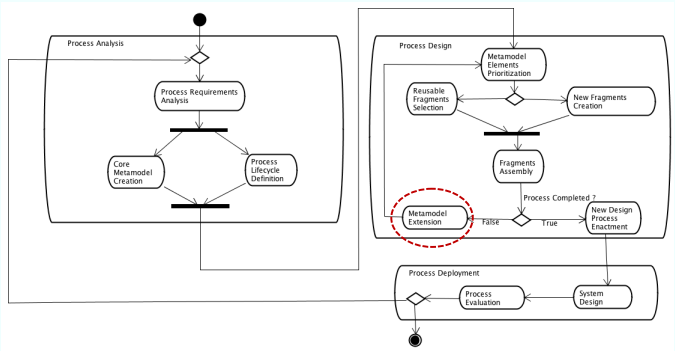
Process Analysis and Design in PRODE



Example: Aspecs process component diagram



Process Analysis and Design in PRODE



Meta-model Extension

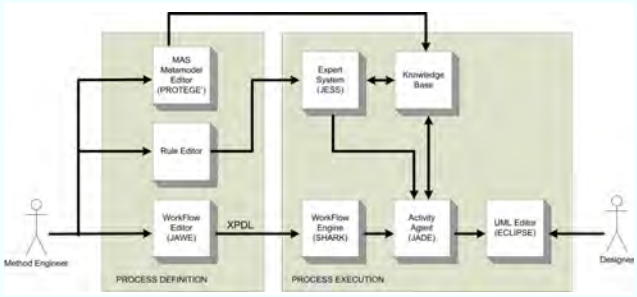
- The Core MAS Metamodel is the starting point for selecting the right fragments from the repository and for assembling them in the new process
- MAS Metamodel extensions come from:
  - the need of incorporating MMMEs referred in selected fragments
  - new process requirements
  - not all design activities 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

[1] M. Cossentino, L. Sabatucci, V. Seidita, S. Gaglio. A Collaborative Tool for Designing and Enacting Design Processes. In Proc. of 24th Annual ACM Symposium on Applied Computing (SAC2009), Agent-Oriented Software Engineering Methodologies and Systems (AOMS@SAC2009) track. 10 March 2009 Honolulu, Hawaii, USA.

Metameth tool architecture



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
→ **Rule Check**

– Auto-composition
– Update
– Import
→ **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 [2]
  - a UML modeling tool (IBM Rational System Developer)
  - (already cited) Jess for realizing the expert system

[2] An open source tool made by Enhydra: http://www.together.at/prod/workflow/twe

Outline

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

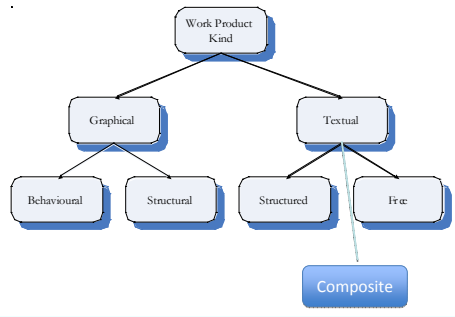
- Requirements
- Analysis
- Design
- Implementation
- Testing
- Deployment
- Coding

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

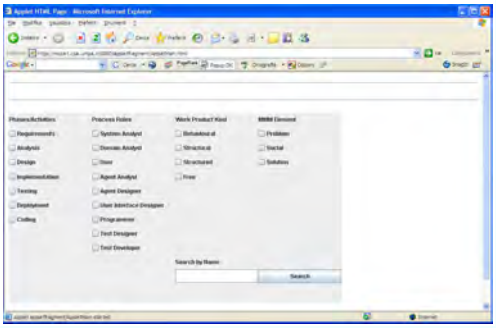
Taxonomy: Work product



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

Fragments retrieval



- 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

AO Design Process Evaluation

- Q.N. Tran, G. C. Low (2005). Comparison of Ten Agent-Oriented Methodologies. In Agent-Oriented Methodologies, chapter XII, pp. 341-367. Idea Group.
- L. Cernuzzi, G. Rossi (2002). On the evaluation of agent oriented methodologies. In: Proc. of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, pp. 21-30.
- Arnon Sturm, Dov Dori, Onn Shehory (2004). A Comparative Evaluation of Agent-Oriented Methodologies, in Methodologies and Software Engineering for Agent Systems, Federico Bergenti, Marie-Pierre Gleizes, Franco Zambonelli (eds.)
- Khanh Hoa Dam, Michael Winikoff (2003). Comparing Agent-Oriented Methodologies. In proc. of the Agent-Oriented Information Systems Workshop at AAMAS03. Melbourne (AUS).
- P. Cuesta, A. Gomez, J. C. Gonzalez, and F. J. Rodriguez (2003). A Framework for Evaluation of Agent Oriented Methodologies. CAEPIA'2003
- L. Cernuzzi, M. Cossentino, F. Zambonelli (2005). Process Models for Agent-Based Development. International Journal on Engineering Applications of Artificial Intelligence (EAAI). Elsevier.

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

Structure of the evaluation framework

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

Details on AO processes evaluation

- From:
  - Arnon Sturm, Dov Dori, Onn Shehory. A Comparative Evaluation of Agent-Oriented Methodologies, in Methodologies and Software Engineering for Agent Systems, Federico Bergenti, Marie-Pierre Gleizes, Franco Zambonelli (eds.)
- 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:
  - Khanh Hoa Dam, Michael Winikoff (2003). Comparing Agent-Oriented Methodologies. In proc. of the Agent-Oriented Information Systems Workshop at AAMAS03. Melbourne (AUS).
- 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/*

Concept/Property	Adelfe	Gaia	Ingenias	OPF	PASSI	Prometheus	TROPOS
Autonomy	H	H	H	H	H/H/M	H	L
Mental attitudes	L	N	H	H	L/L/M	M	M
Proactiveness	M	L	H	H	H/M/H	H	N
Reactiveness	H	L	H	H	H/H/H	H	N
Concurrency	H	M	H	L	H/H/M	H	L
Teamwork and roles	L	H	H	H	M/H/H	L	M
Cooperation model	AMAS th.	Teamwork	ALL	ALL	Task del./ Teamwork	none	Negotiation/ Task del.
Protocols support	H	H	H	H	H/M/H	H	N
Communication modes	ALL	Async mess.	ALL	ALL	Direct	N	
Communication language	ALL	ACL like	ALL	ALL	Speech acts	messages	
Situatedness	H	H	H	H	H/M/M	H	H
Environment type	All episodic	Dynamic Continuous	All discrete	ALL	ALL	ALL	Inacc., Non episodic, Dynam.

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

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

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
  - agreater 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

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 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)

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...

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)

- A. Newell, *The Knowledge Level* [Newell, 1982]
- P. Wegner, *Why Interaction is More Powerful than Algorithms* [Wegner, 1997]
- M. Wooldridge, *Reasoning About Rational Agents* [Wooldridge, 2001]
- M. Wooldridge, N. Jennings, *Intelligent Agents: Theory and Practice* [Wooldridge and Jennings, 1995]
- D. Chess, C. Harrison, A. Kershenbaum, *Mobile Agents: are They a Good Idea?* [Chess et al., 1996]
- V. Parunak, *Go to the Ant: Engineering Principles from Natural Agent Systems* [Parunak, 1997]
- 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]
- N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, B. Odgers, *Autonomous Agents for Business Process Management* [Jennings et al., 2000]
- M. J. Wooldridge and N. R. Jennings, *Software Engineering with Agents: Pitfalls and Pratfalls* [Wooldridge and Jennings, 1999]
- Y. Shoham, *An Overview of Agent-Oriented Programming* [Shoham, 1997]
- K. Siau and M. Rossi, *Evaluation of Information Modeling Methods – A Review* [Siau and Rossi, 1998]
- F. Zambonelli, N. Jennings, M. Wooldridge, *Organizational Abstractions for the Analysis and Design of multi-agent system* [Zambonelli et al., 2001]

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]
  - F. Bergenti, M.-P. Gleizes, F. Zambonelli, *Methodologies and Software Engineering for Agent Systems* [Bergenti 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]
  - C. Iglesias, M. Garijo, J. C. Gonzales, *A Survey of Agent-oriented Methodologies* [Iglesias et al., 1999]
  - J. Gòmez, M.-P. Gleizes, G. Weiss, *A survey of agent-oriented software engineering research* [Gómez et al., 2004]

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

- C. Bernon, M. Cossentino, M.P. Gleizes, P. Turci, *A Study of Some Multi-agent Meta-models* [Bernon et al., 2005b]
- M. Cossentino, N. Gaud, S. Galland, V. Hilaire, A. Koukam, *A Holonic Metamodel for Agent-Oriented Analysis and Design* [Cossentino et al., 2007d]
- 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]
- A. Susi, A. Perini, J. Mylopoulos, P. Giorgini, *The Tropos Metamodel and its Use* [Susi et al., 2005]
- INGENIAS Home Page [Grasia Group, 2009]

References on Processes

- L. Cernuzzi, M. Cossentino, F. Zambonelli, *Process Models for Agent-based Development* [Cernuzzi et al., 2005]
- 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, *Situated Process Engineering for Integrating Processes from Methodologies to Infrastructures* [Molesini et al., 2009d]
- 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],
- A. Molesini, *Meta-Models, Environment and Layers: Agent-Oriented Engineering of Complex Systems* [Molesini, 2008]

References on Method Engineering

- S. Brinkkemper, *Method engineering: engineering the information systems development methods and tools* [Brinkkemper, 1996]
- S. Brinkkemper, M. Saeki, F. Harmsen, *Meta-Modelling Based Assembly Techniques for Situational Method Engineering* [Brinkkemper et al., 1999]
- J.P. Tolvanen, *Incremental method engineering with modeling tools: Theoretical principles and empirical evidence* [Tolvanen, 1998]
- 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]
- V. Seidita, M. Cossentino, S. Galland, N. Gaud, V. Hilaire, A. Koukam and S. Gaglio. The Metamodel: a Starting Point for Design Processes Construction. International Journal of Software Engineering and Knowledge Engineering (IJSEKE). (in printing).

References on MAS Infrastructures

- Surveys
  - M. Dastani, J. J. Gòmez Sanz, *Programming Multi-Agent Systems* [Dastani and Gómez-Sanz, 2005]
- Communication (FIPA-based) Infrastructures
  - F. Bellifemine, A. Poggi, G. Rimassa, *Developing Multi-Agent Systems with a FIPA-Compliant Agent Framework* [Bellifemine et al., 2001]
  - 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
  - P. Ciancarini, A. Omicini, F. Zambonelli, *Multi-agent System Engineering: The Coordination Viewpoint* [Ciancarini et al., 2000]
  - 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]

Bibliography

Bellifemine, F., Poggi, A., and Rimassa, G. (2001).
Developing multi-agent systems with a fipa-compliant agent framework.
Software—Practice & Experience, 31(2):103–128.

Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors (2004).
Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*.
Kluwer Academic Publishers.

Bernon, C., Camps, V., Gleizes, M.-P., and Picard, G. (2005a).
Engineering adaptive multi-agent systems: the adelfe methodology.
In *Agent Oriented Methodologies*, chapter VII, pages 172–202. Idea Group Publishing.

Bernon, C., Cossentino, M., Gleizes, M., and Turci, P. (2005b).
A study of some multi-agent meta-models.
Agent-Oriented Software Engineering V: 5th International . . . .

Bernon, C., Cossentino, M., Gleizes, M. P., Turci, P., and Zambonelli, F. (2004).
A study of some multi-agent meta-models.
In Odell, J., Giorgini, P., and Müller, J. P., editors, *AOSE*, volume 3382 of *Lecture Notes in Computer Science*, pages 62–77. Springer.

Bernon, C., Cossentino, M., and Pavòn, J. (2005c).
An overview of current trends in european aose research.
Informatica Journal, 29(4).

Bernon, C., Cossentino, M., and Pavòn, J. (2006).
Agent-oriented software engineering.
The Knowledge Engineering Review, 20(02):99–116.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004).
Tropos: An agent-oriented software development methodology.
Autonomous Agent and Multi-Agent Systems, 3:203–236.

Brinkkemper, S. (1996).
Method engineering: engineering of information systems development methods and tools.
Information & Software Technology, 38(4):275–280.

Brinkkemper, S., Saeki, M., and Harmsen, F. (1999).
Meta-modelling based assembly techniques for situational method engineering.
Inf. Syst., 24(3):209–228.

Busetta, P., Ronnquist, R., Hodgson, A., and Lucas, A.
Jack intelligent agent.
http://www.agent-software.com.au/.

Cabri, G., Leonardi, L., and Zambonelli, F. (2002).
Engineering mobile agent applications via context-dependent coordination.
Software Engineering, IEEE Transactions on, 28(11):1039–1055.

Caire, G., Coulier, W., Garijo, F., Gòmez-Sanz, J., Pavòn, J., Kearney, P., and Massonet, P. (2004).
The MESSAGE methodology.
In [Bergenti et al., 2004], chapter 9, pages 177–194.

Caire, G., Coulier, W., Garijo, F. J., Gomez, J., Pavòn, J., Leal, F., Chainho, P., Kearney, P. E., Stark, J., Evans, R., and Massonet, P. (2002).
Agent oriented analysis using Message/UML.
In Wooldridge, M., Weiss, G., and Ciancarini, P., editors, *Agent-Oriented Software Engineering II*, volume 2222 of *LNCS*, pages 119–135. Springer.
2nd International Workshop (AOSE 2001), Montreal, Canada, 29 May 2001. Revised Papers and Invited Contributions.

Cernuzzi, L., Cossentino, M., and Zambonelli, F. (2005).
Process models for agent-based development.
Engineering Applications of Artificial Intelligence, 18(2):205–222.

Cervenka, R., Trencansky, I., Calisti, M., and Greenwood, D. (2005).
AML: Agent Modeling Language Toward Industry-Grade Agent-Based Modeling.
Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004: Revised Selected Papers.

Chella, A., Cossentino, M., Sabatucci, L., and Seidita, V. (2006).
Agile passi: An agile process for designing agents.
Journal of Computer Systems Science and Engineering.

Chess, D. M., Harrison, C. G., and Kershenbaum, A. (1996).
Mobile agents: Are they a good idea?
In Vitek, J. and Tschudin, C. F., editors, *Mobile Object Systems*, volume 1222 of *Lecture Notes in Computer Science*, pages 25–45. Springer.

Ciancarini, P., Omicini, A., and Zambonelli, F. (2000).
Multiagent system engineering: The coordination viewpoint.
In Jennings, N. R. and Lespérance, Y., editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *LNAI*, pages 250–259. Springer.
6th International Workshop (ATAL'99), Orlando, FL, USA, 15–17 July 1999. Proceedings.

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

Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., and Russo, W. (2008).
Passim: A simulation-based process for the development of multi-agent systems.
International Journal on Agent Oriented Software Engineering (IJAOSE).

Cossentino, M., Gaglio, S., Garro, A., and Seidita, V. (2007a).
Method fragments for agent design methodologies: from standardisation to research.
International Journal of Agent-Oriented Software Engineering (IJAOSE), 1(1):91–121.

Cossentino, M., Gaglio, S., Garro, A., and Seidita, V. (2007b).
Method fragments for agent design methodologies: from standardisation to research.
International Journal of Agent Oriented Software Engineering, 1(1):91–121.

Cossentino, M., Gaglio, S., Sabatucci, L., and Seidita, V. (2005).
The passi and agile passi mas meta-models compared with a unifying proposal.
In *Proc. of the CEEMAS'05 Conference. Lecture Notes in Computer Science*, pages 183–192. Springer Verlag, Budapest, Hungary.

Cossentino, M., Gaglio, S., and V., S. (2007c).
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).*

Cossentino, M., Gaud, N., Galland, S., Hilaire, V., and Koukam, A. (2007d).
A Holonic Metamodel for Agent-Oriented Analysis and Design.
LECTURE NOTES IN COMPUTER SCIENCE, 4659:237.

Cossentino, M., Gaud, N., Hilaire, V., Galland, S., and Koukam, A. (2010).
Aspecs: an agent-oriented software process for engineering complex systems.
International Journal of Autonomous Agents and Multi-Agent Systems (IJAAMAS).

Dastani, M. and Gómez-Sanz, J. (2005).
Programming multi-agent systems.
Knowledge Engineering Review.

DeLoach, S., Wood, M., and Sparkman, C. (2001).
Multiagent Systems Engineering.
International Journal of Software Engineering and Knowledge Engineering, 11(3):231–258.

DeLoach, S. A. (2006).
Engineering organization-based multiagent systems.
In Garcia, A. F., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., and Romanovsky, A., editors, *Software Engineering for Multi-Agent Systems IV. Research Issues and Practical Applications*, volume 3914 of *LNCS*, pages 109–125. Springer.

DeLoach, S. A. (2008).
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/.

Fuggetta, A. (2000).
Software process: a roadmap.
In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 25–34, New York, NY, USA. ACM Press.

Garijo, F. J., Gòmez-Sanz, J. J., and Massonet, P. (2005).
The MESSAGE methodology for agent-oriented analysis and design.
In [Henderson-Sellers and Giorgini, 2005], chapter VIII, pages 203–235.

Ghezzi, C., Jazayeri, M., and Mandrioli, D. (2002).
Foundamental of Software Engineering.
Prentice Hall, second edition.

Gómez, J. J., Gleizes, M.-P., and Weiss, G. (2004).
Methodologies and Software Engineering for Agent Systems, chapter A survey of agent-oriented software engineering research.
Kluwer.

Grasia Group (2009).
Ingenias meta model.
http://grasia.fdi.ucm.es/main/?q=en/node/135.

Henderson-Sellers, B. (2003).
Method engineering for oo systems development.
Commun. ACM, 46(10):73–78.

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

Henderson-Sellers, B. and Debenham, J. (2003).
Towards open methodological support for agent-oriented system development.
In *Procs. First International Conference on Agent-Based Technologies and Systems*, pages 14–24, Canada.

Henderson-Sellers, B. and Giorgini, P. (2005).
Agent Oriented Methodologies.
Idea Group Publishing, Hershey, PA, USA.

Henderson-Sellers, B. and Gonzalez-Perez, C. (2005).
A comparison of four process metamodels and the creation of a new generic standard.
Information and Software Technology, 47(1):49 –65.

Iglesias, C., Garijo, M., and Gonzalez, J. (1999).
A Survey of Agent-Oriented Methodologies.
Intelligent Agents V: Agent Theories, Architectures, and Languages: 5th International Workshop, Atal'98: Paris, France, July 1998: Proceedings.

Jennings, N. R. (2000).
On agent-based software engineering.
Artificial Intelligence, 117(2):277–296.

Jennings, N. R. (2001).
An agent-based approach for building complex software systems.
Communication ACM, 44(4):35–41.

Jennings, N. R., Faratin, P., Norman, T. J., O'Brien, P., and Odgers, B. (2000).
Autonomous agents for business process management.
Int. Journal of Applied Artificial Intelligence, 2(14):145–189.

Luck, M., Ashri, R., and D'Inverno, M. (2004).
Agent-Based Software Development.
Artech House.

Molesini, A. (2008).
Meta-Models, Environment and Layers: Agent-Oriented Engineering of Complex Systems.
PhD thesis, Dottorato in Ingegneria Elettronica, Informatica e delle Telecomunicazioni, Bologna, Italy.

Molesini, A., Denti, E., and Omicini, A. (2005).
MAS meta-models on test: UML vs. OPM in the SODA case study.
In Pěchouček, M., Petta, P., and Varga, L. Z., editors, *Multi-Agent Systems and Applications IV*, volume 3690 of *LNAI*, pages 163–172. Springer.
4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05), Budapest, Hungary, 15–17 September 2005, Proceedings.

Molesini, A., Denti, E., and Omicini, A. (2008a).
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.

Molesini, A., Denti, E., and Omicini, A. (2009a).
Agent-based conference management: A case study in SODA.
International Journal of Agent-Oriented Software Engineering.

Molesini, A., Denti, E., and Omicini, A. (2009b).
An agent-oriented software engineering approach to home intelligence.
In Filipe, J., Fred, A., and Sharp, B., editors, *International Conference on Agents and Artificial Intelligence (ICAART 2009)*, pages 377–384, Porto, Portugal. INSTICC.

Molesini, A., Denti, E., and Omicini, A. (2009c).
RBAC-MAS & SODA: Experimenting RBAC in AOSE.
In Artikis, A., Picard, G., and Vercouter, L., editors, *Engineering Societies in the Agents World IX*, volume 5485 of *LNCS*. Springer.
9th International Workshop (ESAW'08), 24–26 September 2008, Saint-Étienne, France. Revised Selected Papers.

Molesini, A., Nardini, E., Denti, E., and Omicini, A. (2008b).
Advancing object-oriented standards toward agent-oriented methodologies: SPEM 2.0 on SODA.
In Baldoni, M., Cossentino, M., De Paoli, F., and Seidita, V., editors, *9th Workshop "From Objects to Agents" (WOA 2008) – Evolution of Agent Development: Methodologies, Tools, Platforms and Languages*, pages 108–114, Palermo, Italy. Seneca Edizioni.

Molesini, A., Nardini, E., Denti, E., and Omicini, A. (2009d).
Situated process engineering for integrating processes from methodologies to infrastructures.
In Shin, S. Y., Ossowski, S., Menezes, R., and Viroli, M., editors, *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, volume II, pages 699–706, Honolulu, Hawai'i, USA. ACM.

Nardini, E., Molesini, A., Omicini, A., and Denti, E. (2008).
SPEM on test: the SODA case study.
In Wainwright, R. L., Haddad, H. M., Menezes, R., and Viroli, M., editors, *23th ACM Symposium on Applied Computing (SAC 2008)*, volume 1, pages 700–706, Fortaleza, Ceará, Brazil. ACM.
Special Track on Software Engineering.

Newell, A. (1982).
The knowledge level.
Artificial Intelligence, 18(1):87–127.

Numi Tran, Q.-N. and Low, G. C. (2005).
Comparison of ten agent-oriented methodologies.
In [Henderson-Sellers and Giorgini, 2005], chapter XII, pages 341–367.

Object Management Group (2008).
Software & Systems Process Engineering Meta-Model Specification 2.0.
`http://www.omg.org/spec/SPEM/2.0/PDF`.

OPEN Working Group.
Open home page.
http://www.open.org.au/index.html.

Padgham, L. and Winikof, M. (2003).
Prometheus: A methodology for developing intelligent agents.
In Giunchiglia, F., Odell, J., and Weiss, G., editors, *Agent-Oriented Software Engineering III*, volume 2585 of *LNCS*, pages 174–185. Springer.
3rd International Workshop (AOSE 2002), Bologna, Italy, 15 July 2002. Revised Papers and Invited Contributions.

Padgham, L. and Winikoff, M. (2004).
Developing intelligent agent systems: a practical guide.
John Wiley and Sons.

Padgham, L., Winikoff, M., DeLoach, S., and Cossentino, M. (2009).
A unified graphical notation for aose.
In Agent-Oriented Software Engineering 2008. Lecture Notes in Computer Science, 5386-0086.

Parunak, H. V. D. (1997).
"go to the ant": Engineering principles from natural agent systems.
Annals of Operation Research, 75:69–101.

Pavòn, J., Gòmez-Sanz, J. J., and Fuentes, R. (2005).
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.
http://fipa-os.sourceforge.net.

Ralyté, J. and Rolland, C. (2001a).
An approach for method reengineering.
In Kunii, H. S., Jajodia, S., and Sølvberg, A., editors, *ER*, volume 2224 of *Lecture Notes in Computer Science*, pages 471–484. Springer.
Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling, Yokohama, Japan, November 27-30, 2001, Proceedings.

Ralyté, J. and Rolland, C. (2001b).
An assembly process model for method engineering.
In Dittrich, K. R., Geppert, A., and Norrie, M. C., editors, *CAiSE*, volume 2068 of *Lecture Notes in Computer Science*, pages 267–283. Springer.
Advanced Information Systems Engineering, 13th International Conference, CAiSE 2001, Interlaken, Switzerland, June 4-8, 2001, Proceedings.

Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009).
Environment programming in CArtAgO.
In Bordini, R. P., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., editors, *Multi-Agent Programming II: Languages, Platforms and Applications*, Multiagent Systems, Artificial Societies, and Simulated Organizations, chapter 8, pages 259–288. Springer.

SEI (2006a).
Cmmi for development version 1.2.
Technical report, Software Engineering Institute of the Carnagie Mellon University.

SEI (2006b).
Standard cmmi appraisal method for process improvement (scampi) a, version 1.2: Method definition document.
Technical report, Software Engineering Institute of the Carnagie Mellon University.

Seidita, V., Cossentino, M., and Gaglio, S. (2006).
A repository of fragments for agent systems design.
In *Proc. Of the Workshop on Objects and Agents (WOA06*, pages 130–137.

Seidita, V., Cossentino, M., and Gaglio, S. (2008).
Using and extending the spem specifications to represent agent oriented methodologies.
In Luck, M. and Gómez-Sanz, J. J., editors, *AOSE*, volume 5386 of *Lecture Notes in Computer Science*, pages 46–59. Springer.

Seidita, V., Cossentino, M., and Gaglio, S. (2009a).
Using and extending the spem specifications to represent agent oriented methodologies.
In *Agent-Oriented Software Engineering 2008. Lecture Notes in Computer Science*, volume 5386-0086. Springer-Verlag GmbH.

Seidita, V., Cossentino, M., Galland, S., Gaud, N., Hilaire, V., Koukam, A., and Gaglio, S. (2009b).
The metamodel: a starting point for design processes construction.
International Journal of Software Engineering and Knowledge Engineering (IJSEKE).

Shoham, Y. (1997).
An overview of agent-oriented programming.
pages 271–290.

Siau, K. and Rossi, M. (1998).
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.

Sommerville, I. (2007).
Software Engineering 8th Edition.
Addison-Wesley.

Susi, A., Perini, A., Mylopoulos, J., and Giorgini, P. (2005).
The tropos metamodel and its use.
Informatica (Slovenia), 29(4):401–408.

Tolvanen, J.-P. (1998).
Incremental method engineering with modeling tools: Theoretical principles and empirical evidence.
PhD thesis.

Viroli, M., Casadei, M., and Omicini, A. (2009).
A framework for modelling and implementing self-organising coordination.
In Shin, S. Y., Ossowski, S., Menezes, R., and Viroli, M., editors, *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, volume III, pages 1353–1360, Honolulu, Hawai'i, USA. ACM.

Wegner, P. (1997).
Why interaction is more powerful than algorithms.
Commun. ACM, 40(5):80–91.

Wooldridge, M. (2001).
Reasoning about rational agents.
MIT Press, Cambridge, MA, USA.

Wooldridge, M. and Jennings, N. R. (1995).
Intelligent agents: Theory and practice.
Knowledge Engineering Review, 10(2):115–152.

Wooldridge, M., Jennings, N. R., and Kinny, D. (2000).
The gaia methodology for agent-oriented analysis and design.
Autonomous Agents and Multi-Agent Systems, 3(3):285–312.

Wooldridge, M. J. and Jennings, N. R. (1999).
Software engineering with agents: Pitfalls and pratfalls.
IEEE Internet Computing, 3(3):20–27.

Zambonelli, F., Jennings, N. R., and Wooldridge, M. (2001).
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.

Zambonelli, F., Jennings, N. R., and Wooldridge, M. (2003).
Developing multiagent systems: The Gaia methodology.
ACM Transactions on Software Engineering and Methodology (TOSEM), 12(3):317–370.

Zambonelli, F. and Omicini, A. (2004).
Challenges and research directions in agent-oriented software engineering.
Autonomous Agents and Multi-Agent Systems, 9(3):253–283.
Special Issue: Challenges for Agent-Based Computing.

..... THANKS!!!

Documentation and Fragmentation of Agent Oriented Methodologies and Processes

Ambra Molesini[1] Massimo Cossentino[2]

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

[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