

Agent Oriented Software Engineering

Ambra Molesini¹ Massimo Cossentino²

¹ALMA MATER STUDIORUM – Università di Bologna (Italy)
`ambra.molesini@unibo.it`

²Italian National Research Council - ICAR Institute - Palermo (Italy)
`cossentino@pa.icar.cnr.it`

11th European Agent Systems Summer School
Torino, Italy
September 2009

Outline

- Part I: General Concepts
- Part II: Agent Oriented Software Engineering
- Part III: Meta-models
- Part IV: Situational Method Engineering
- Part V: Research directions and conclusions

Outline of Part I

1 Software Engineering

2 Software Processes

3 Methodologies

Outline

- Part I: General Concepts
- **Part II: Agent Oriented Software Engineering**
- Part III: Meta-models
- Part IV: Situational Method Engineering
- Part V: Research directions and conclusions

Outline of Part II

4 Why do we need AOSE?

5 Why agents and MAS?

Outline

- Part I: General Concepts
- Part II: Agent Oriented Software Engineering
- **Part III: Meta-models**
- Part IV: Situational Method Engineering
- Part V: Research directions and conclusions

Outline of Part III

6 MAS Meta-model

7 Process Meta-model

- SPEM
- OPF & OPEN

Outline

- Part I: General Concepts
- Part II: Agent Oriented Software Engineering
- Part III: Meta-models
- **Part IV: Situational Method Engineering**
- Part V: Research directions and conclusions

Outline of Part IV

8 Method Engineering in traditional SE

- Method Fragment Representation
- Method Fragment Assembly

9 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

Outline

- Part I: General Concepts
- Part II: Agent Oriented Software Engineering
- Part III: Meta-models
- Part IV: Situational Method Engineering
- Part V: Research directions and conclusions

Outline of Part V

10 Research directions and visions

11 Conclusions

Part I

General Concepts

Outline

1 Software Engineering

2 Software Processes

3 Methodologies

Software

- Software is **abstract** and **intangible** [Sommerville, 2007]:
 - ▶ it is not constrained by materials, or governed by physical laws, or by manufacturing process
- On the one hand, this simplifies software engineering as there are no physical limitations on the potential of software
- On the other hand, the lack of natural constraints means that software can easily become extremely complex and hence very difficult to understand
- So, software engineers should
 - ▶ adopt a **systematic and organised approach** to their work
 - ▶ use **appropriate tools and techniques** depending on the problem to be solved, the development constraints and the resources available

Software Engineering

What is Software Engineering?

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

Software Engineering

What is Software Engineering?

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

What is the aim of Software Engineering?

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

Software Engineering: Concerns

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

Software Engineering Abstractions

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

Outline

1 Software Engineering

2 Software Processes

3 Methodologies

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.

Software Process: Activities

Generic activities in all software processes are [Sommerville, 2007]:

Specification — What the system should do and its development constraints

Development — Production of the software system

Validation — Checking that the software is what the customer wants

Evolution — Changing the software in response to changing demands

The Ideal Software Process

The Ideal Software Process?

There is no an ideal process

[Sommerville, 2007]

Many Sorts of Software Processes

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

Software Process Model

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

Software Process Model: Examples

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

Generic Software Process Models

- Generic process models

- Waterfall** — separate and distinct phases of specification and development

- Iterative development** — specification, development and validation are interleaved

- Component-based software engineering** — the system is assembled from existing components

Outline

1 Software Engineering

2 Software Processes

3 Methodologies

Methodologies vs. Methods: General Issue

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

Methodologies vs. Methods in Software Engineering

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

Method

Method [Cernuzzi et al., 2005]

- A method prescribes a way of performing some kind of activity within a process, in order to properly produce a specific output (i.e., an artefact or a document) starting from a specific input (again, an artefact or a document).
- Any phase of a process, to be successfully applicable, should be complemented by some methodological guidelines (including the identification of the techniques and tools to be used, and the definition of how artifacts have been produced) that could help the involved stakeholders in accomplishing their work according to some defined best practices

Methodology [Ghezzi et al., 2002]

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

Methodologies vs. Software Process

- Based on the above definitions, and comparing software processes and methodologies, we can find some common elements in their scope [Cernuzzi et al., 2005]
 - ▶ both are focusing on what we have to do in the different activities needed to construct a software system
 - ▶ however, while the software development process is more centered on the global process including all the stages, their order and time scheduling, the methodology focuses more directly on the specific techniques to be used and artifacts to be produced
- In this sense, we could say that methodologies focus more explicitly on *how* to perform the activity or tasks in some specific stages of the process, while processes may also cover more general management aspects, e.g., basic questions about *who* and *when*, and *how much*

Example: OO Software Engineering

- Abstractions:
 - ▶ objects, classes, inheritance, services.
- Processes:
 - ▶ RUP, OPEN, etc. . .
- Methodologies:
 - ▶ object-oriented analysis and design. . .
 - ▶ centered around object-oriented abstractions
- Tools (Modeling Techniques):
 - ▶ UML (standard), E-R, finite state automata, visual languages. . .

Part II

Agent Oriented Software Engineering

Outline

4 Why do we need AOSE?

5 Why agents and MAS?

Why do we need Agent-Oriented Software Engineering?

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

What are agents?

- There has been some debate on what an agent is, and what could be appropriately called an agent
- Two main viewpoints (centered on different perspectives on autonomy):
 - ▶ the (strong) Artificial Intelligence viewpoint
 - ★ an agent must be, **proactive**, **intelligent**, and it must **converse** instead of doing client-server computing
 - ▶ the (weak) Software Engineering Viewpoint
 - ★ an agent is a **software component** with internal (either reactive or proactive) **threads** of execution, and that can be engaged in complex and stateful **interactions protocols**

What are Multiagent Systems?

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

SE Viewpoint on Agent-Oriented Computing

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

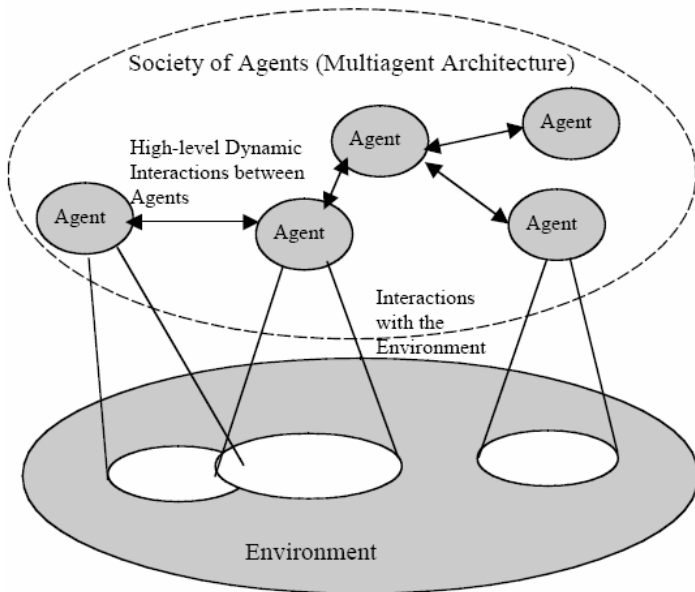
SE Implications of Agent Characteristics

- Autonomy
 - ▶ control encapsulation as a dimension of modularity
 - ▶ conceptually simpler to tackle than a single (or multiple inter-dependent) locus of control
- Situatedness
 - ▶ clear separation of concerns between
 - ★ the active computational parts of the system (the agents)
 - ★ the resources of the environment
- Interactivity
 - ▶ communication, coordination
 - ▶ collaborative or competitive interaction
 - ▶ not a single characterising protocol of interaction
 - ▶ interaction protocol as an additional SE dimension

SE Implications of Agent Characteristics

- Sociality
 - ▶ not a single characterising protocol of interaction
 - ▶ interaction as an additional SE dimension
- Openness
 - ▶ controlling self-interested agents, malicious behaviours, and badly programmed agents
 - ▶ dynamic re-organisation of software architecture
- Mobility and Locality
 - ▶ additional dimension of autonomous behaviour
 - ▶ improve locality in interactions

MAS Characterisation



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, organisations, responsibilities, beliefs, desires and intentions. . .
 - ▶ not directly translating into concrete entities of the software system
 - ★ e.g. the concept of role is an aspect of an agent, not an agent

Agent-Oriented Tools

- SE requires tools to
 - ▶ represent software
 - ★ e.g., interaction diagrams, E-R diagrams, etc. . .
 - ▶ verify properties
 - ★ e.g., petri nets, formal notations, etc. . . .
- AOSE requires
 - ▶ specific agent-oriented tools
 - ★ e.g., UML per se is not suitable to model agent systems and their interactions (object-oriented abstractions not agent-oriented ones)

Outline

4 Why do we need AOSE?

5 Why agents and MAS?

Why Agents and Multi-Agent Systems?

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

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

MASs already proved effective in dealing with open, distributed, complex problems that are considered hard challenges for classical software engineering approaches

There is much more to agent-oriented software engineering

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

Agents and Multi-Agent Systems are (virtually) Everywhere

- Examples of components that can be modelled (and observed) in terms of agents:
 - ▶ autonomous network processes
 - ▶ computing-based sensors
 - ▶ PDAs
 - ▶ robots
- Example of software systems that can be modelled as multi-agent systems:
 - ▶ internet applications
 - ▶ P2P systems
 - ▶ sensor networks
 - ▶ pervasive computing systems

Part III

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

6 MAS Meta-model

7 Process Meta-model

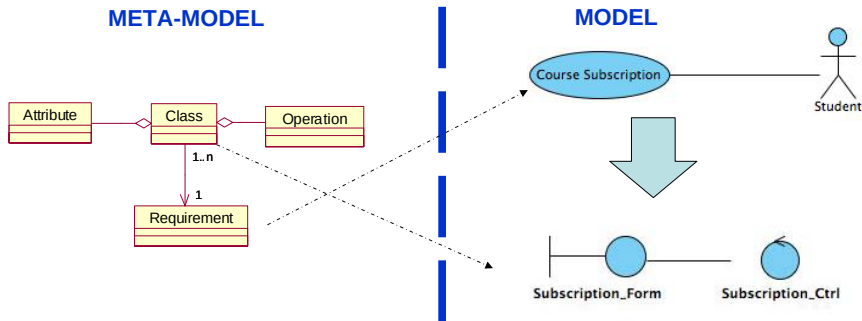
- SPEM
- OPF & OPEN

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

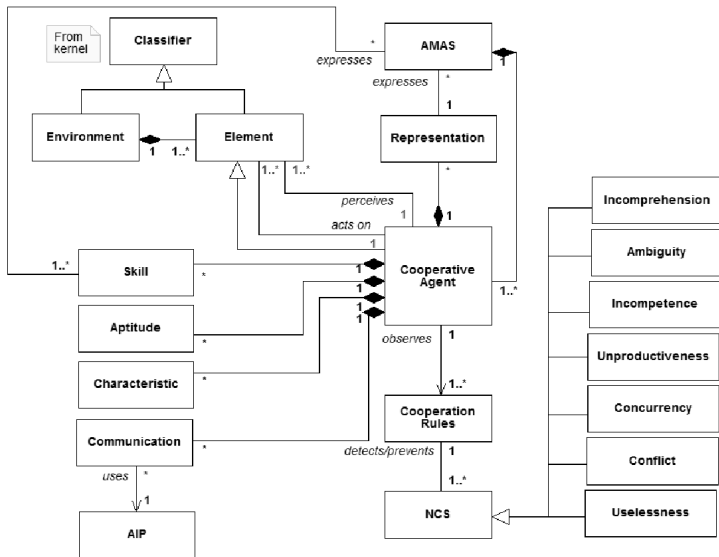
- Designing a software means instantiating its meta-model



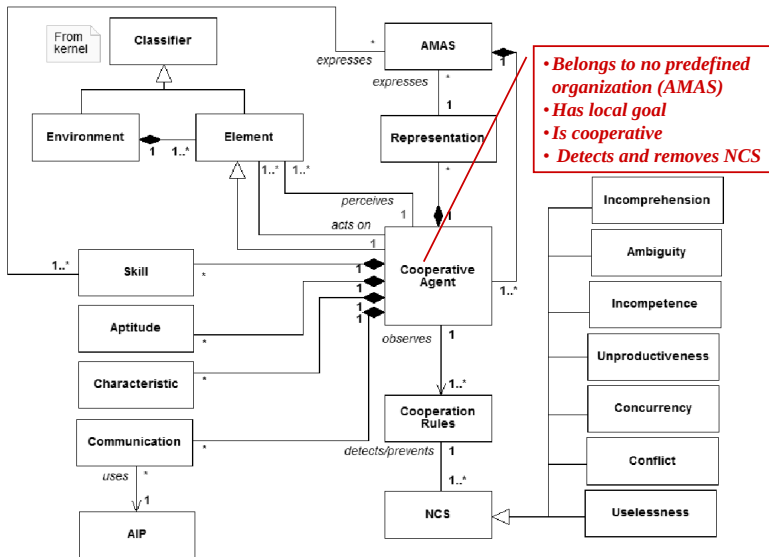
Example of MAS meta-models

- From existing AO design methodologies
 - ▶ ADELFE
 - ▶ Gaia
 - ▶ PASSI
 - ▶ TROPOS
 - ▶ SODA

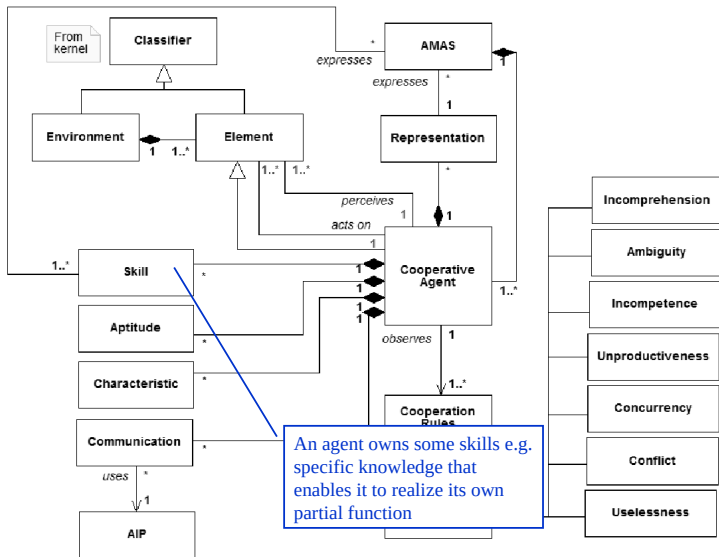
The ADELFE Meta-model



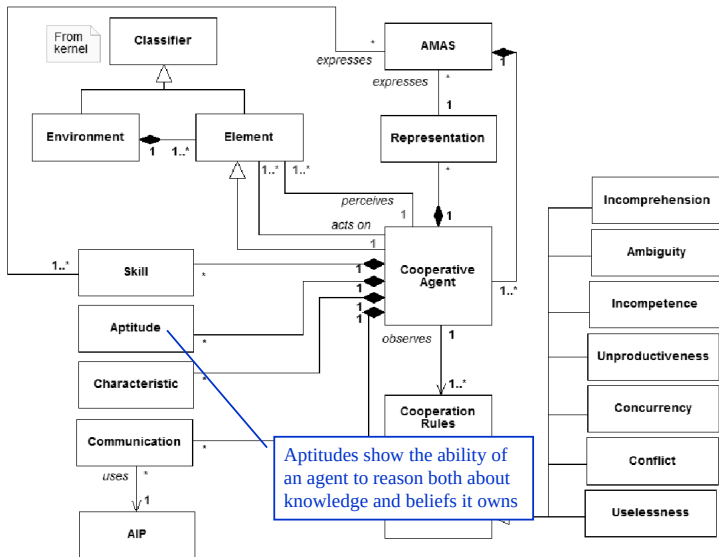
The ADELFE Meta-model



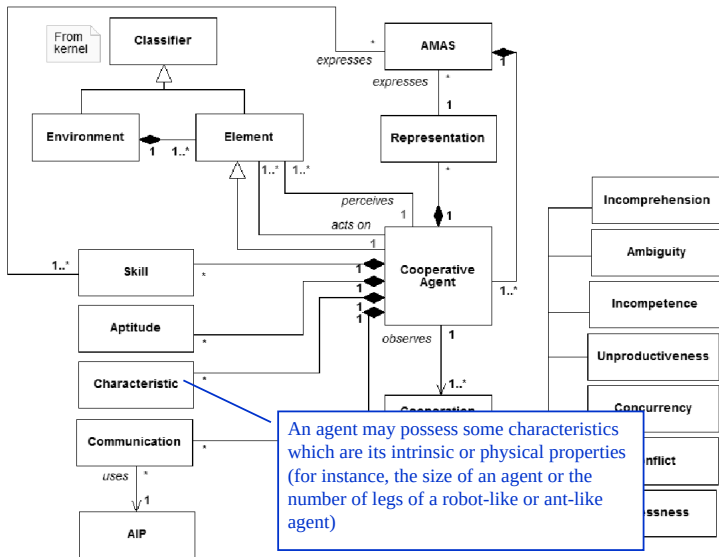
The ADELFE Meta-model



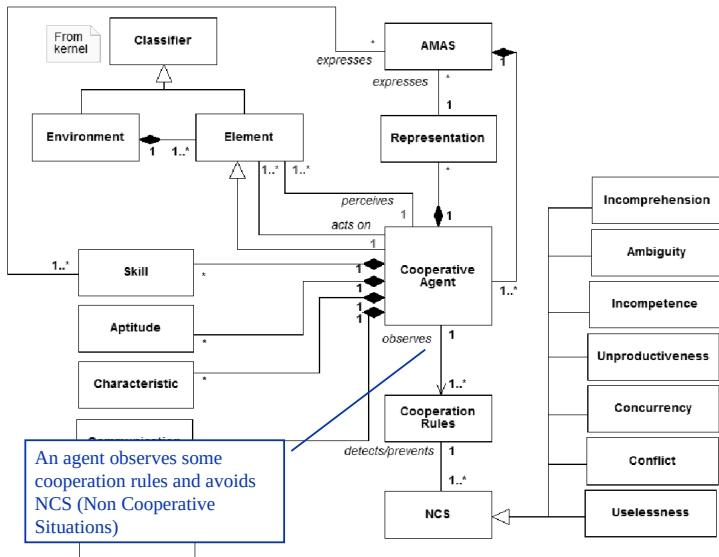
The ADELFE Meta-model



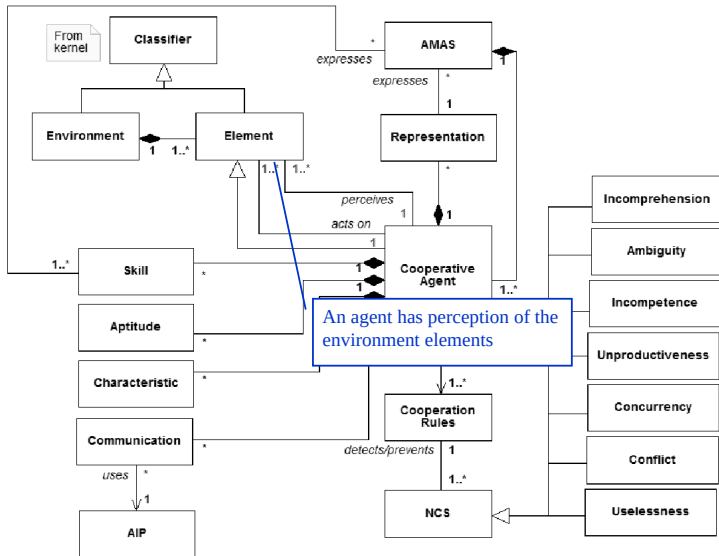
The ADELFE Meta-model



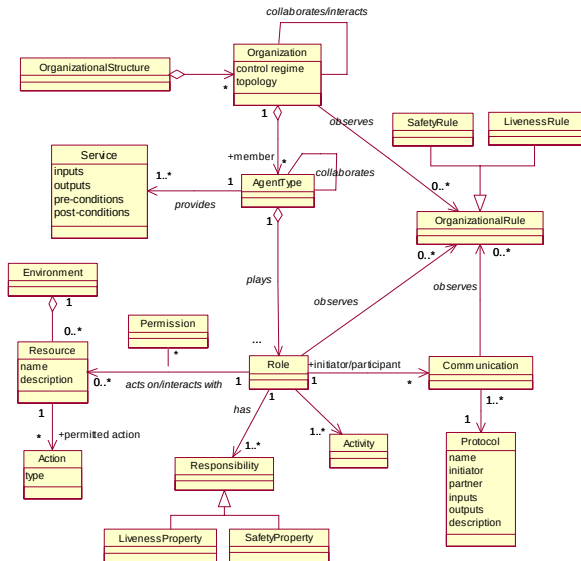
The ADELFE Meta-model



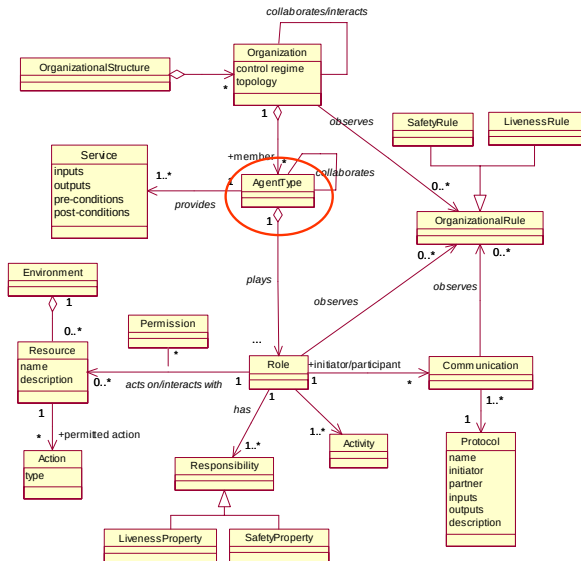
The ADELFE Meta-model



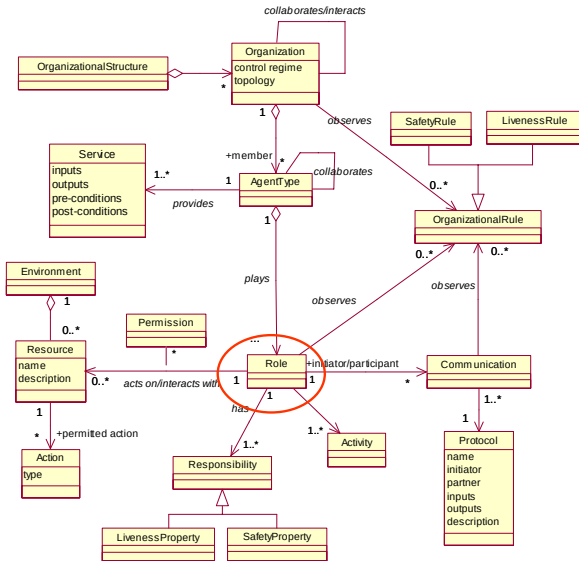
The Gaia Meta-model



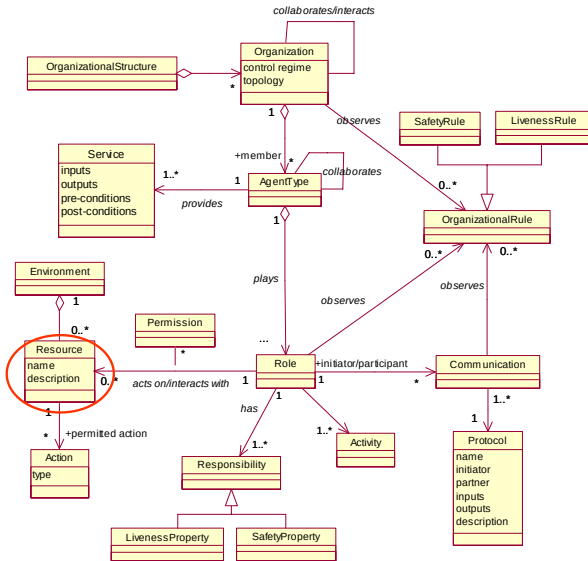
The Gaia Meta-model



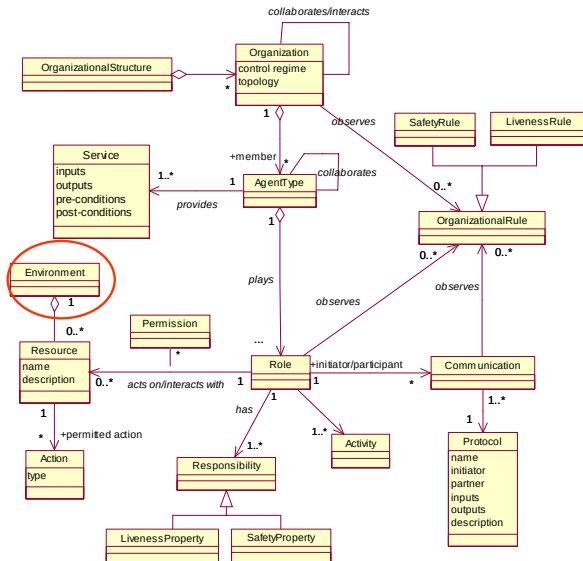
The Gaia Meta-model



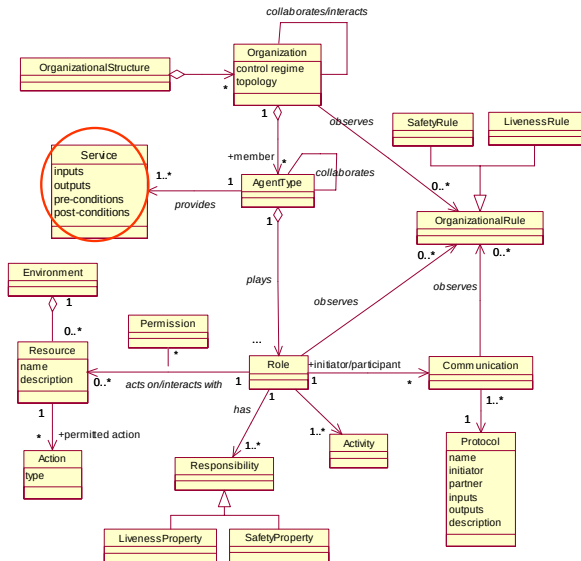
The Gaia Meta-model



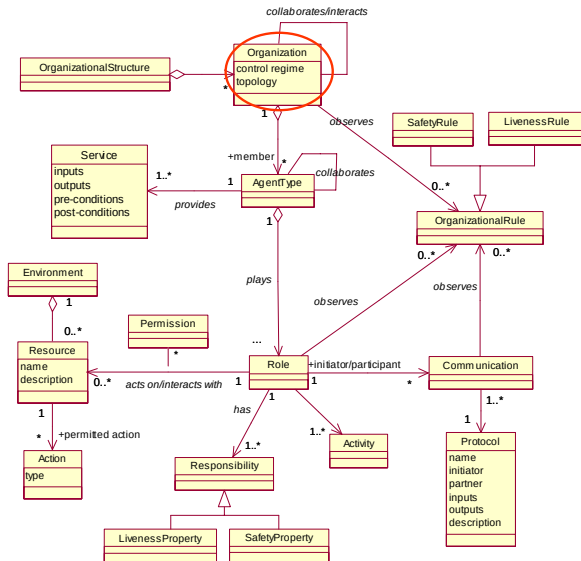
The Gaia Meta-model



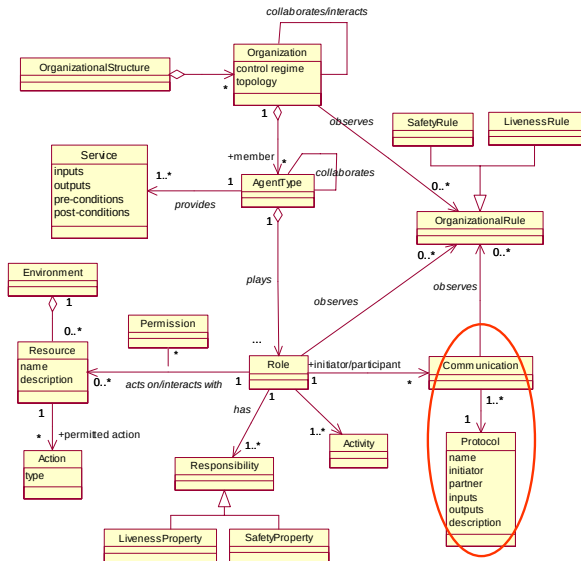
The Gaia Meta-model



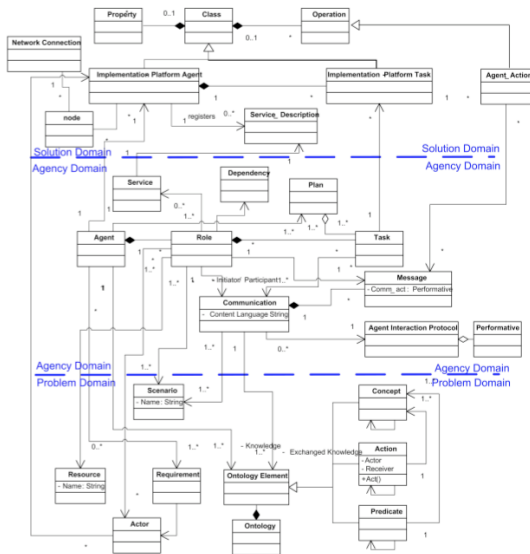
The Gaia Meta-model



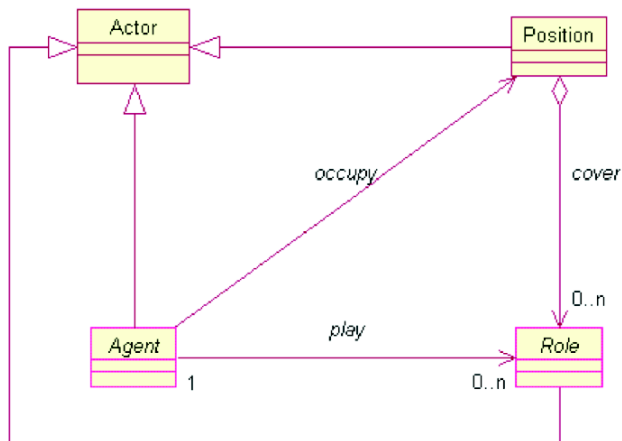
The Gaia Meta-model



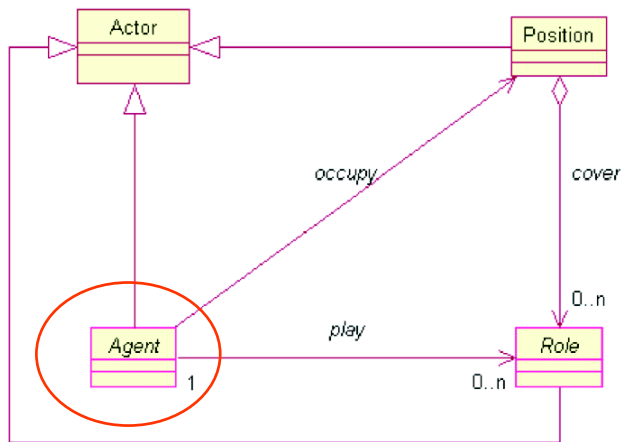
The PASSI Meta-model



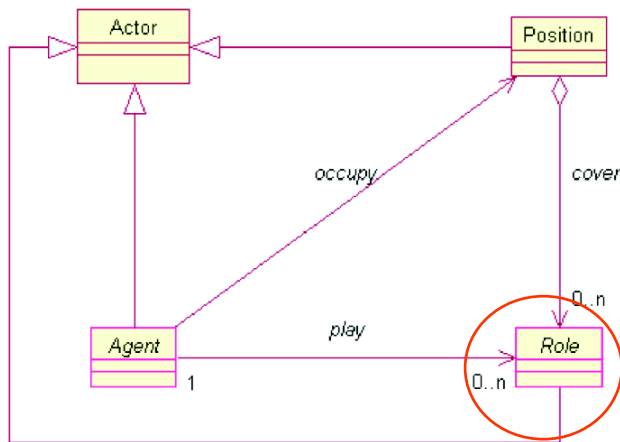
The Tropos Meta-model



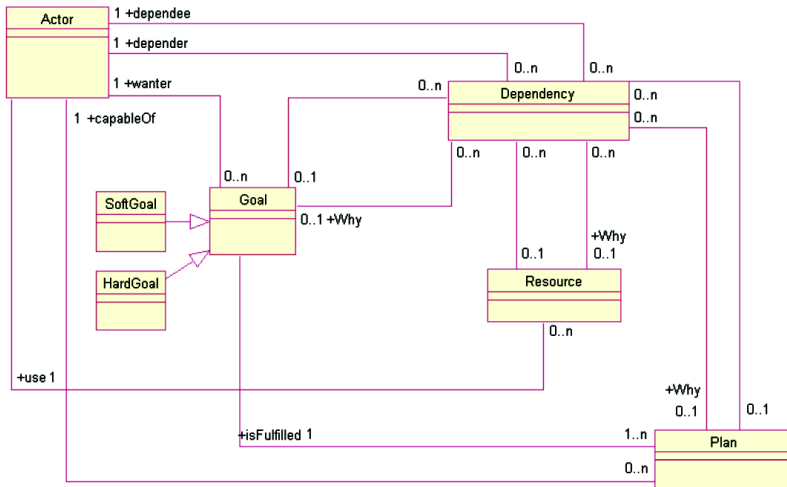
The Tropos Meta-model



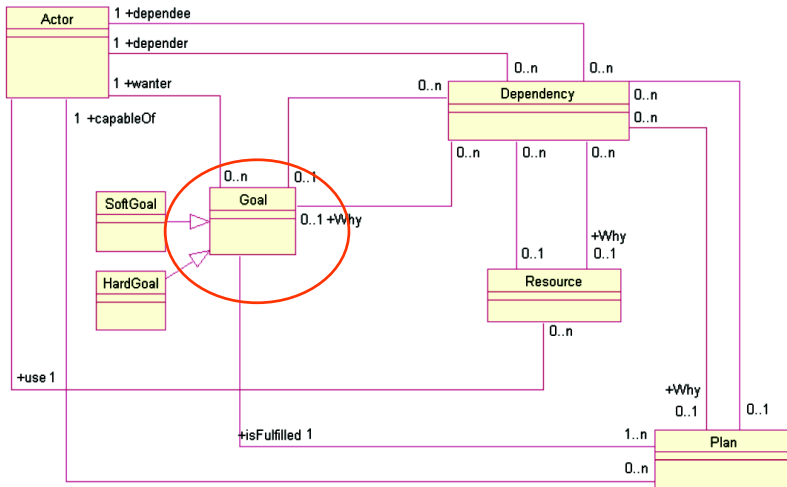
The Tropos Meta-model



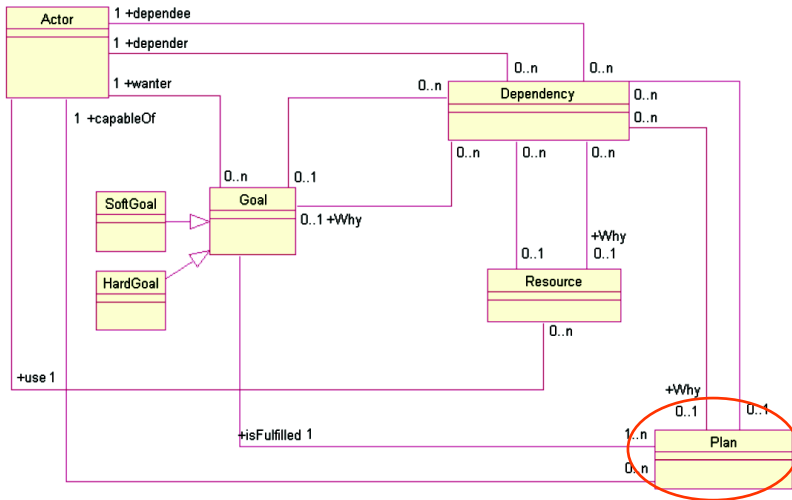
The Tropos Meta-model



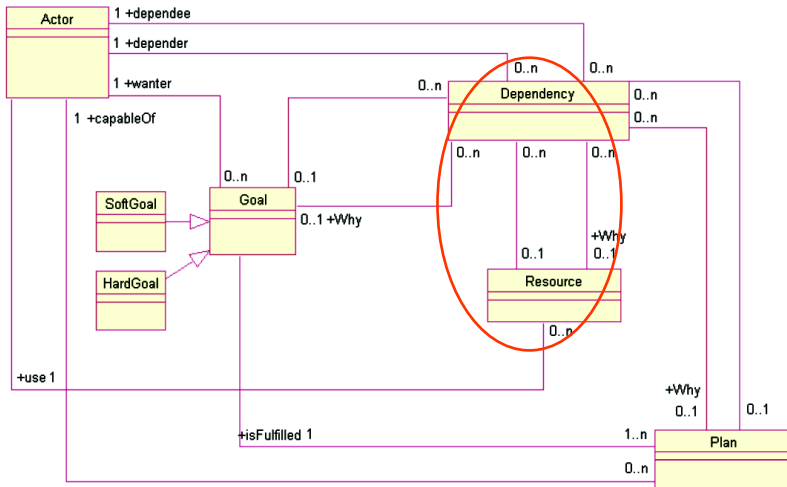
The Tropos Meta-model



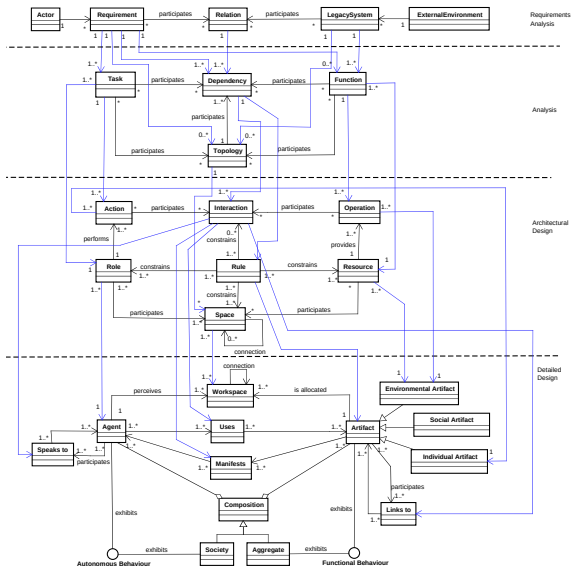
The Tropos Meta-model



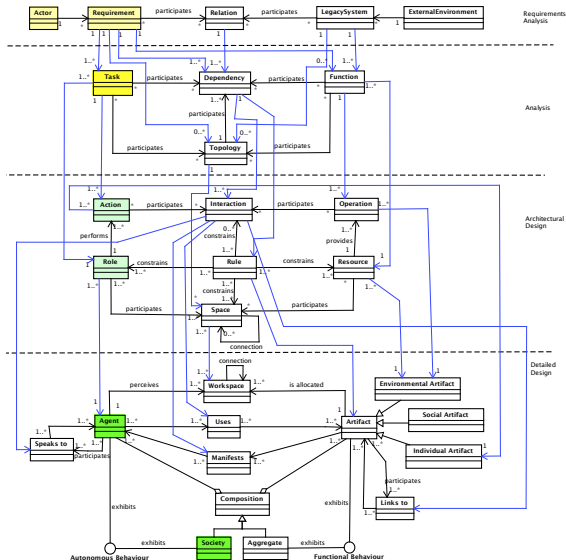
The Tropos Meta-model



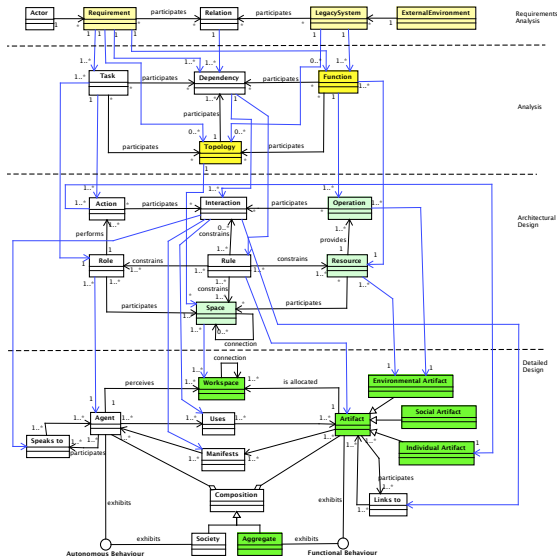
The SODA Meta-model



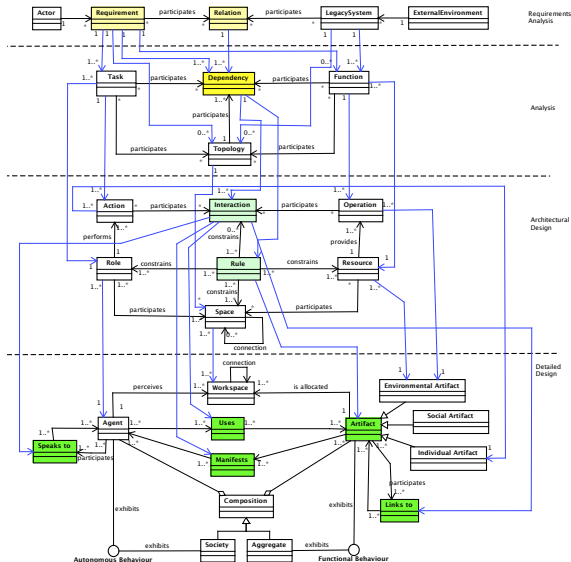
The SODA Meta-model



The SODA Meta-model



The SODA Meta-model



Outline

6 MAS Meta-model

7 Process Meta-model

- SPEM
- OPF & OPEN

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]

Outline

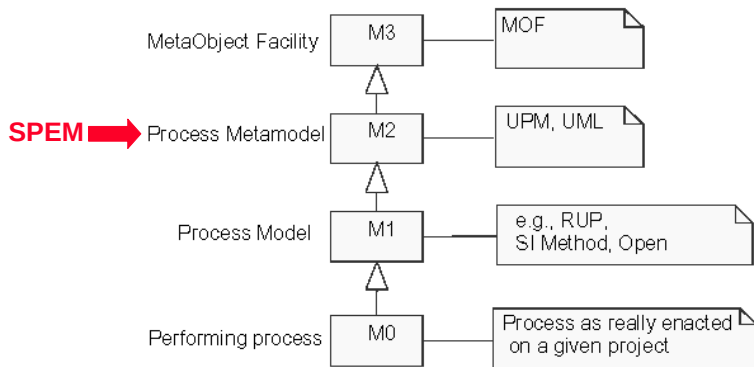
6 MAS Meta-model

7 Process Meta-model

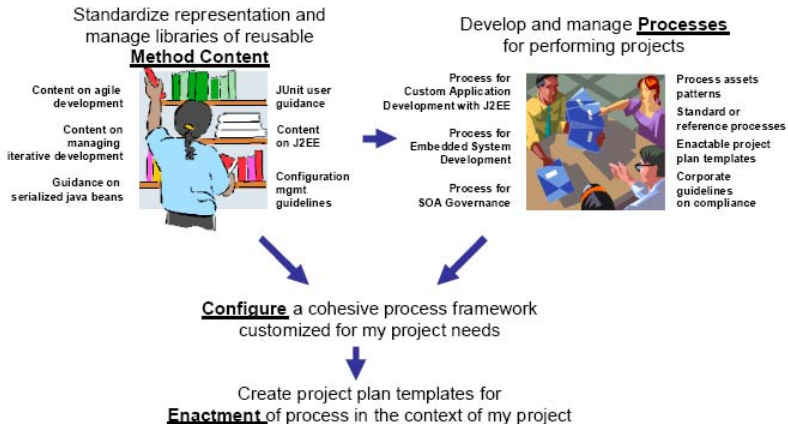
- SPEM
- OPF & OPEN

- 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

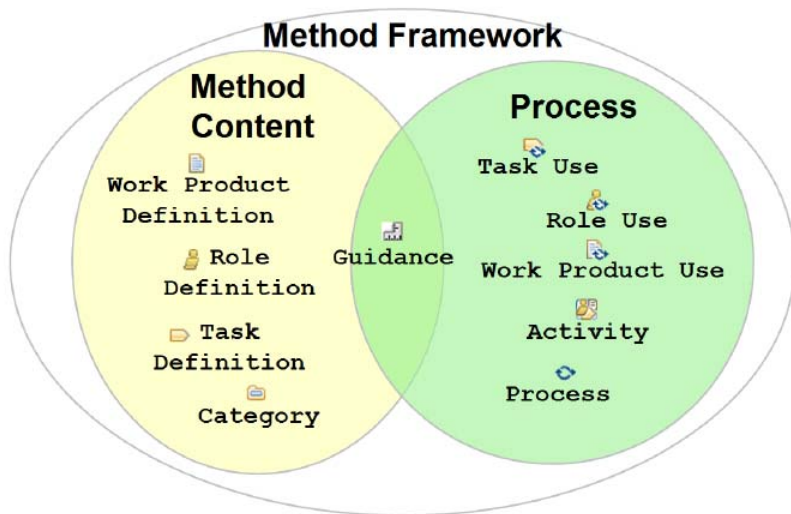


- The goals of SPEM are to:
 - ▶ support the representation of one specific development process
 - ▶ support the maintenance of several unrelated processes
 - ▶ provide process engineers with mechanisms to consistently and effectively manage whole families of related processes promoting process reusability



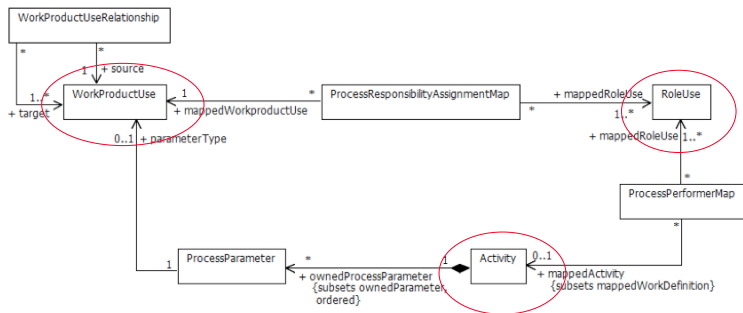
- Clear separation between
 - ▶ **Method Contents** — introduce the concepts to document and manage development processes through natural language description
 - ▶ **Processes** — defines a process model as a breakdown or decomposition of nested *Activities*, with the related *Roles* and input / output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

SPEM: Method Content and Process



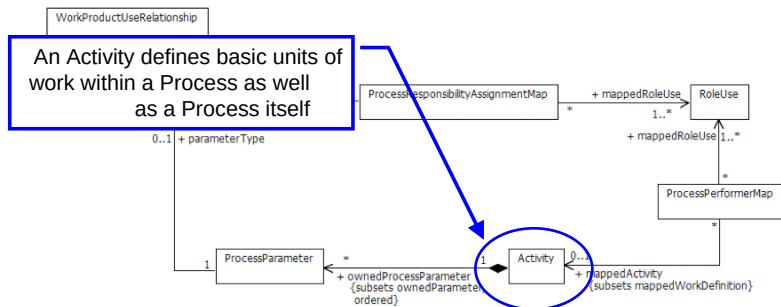
Roles, Activities & WorkProducts

- 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**



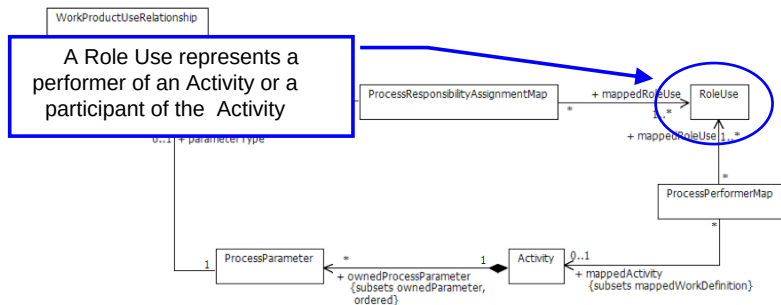
Roles, Activities & WorkProducts

- 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**



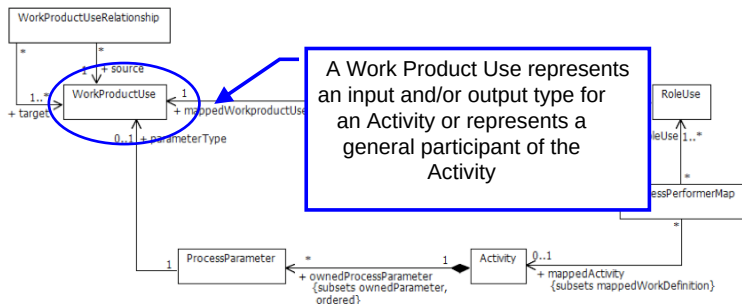
Roles, Activities & WorkProducts

- 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

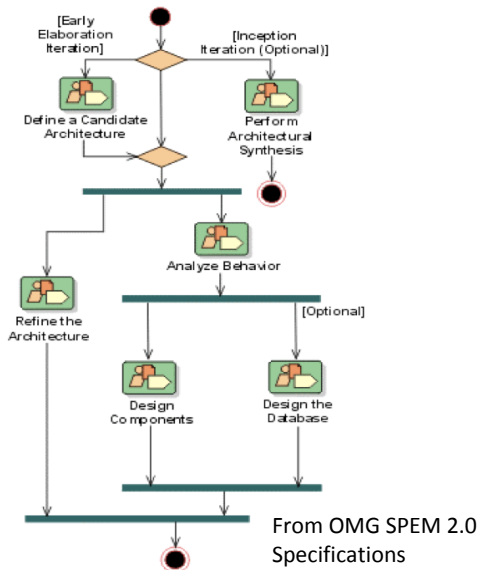
- A software development process is seen as a collaboration between abstract active entities called **process roles** that perform operations called **activities** on concrete, tangible entities called **work products**



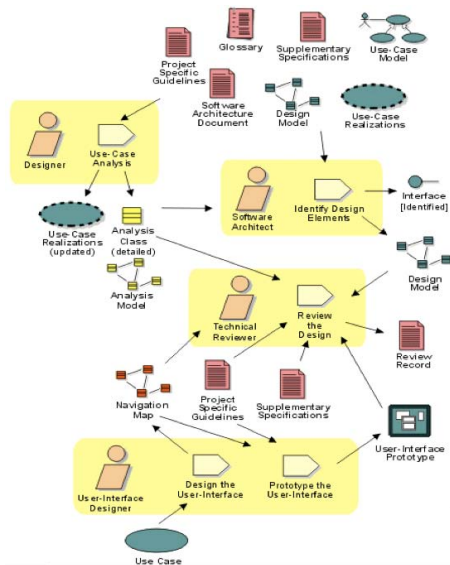
SPEM Notation

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: WorkFlow Diagram

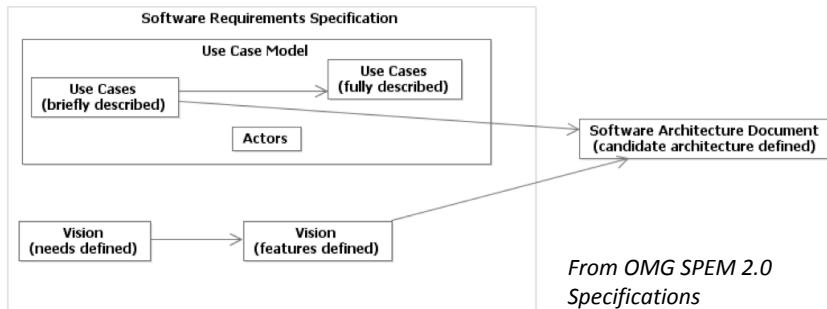


SPEM: Activity Details Diagram

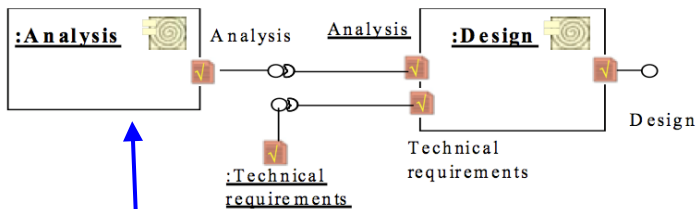


*From OMG SPEM 2.0
Specifications*

SPEM: Work Product Dependency Diagram



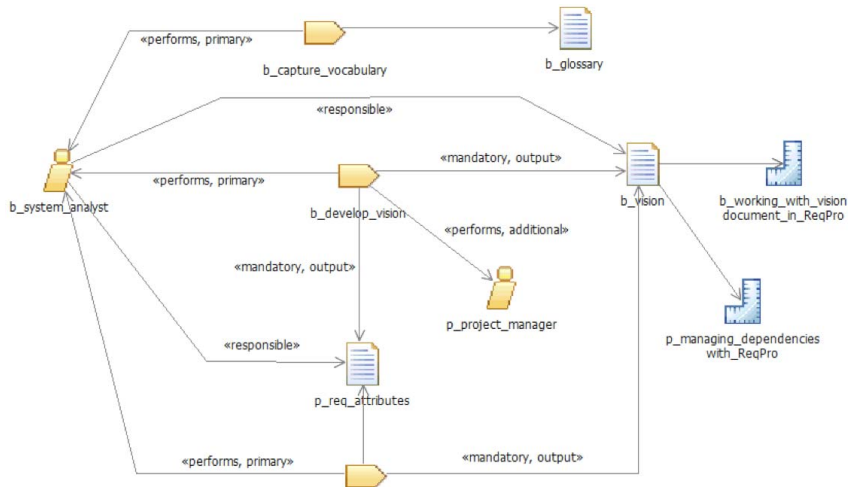
SPEM: Process Component Diagram



A Process Component contains exactly one Process represented by an Activity, and defines a set of Work Product Ports that define the inputs and outputs for a Process Component.

From OMG SPEM 2.0 Specifications

SPEM: Class Diagram



From OMG SPEM 2.0 Specifications

Outline

6 MAS Meta-model

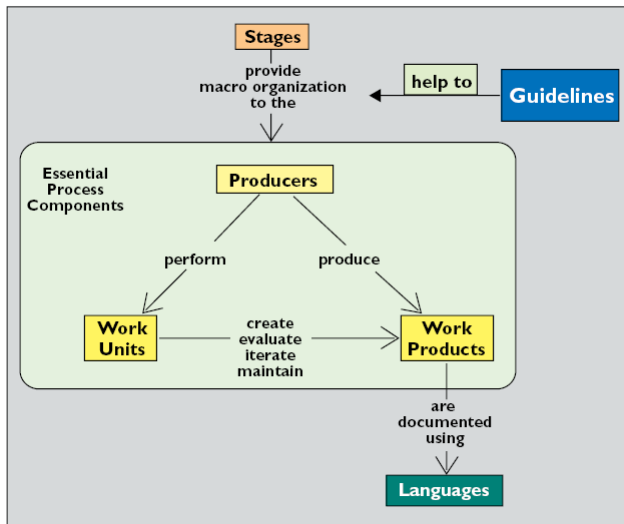
7 Process Meta-model

- SPEM
- OPF & OPEN

OPEN

- Object-oriented Process, Environment, and Notation (OPEN) [OPEN Working Group,] is a full lifecycle, process-focussed, methodological approach that was designed for the development of software intensive applications
- OPEN is defined as a process framework, known as the OPF (OPEN Process Framework)
- This is a process meta-model from which can be generated an organisationally-specific process (instance)
- Each of these process instances is created by choosing specific Activities, Tasks and Techniques (three of the major metalevel classes) and specific configurations
- The definition of process include not only descriptions of phases, activities, tasks, and techniques but issues associated with human resources, technology, and the life-cycle model to be used

Metalevel Classes [Henderson-Sellers, 2003]



Work Product & Language & Producer

- A **work product** is any significant thing of value (e.g., document, diagram, model, class, application) that is developed during a project
- A **language** is the medium used to document a work product. Use case and object models are written using a modelling language such as the Unified Modeling Language (UML) or the OPEN Modelling Language (OML)
- A **producer** is anything that produces (i.e., creates, evaluates, iterates, or maintains), either directly or indirectly, versions of one or more work products. The OPF distinguishes between those direct producers (persons as well as roles played by the people and tools that they use) and indirect producers (teams of people, organisations and endeavours)

Work Unit

- A **work unit** is a functionally cohesive operation that is performed by a producer during an endeavour and that is reified as an object to provide flexibility during instantiation and tailoring of a process
- The OPF provides the following predefined classes of work units:
 - ▶ Task – functionally cohesive operation that is performed by a direct producer. A task results in the creation, modification, or evaluation of a version of one or more work products
 - ▶ Technique – describes in full detail how a task are to be done
 - ▶ Activity – cohesive collection of workflows that produce a related set of work products. Activities in OPEN are coarse granular descriptions of what needs to be done

Stage

- A **stage** is a formally identified and managed duration or a point in time, and it provides a macro organisation to the work units
- The OPF contains the following predefined classes of stage:

Cycle — there are several types of cycle e.g. lifecycle

Phase — consisting of a sequence of one or more related builds, releases and deployments

Workflow — a sequence of contiguous task performances whereby producers collaborate to produce a work product

Build — a stage describing a chunk of time during which tasks are undertaken

Release — a stage which occurs less frequently than a build. In it, the contents of a build are released by the development organisation to another organisation

Deployment — occurs when the user not only receives the product but also, probably experimentally, puts it into service for on-site evaluation

Milestone — is a kind of Stage with no duration. It marks an event occurring

What I can do with process meta-models?

- Process meta-models are useful for representing methodologies and therefore they are essential for creating a new one
- Assembling methodologies by reusing fragments of existing ones in order to perfectly manage a specific problem is the approach proposed by (situational) method engineering
- We will now:
 - ▶ introduce method engineering as a way for creating a new methodology

Part IV

Situational Method Engineering

Outline

8 Method Engineering in traditional SE

- Method Fragment Representation
- Method Fragment Assembly

9 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

Methodologies

- As for software development, individual methodologies are often created with specific purposes in mind [Henderson-Sellers, 2005]
 - ▶ particular domains
 - ▶ particular segments of the lifecycle
- Users often make the assumption that a methodology is not in fact constrained but, rather, is universally applicable
- This can easily lead to *methodology failure*, and to the total rejection of methodological thinking by software development organisation
- The creation of a single universally applicable methodology is an unattainable goal
- We should ask ourselves how could we create a methodological environment in which the various demands of different software developers might be satisfied altogether

Method Engineering

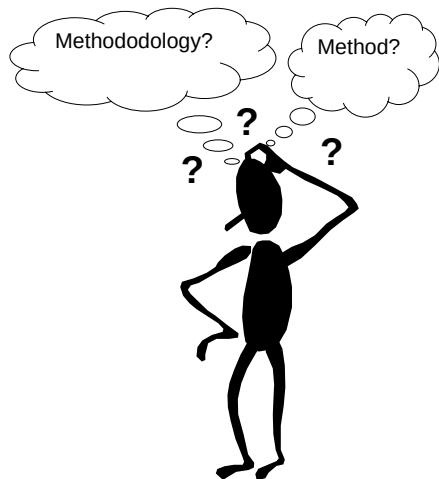
Method Engineering [Brinkkemper, 1996]

Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems

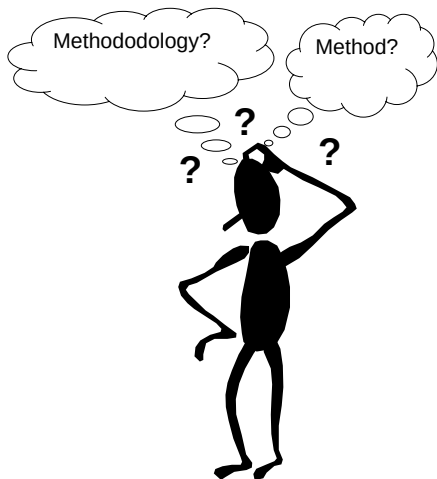
Different definitions [Brinkkemper, 1996]

- **Method**: an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products
- **Methodology**: the systematic description, explanation and evaluation of all aspects of methodical information systems development
- ... these definitions are different from the definitions we have given in the previous Section...

Method & Methodology

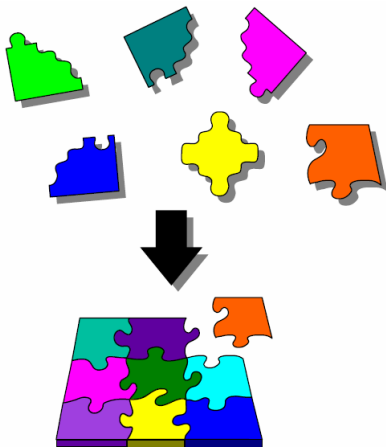


Method & Methodology

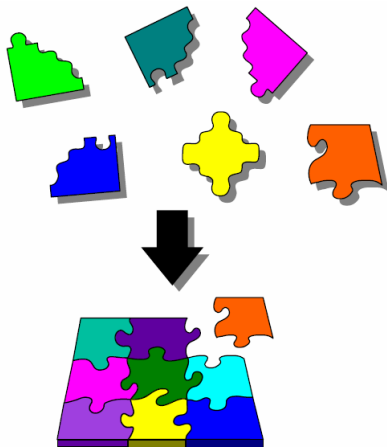


- All the concepts and ideas used in the Method Engineering are also applicable to our definitions of methodology and method
- Method Engineering tries to model methodological processes and products by isolating conceptual *method fragments*
- These fragments act as methodological “building blocks”

Method Engineering: Motivations



Method Engineering: Motivations



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

Method Engineering: Concerns

- Similarly as software engineering is concerned with all aspects of software production, 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

Meta-Modelling Techniques

- The design and evaluation of methods and tools require special purpose specification techniques, called meta-modelling techniques, for describing their procedural and representational capabilities.
- Issues are:
 - ▶ what are the proper constructs for meta-modelling?
 - ▶ what perspectives of meta-models should be distinguished?
 - ▶ is there an optimal technique for meta-modelling, or is the adequacy of the technique related to the purpose of the investigation?

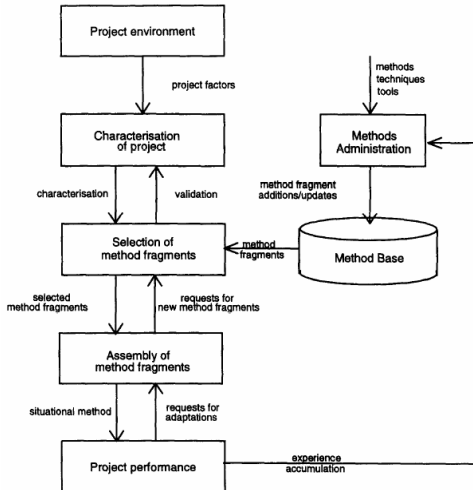
Tool Interoperability

- A lot of tools that only cover part of the development life-cycle exist
- So the system development practice has to deal with the proper integration of tools at hand
- Open problems are related to the overall architecture of the integrated tools
 - ▶ should this be based on the storage structure (i.e. the repository) in a data-integration architecture, or on a communication structure between the functional components in a control-integration architecture?

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]



Situational Method Engineering I

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

Situational Method Engineering II

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

Method Fragments

- [Brinkkemper et al., 1999] classifies method fragments according to three different dimensions

Perspective — product and process

Abstraction level — conceptual and technical

Layer of granularity — five different level

Perspective

- Perspective distinguishes product fragments and process fragments
 - ▶ **product fragments** model the structures of the products (deliverables, diagrams, tables, models) of a systems development method
 - ▶ **process fragments** are models of the development process. Process fragments can be either high-level project strategies, called method outlines, or more detailed procedures to support the application of specification techniques

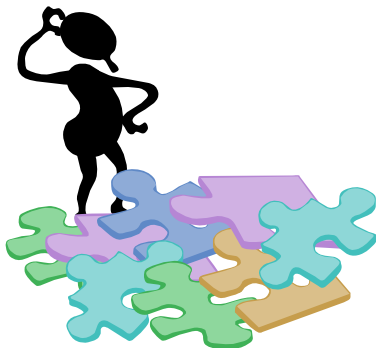
Abstraction level

- Abstraction level distinguishes conceptual level and technical level
 - ▶ method fragments at the conceptual level are descriptions of information systems development methods or part of them
 - ▶ technical method fragments are implementable specifications of the operational parts of a method, i.e. the tools
- Some conceptual fragments are to be supported by tools, and must therefore be accompanied by corresponding technical fragments
- One conceptual method fragment can be related to several external and technical method fragments

Layer of granularity

- A method fragment can reside on one of five possible granularity layers
 - Method** — addressing the complete method for developing the information system
 - Stage** — addressing a segment of the life-cycle of the information system
 - Model** — addressing a perspective of the information system
 - Diagram** — addressing the representation of a view of a Model layer method fragment
 - Concept** — addressing the concepts and associations of the method fragments on the Diagram layer, as well as the manipulations defined on them

And Now?



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

Outline

8 Method Engineering in traditional SE

- Method Fragment Representation
- Method Fragment Assembly

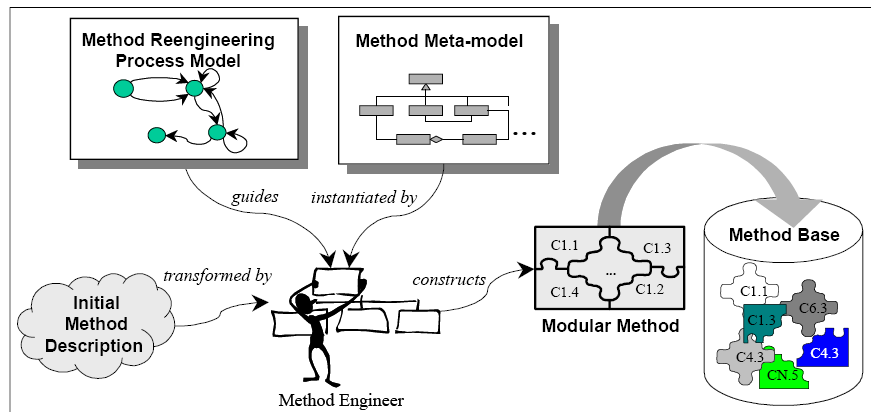
9 Method Engineering in AOSE

- 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 Fragments Representation

- In the last decade a lot of work has been done in the context of Method Engineering
- However this technique is not entered in the mainstream of the Software Engineering
 - ▶ there is no consensus in academia nor significative industrial effort towards a standardisation
 - ▶ each research group created its own method fragment representation
- Here we briefly introduce the work of Ralyté and Rolland, that inspired the work of the FIPA group in the context of AOSE
- The OPEN framework by Brian Henderson-Sellers will also be presented because it is actually the largest existing framework

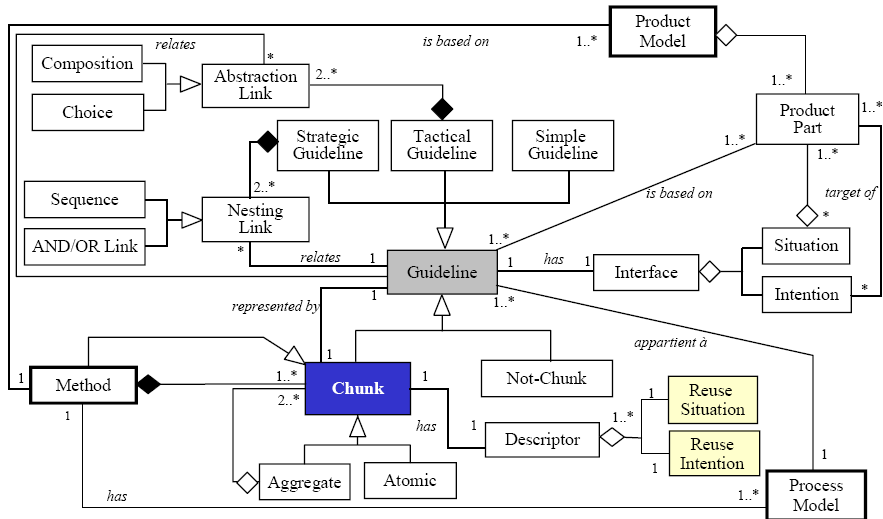
Method Reengineering [Ralyté and Rolland, 2001a]



Method Reengineering

- In this approach Ralyté and Rolland adopt the notion of *method chunk* [Ralyté and Rolland, 2001a]
- A method chunk ensures a tight coupling of some process part and its related product part. It is a coherent module and any method is viewed as a set of loosely coupled method chunks expressed at different levels of granularity
- Their method meta-model is presented in the next slide. . .

The Method Meta-model [Ralyté and Rolland, 2001a]



Method Meta-model

- According to this meta-model a method is also viewed as a **method chunk** of the highest level of granularity
- The definition of the method chunk is *process-driven* in the sense that a chunk is based on the decomposition of the method process model into reusable guidelines
- Thus, the core of a method chunk is its guideline to which are attached the associated product parts needed to perform the process encapsulated in this guideline
- A guideline embodies method knowledge to guide the application engineer in achieving an intention in a given situation
- Therefore, the guideline has an interface, which describes the conditions of its applicability (the situation) and a body providing guidance to achieve the intention, i.e. to proceed in the construction of the target product

Guidelines

- The body of the guideline details how to apply the chunk to achieve the intention
- The interface of the guideline is also the interface of the corresponding method chunk
- Guidelines in different methods have different contents, formality, granularity, etc.
- In order to capture this variety, the authors identify three types of guidelines: simple, tactical and strategic

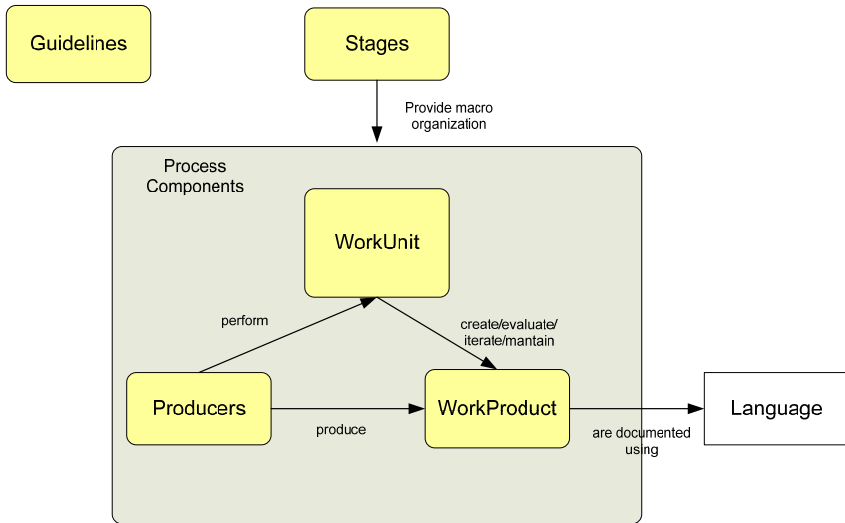
Guidelines Types

- A *simple guideline* may have an informal content advising on how to proceed to handle the situation in a narrative form. It can be more structured comprising an executable plan of actions leading to some transformation of the product under construction
- A *tactical guideline* is a complex guideline, which uses a tree structure to relate its sub-guidelines one with the others
- A *strategic guideline* is a complex guideline called a map which uses a graph structure to relate its sub-guidelines. Each sub-guideline belongs to one of the three types of guidelines. A strategic guideline provides a strategic view of the development process telling which intention can be achieved following which strategy

OPF method fragment

- It is part of existing methodologies and used to construct new ones
- It is generated and stored in a repository with all its guidelines basing on OPF Metamodel
- The Metamodel is composed of five main metaclasses
- Each metaclass produces a method fragment

The OPF Meta-model



The OPF Meta-model

- Main elements of OPF Metamodel

- ▶ WorkUnit, operations (Activities, Tasks and Techniques) performed by Producers during a development process in order to produce a specific WorkProduct
- ▶ Activities and Tasks describe what is to be done whereas
- ▶ Techniques describe how to do a portion of work
- ▶ WorkProduct, the artefact
- ▶ Producers, the stakeholder performing a work unit
- ▶ Stage, provides the organisation of process in terms of phases and lifecycles
- ▶ Guideline, useful to instantiate the metamodel elements, to create method components and to create the method itself

Outline

8 Method Engineering in traditional SE

- Method Fragment Representation
- **Method Fragment Assembly**

9 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 Assembly

- In the last decade a lots of work is done in the context of Method Assembly
- This leads to a proliferation of different techniques for Method Assembly, and each of them adopts a peculiar representation and phases
- Here we briefly show some rules from Brinkkemper, the Method Assembly techniques by Ralyté and Rolland and the OPEN by Brian Henderson-Sellers

Brinkkemper's Rules I

[Brinkkemper et al., 1999] introduce several general rules for method assembly

- Rule 1 — At least one concept, association or property should be newly introduced to each method fragment to be assembled, i.e. a method fragment to be assembled should not be a subset of another
- Rule 2 — We should have at least one concept and/or association that connects between two method fragments to be assembled
- Rule 3 — If we add new concepts, they should be connectors to both of the assembled method fragments
- Rule 4 — If we add new associations, the two method fragments to be assembled should participate in them
- Rule 5 — There are no isolated parts in the resulting method fragments

Brinkkemper's Rules II

- Rule 6 — There are no concepts which have the same name and which have the different occurrences in a method description
- Rule 7 — The activity of identifying the added concepts and associations that are newly introduced for method assembly should be performed after their associated concepts are identified
- Rule 8 — Let A and B be the two method fragments to be assembled, and C the new method fragment. In C, we should have at least one product which is the output of A and which is the input of B, or the other way round
- Rule 9 — Each product fragment should be produced by a “corresponding” process fragment
- Rule 10 — Suppose a product fragment has been assembled. The process fragment that produces this product fragment consists of the process fragments that produce the components of the product fragment

Brinkkemper's Rules III

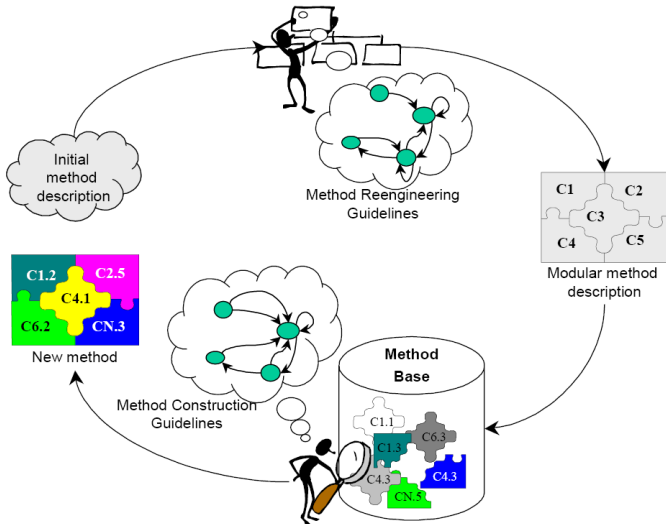
- Rule 11 — A technical method fragment should supports a conceptual method fragment
- Rule 12 — If an association exists between two product fragments, there should exist at least one association between their respective components
- Rule 13 — There are no “meaningless” associations in product fragments, i.e. every association is “meaningful” in the sense that it can semantically consistently connect to specific concepts

A Different Approach

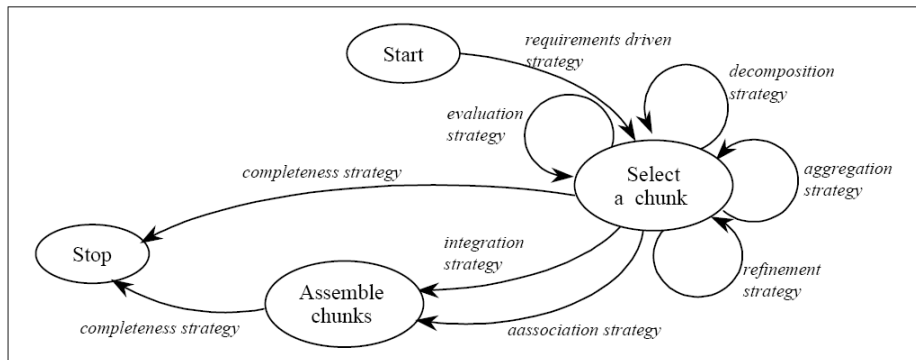
- Jolita Ralyté and Colette Rolland have proposed an approach for assembling method chunks
- In particular they distinguished two different assembly strategies:
 - ▶ association – The assembly process by association consists in connecting chunks such that the first one produces a product which is the source of the second chunk
 - ▶ integration – The assembly process by integration consists in identifying the common elements in the chunks product and process models and merging them

Assembly Based Method Engineering

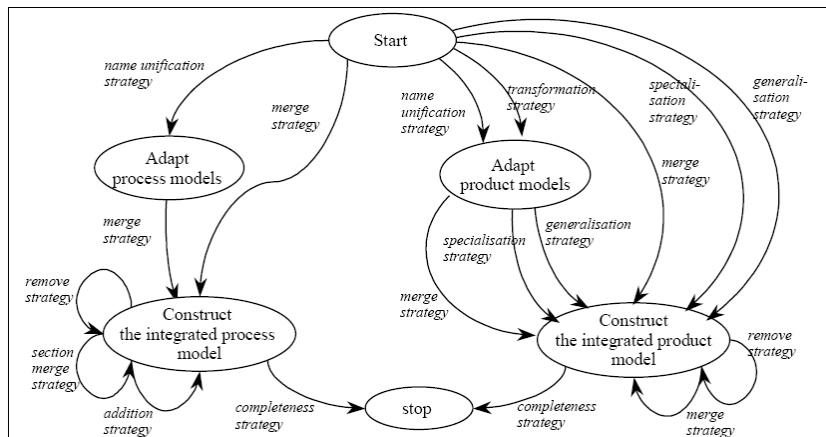
[Ralyté and Rolland, 2001a]



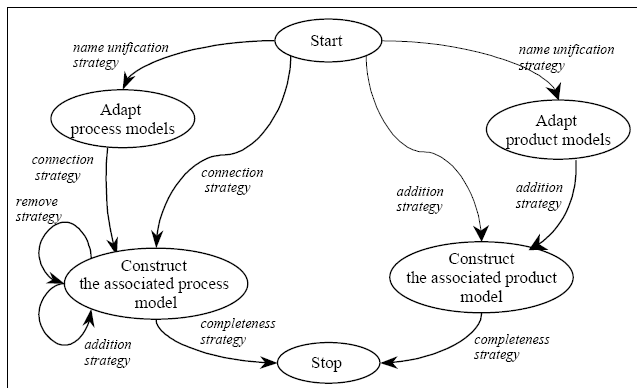
Assembly Map [Ralyté and Rolland, 2001b]



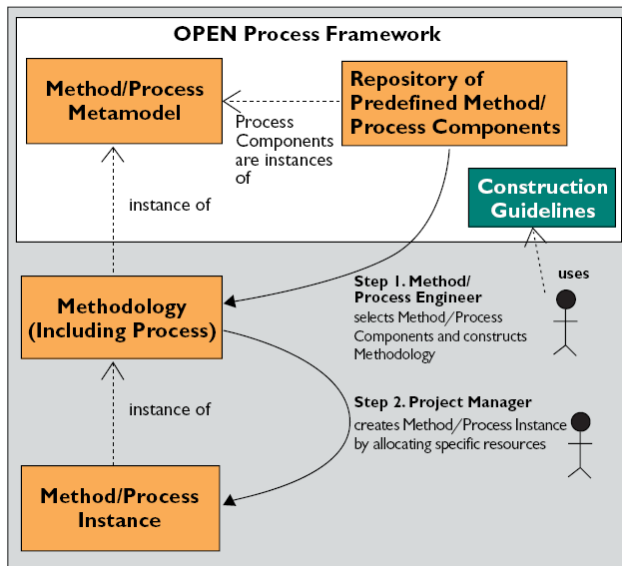
Integration Map [Ralyté and Rolland, 2001b]



Association Map [Ralyté and Rolland, 2001b]



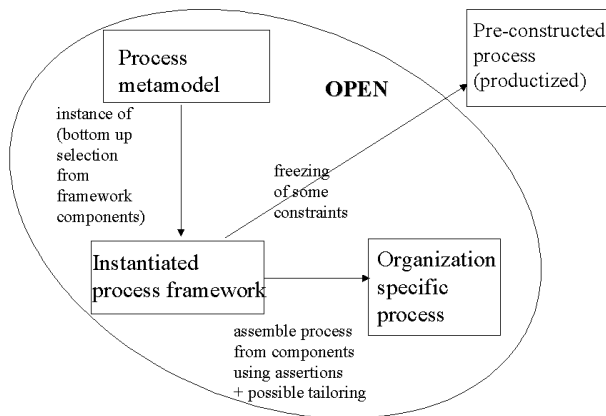
OPEN Process Framework [Henderson-Sellers, 2003]



Usage Guidelines

- The core of the Method Assembly in OPF are *usage guidelines* covering:
 - ▶ instantiating the class library to produce actual process components
 - ▶ choosing the best process components
 - ▶ tailoring the fine detail inside the chosen process components
 - ▶ extending the existing class library of predefined process components

OPEN Process Framework [OPEN Working Group,]



It is even possible to take sets of constraints to create “pre-tailored” styles of OPEN process

Process Construction Guidelines

- A process construction guideline is a usage guideline intended to help process engineers instantiate the development process framework and then select the best component instances in order to create the process itself
- Specifically, it will provide guidance concerning how to:
 - ▶ select the work products to develop
 - ▶ select the producers (e.g., roles, teams, and tools) to develop these work products
 - ▶ select the work units to perform
 - ▶ how to allocate tasks and associated techniques to the producers
 - ▶ how to group the tasks into workflows, activities
 - ▶ select stages of development that will provide an overall organisation to these work units

Matrix

- OPEN recommends construction of a number of matrices linking, for example, Activities with Tasks and Tasks with Techniques
- The possibility values in these matrices indicate the likelihood of the effectiveness of each individual pair
- These values should be tailored to a specific organisation or a specific project
- Typically a matrix should have five levels of evaluation: mandatory, recommended, optional, discouraged, forbidden
- However a two levels evaluation matrix (use/do not use) gives good results

Matrix Example [Henderson-Sellers, 2003]

Task	Activity					
	1	2	3	4	5	6
Code		x				
Construct the object model		x				x
Develop and implement resource allocation plan						
develop iteration plan	x					
develop timebox plan	x					
set up metrics collection program	x					
specify quality goals	x					
Evaluate quality			x	x		x
Identify CIRTs (Class, Instance, Role, or Type)		x				
Map roles onto classes		x(OOP)				
Test			x	x	x	
Write manuals and other documentation			x	x	x	x
key 1. Project planning 2. Modeling and Implementation: OO analysis, design, programming 3. Verification and validation 4. User review 5. Consolidation 6. Evaluation						

Tailoring Guidelines

- Once the process framework has been instantiated and placed into effect, one typically finds that one needs to perform some fine-tuning by tailoring the instantiated process components as lessons are learned during development
- Tailoring guidelines are usage guidelines intended to help process engineers tailor the instantiated process components

Extension Guidelines

- No class library can ever be totally complete
- Because of the large differences between development projects, new classes of process components will eventually be needed
- Also, software engineering is an evolving discipline, and new process components will need to be added as the field advance
- A process framework should therefore come with extension guidelines, whereby an extension guideline is a usage guideline intended to help the process engineer extend the existing development process framework by adding new classes of process components

Outline

8 Method Engineering in traditional SE

- Method Fragment Representation
- Method Fragment Assembly

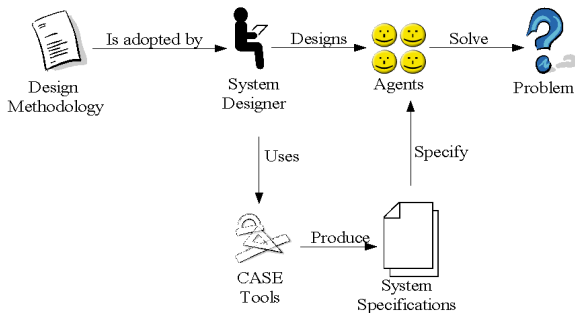
9 Method Engineering in AOSE

- 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

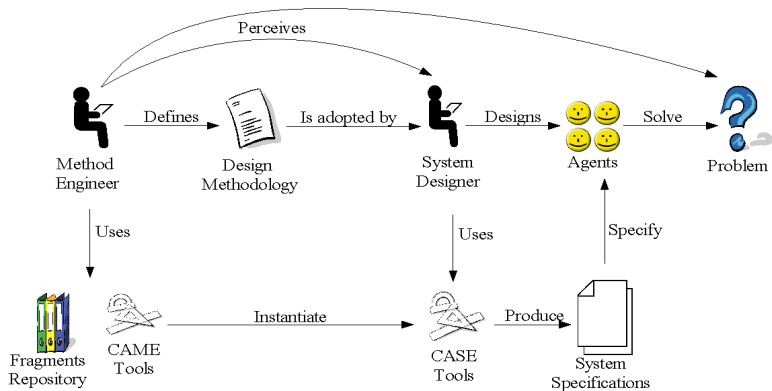
Agent Oriented Situational Method Engineering

- 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)

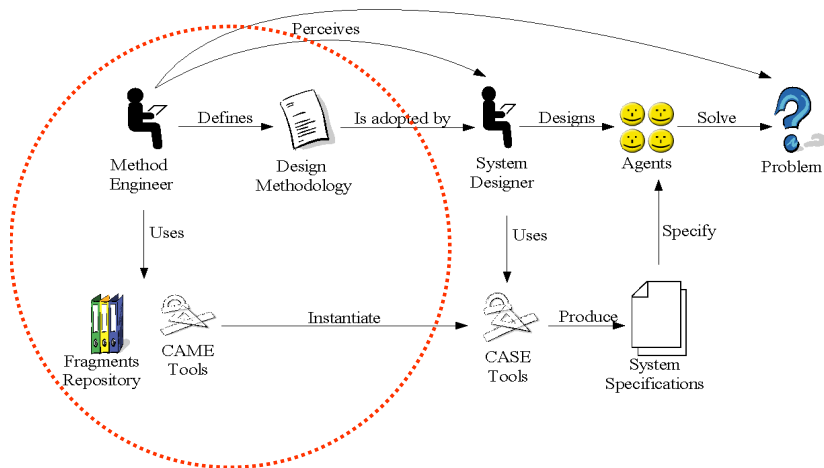
The normal agent development process



Situational Method Engineering



Situational Method Engineering



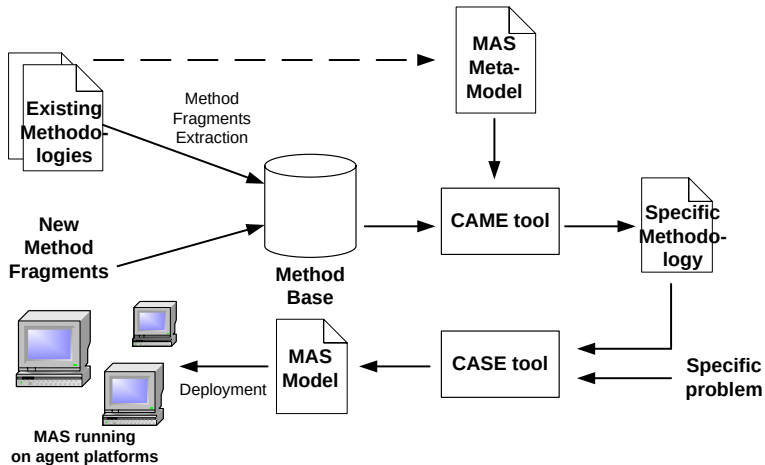
Adopting Situational Method Engineering

- 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?)

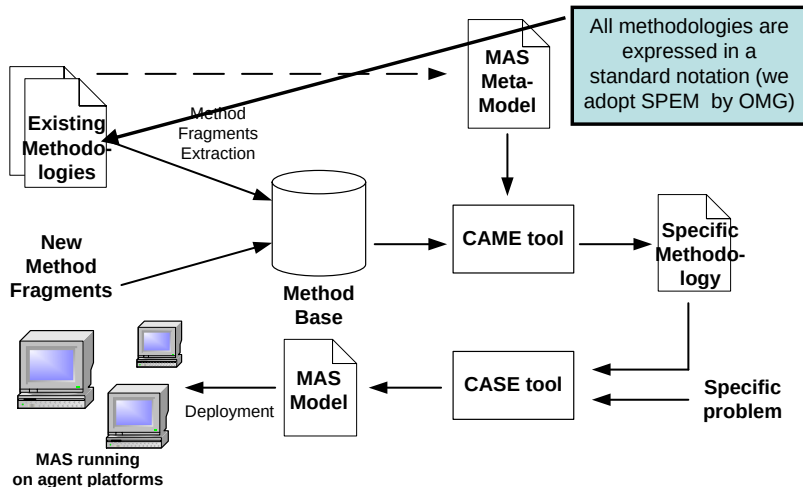
CAME Tool

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

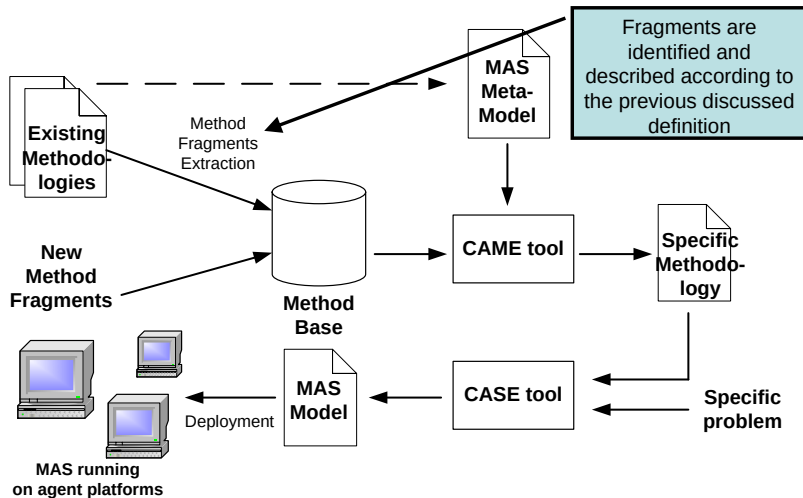
The new process production



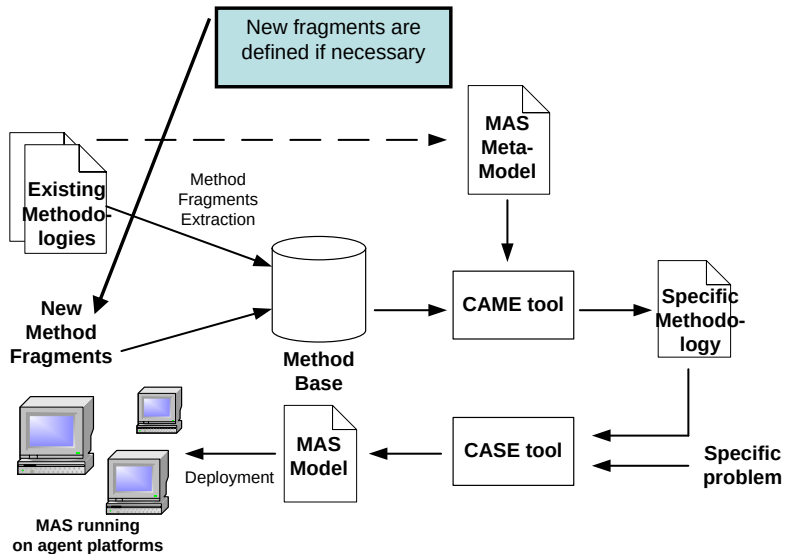
The new process production



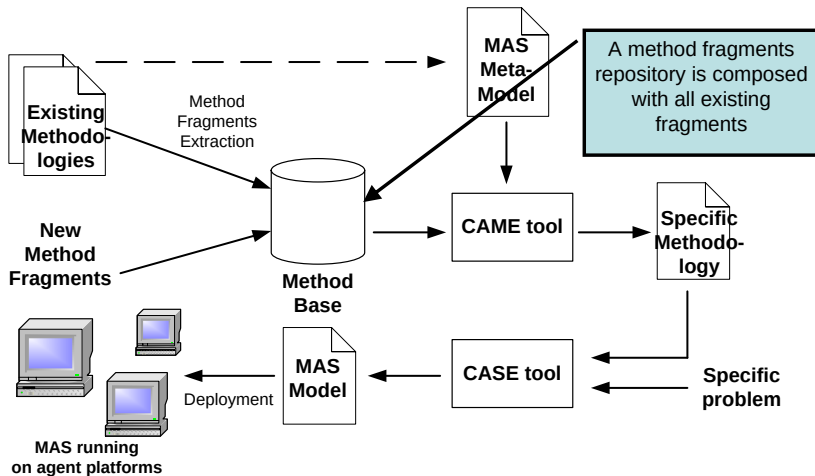
The new process production



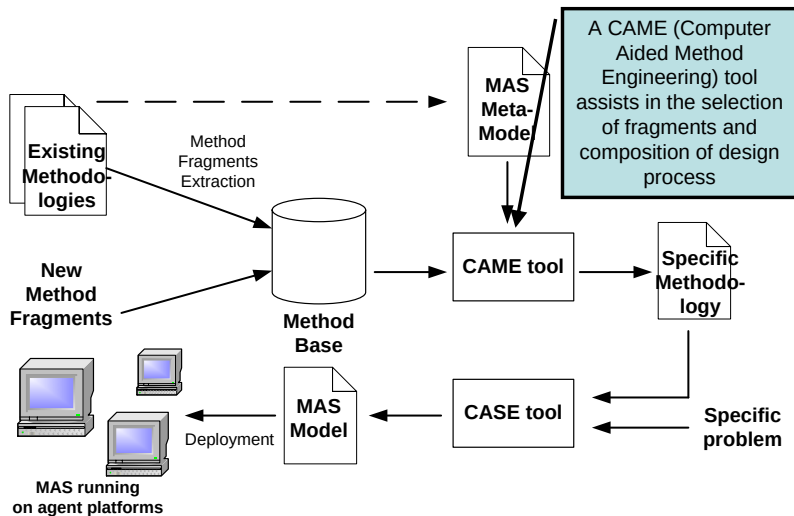
The new process production



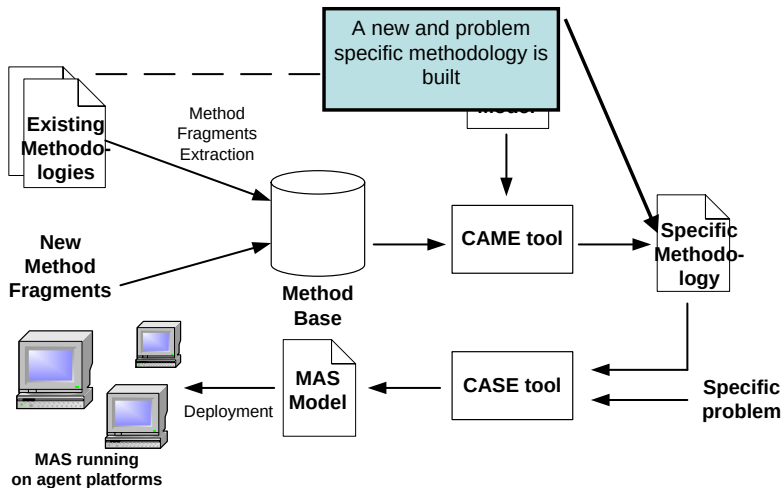
The new process production



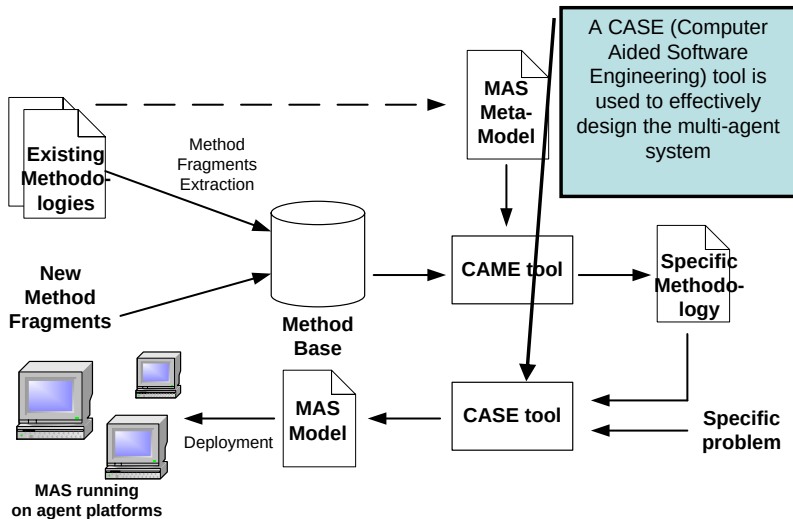
The new process production



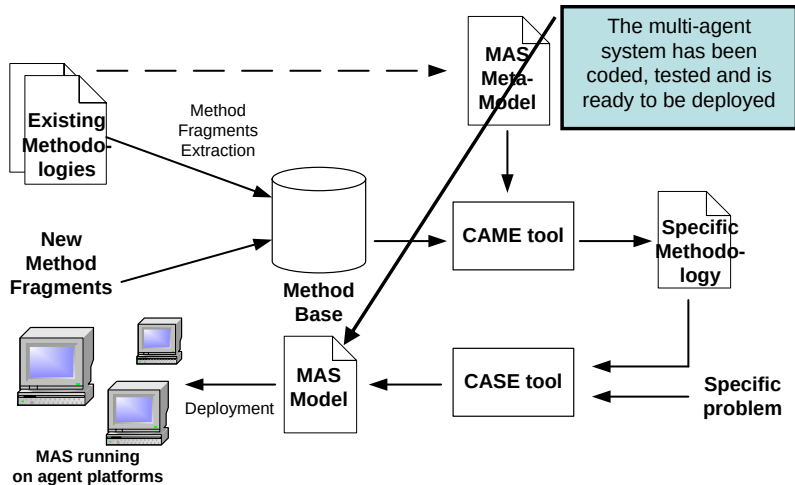
The new process production



The new process production



The new process production



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

8 Method Engineering in traditional SE

- Method Fragment Representation
- Method Fragment Assembly

9 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

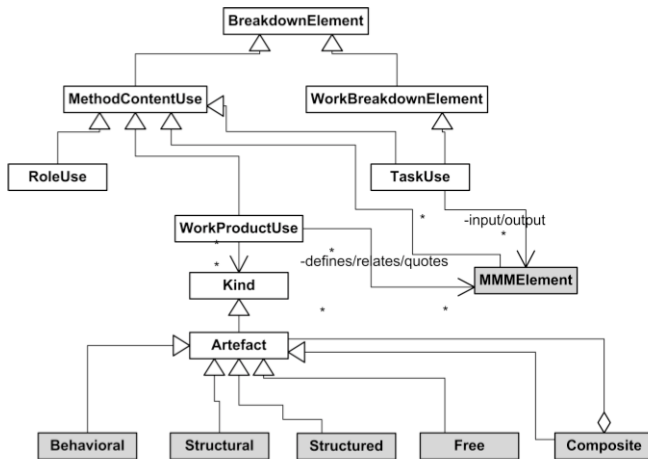
Extending SPEM Specifications [Seidita et al., 2009a]

- 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

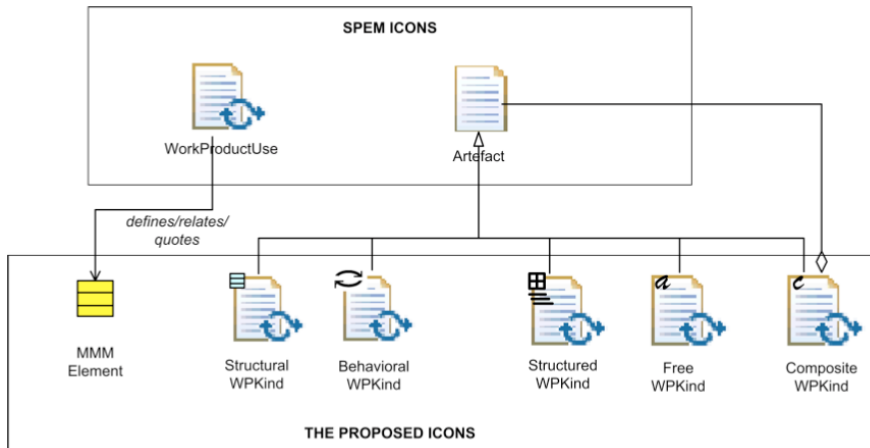
Extending SPEM Specifications [Seidita et al., 2009a]

- 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

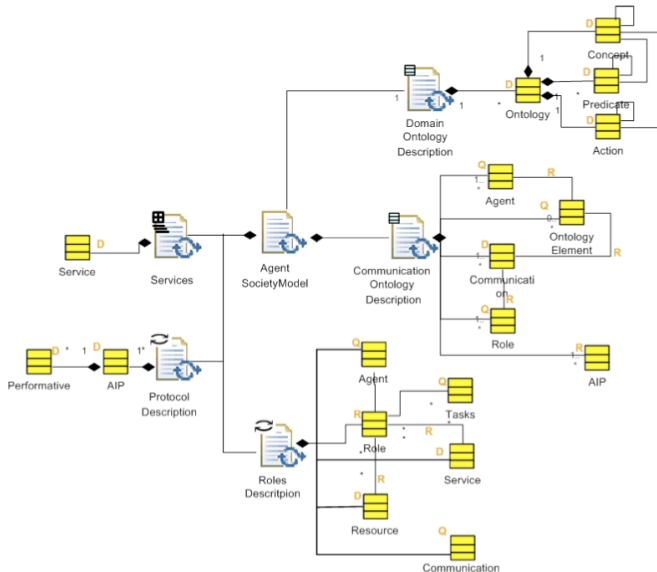
Extending SPEM Specifications [Seidita et al., 2009a]



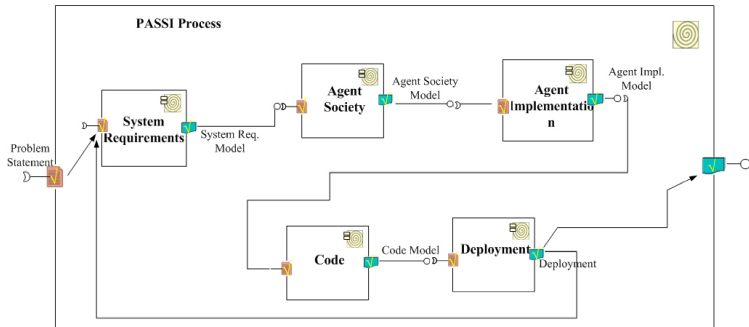
Proposed icons



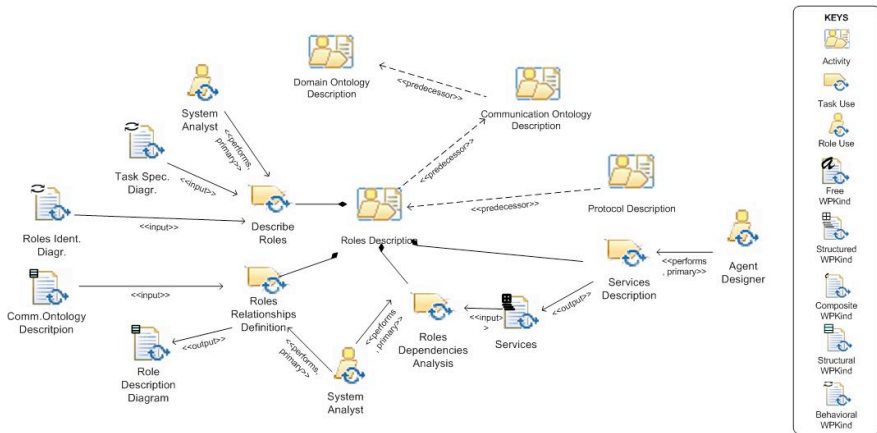
The dependency diagram



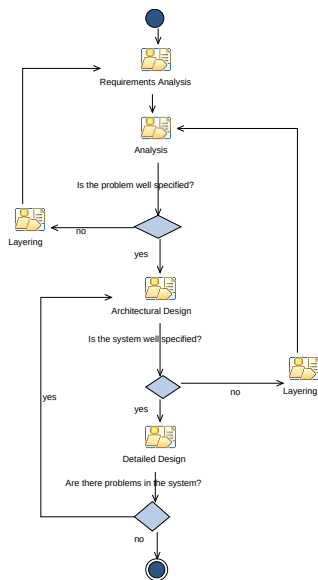
Example: PASSI component diagram



Example: PASSI process activity diagram



Example: the SODA process



Outline

8 Method Engineering in traditional SE

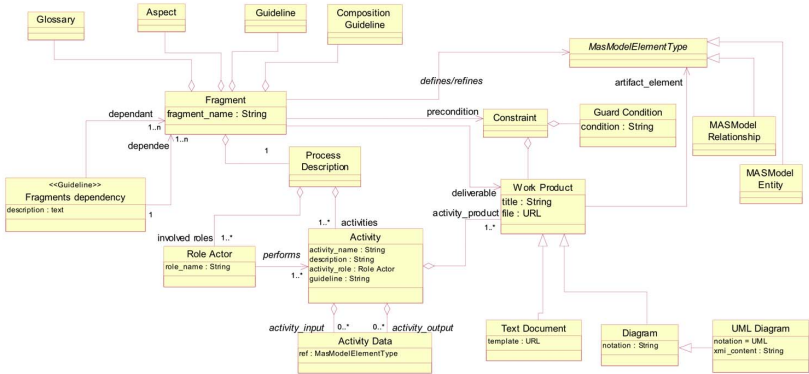
- Method Fragment Representation
- Method Fragment Assembly

9 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 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

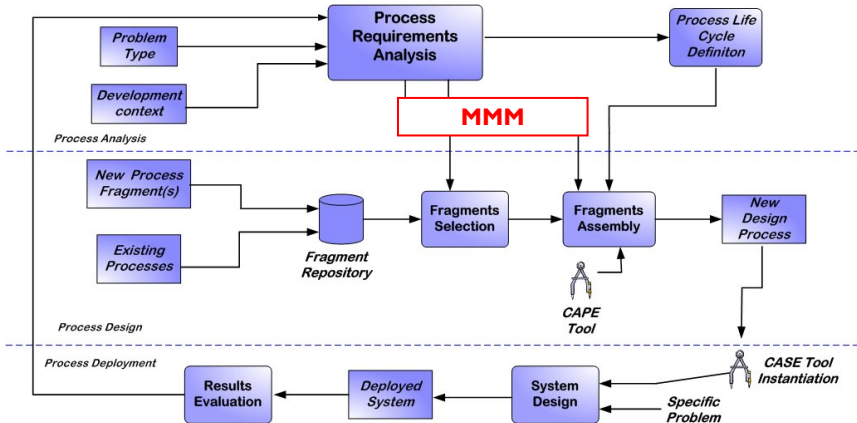
8 Method Engineering in traditional SE

- Method Fragment Representation
- Method Fragment Assembly

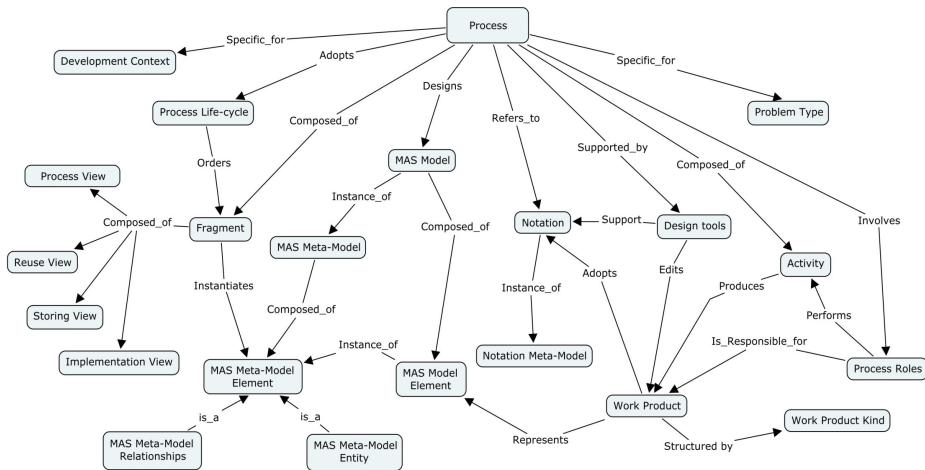
9 Method Engineering in AOSE

- 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., 2009b]



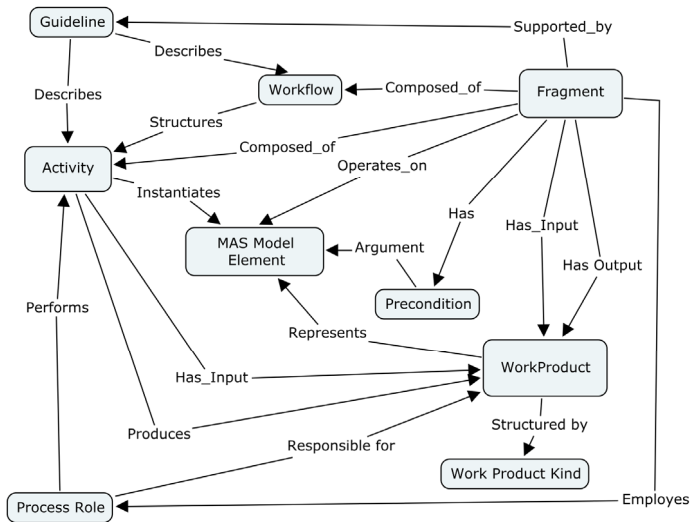
The PRODE Process Representation



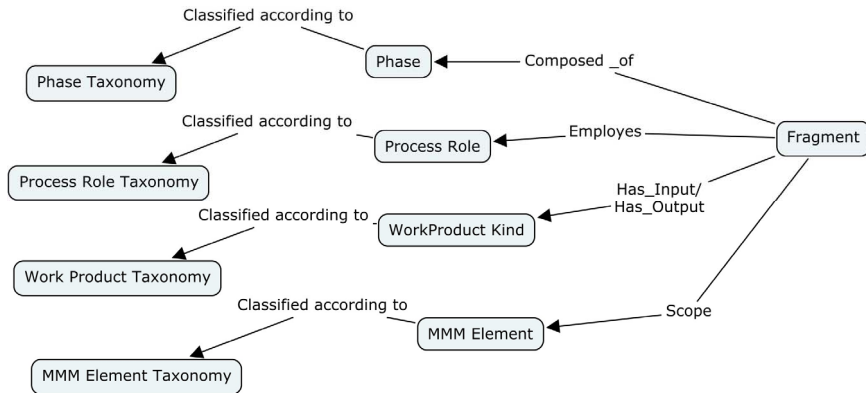
Applying the Proposed Method Fragment Definition

- 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.

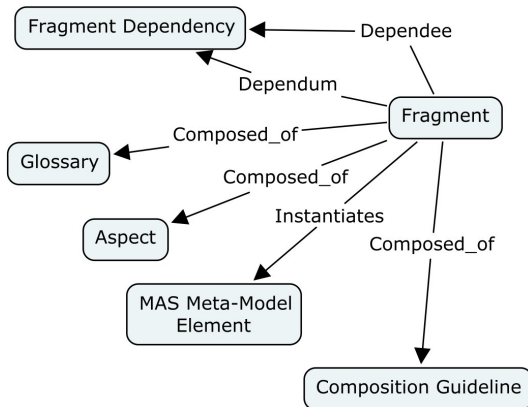
The process view



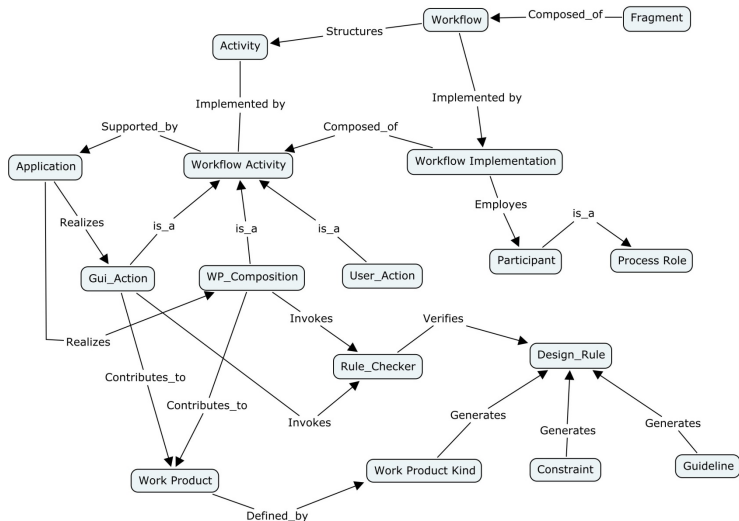
The storing view



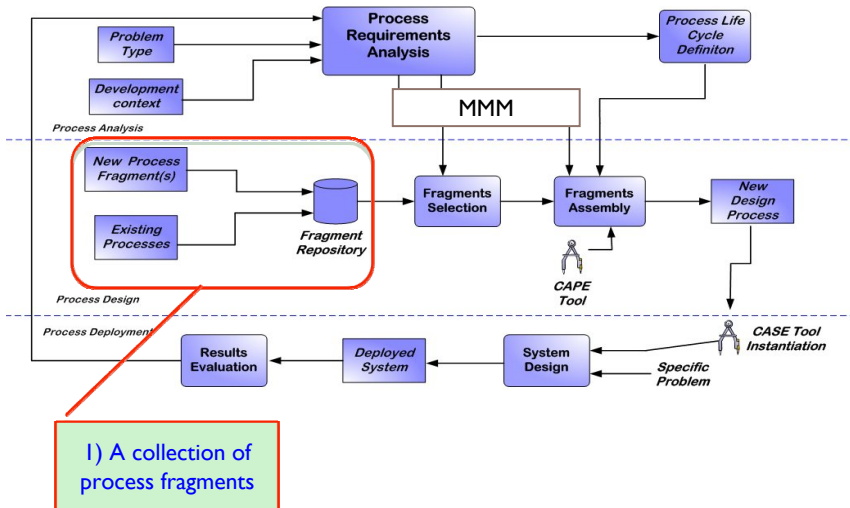
The reuse view



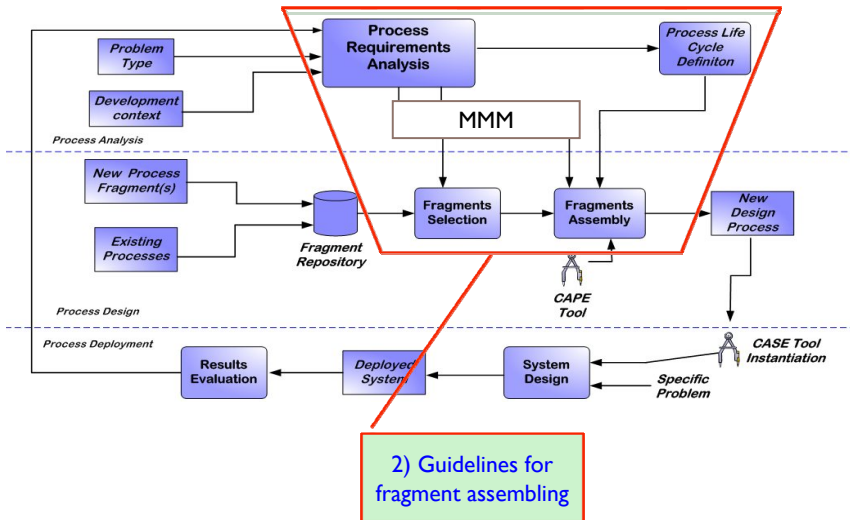
The implementation view



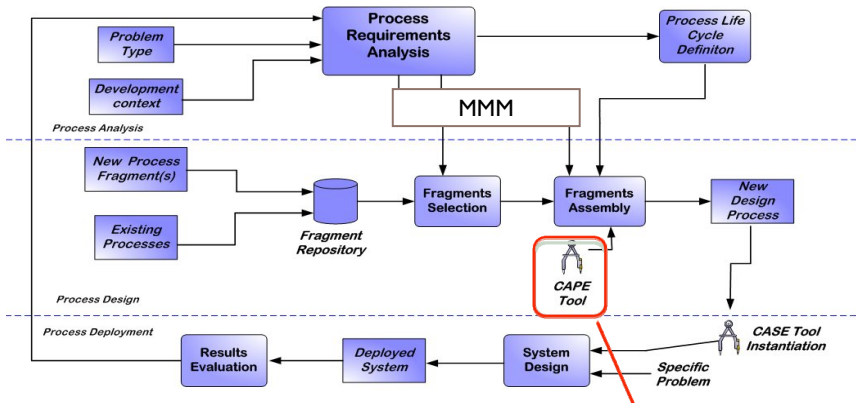
PRODE divided in three main areas of research



PRODE divided in three main areas of research

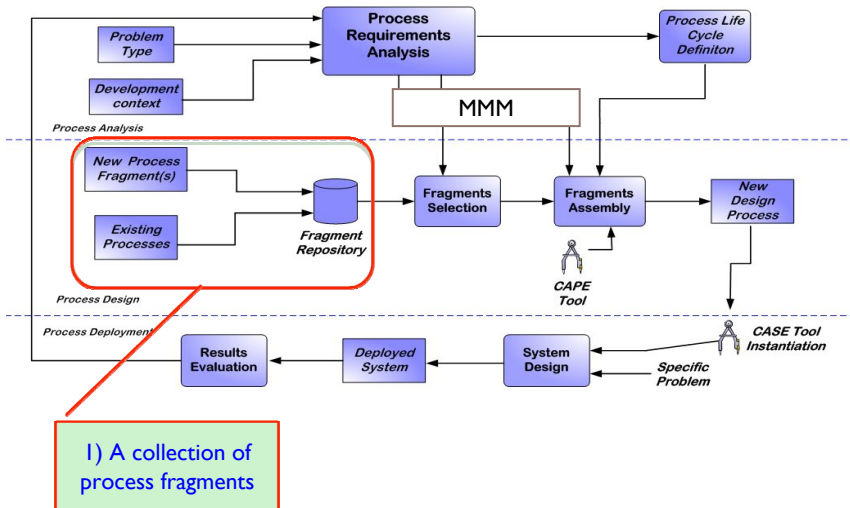


PRODE divided in three main areas of research

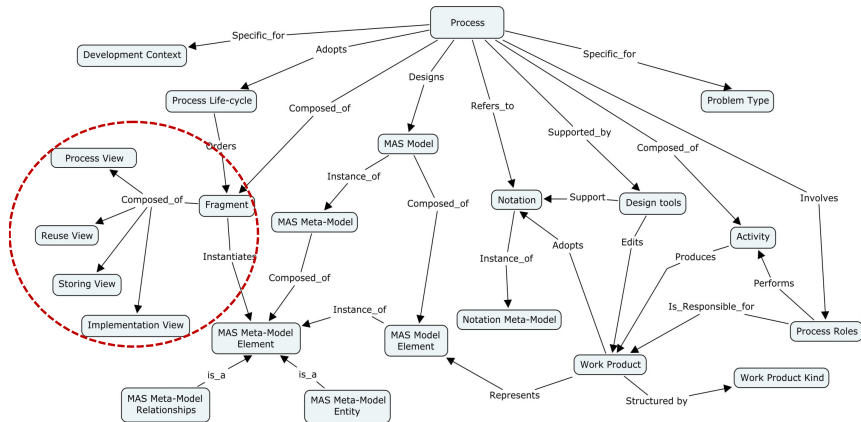


3) A CAPE (Computer Aided Process Engineering) tool

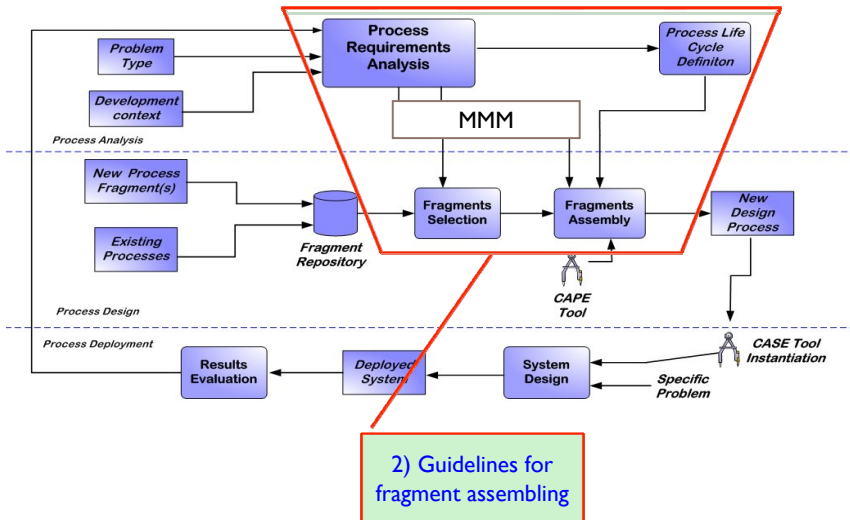
The fragment collection in PRODE



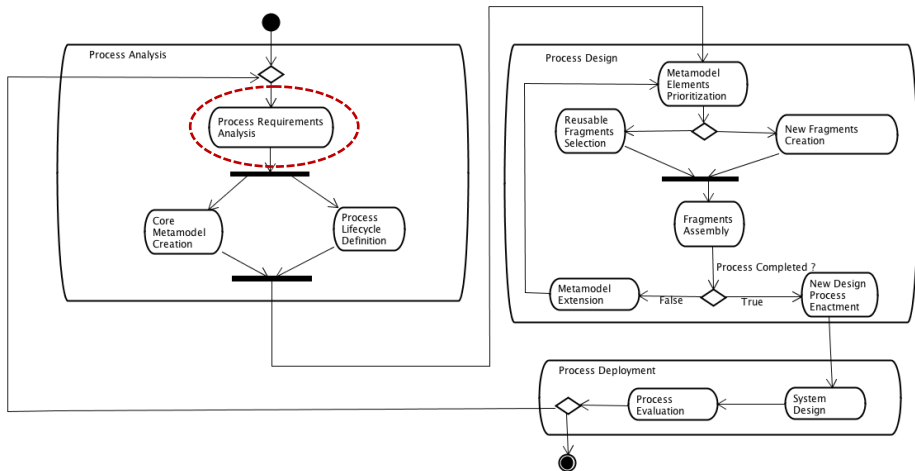
The PRODE Process Representation



Guidelines for fragments assembling



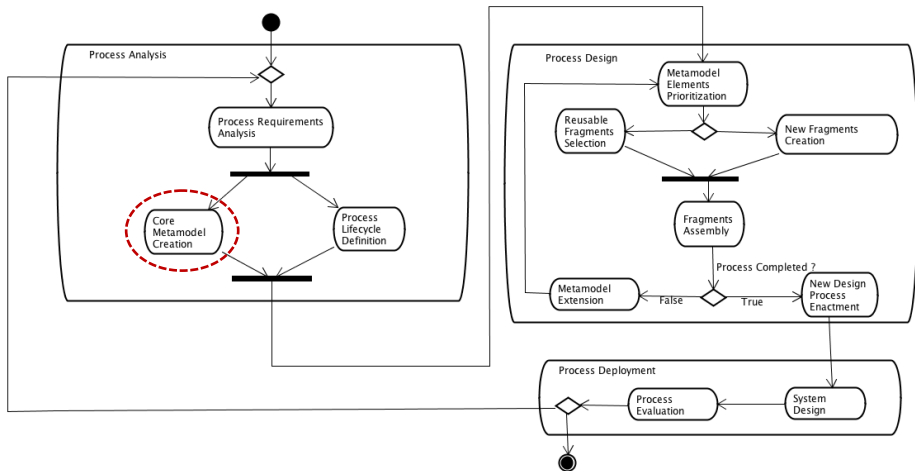
Process Analysis and Design in PRODE



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	

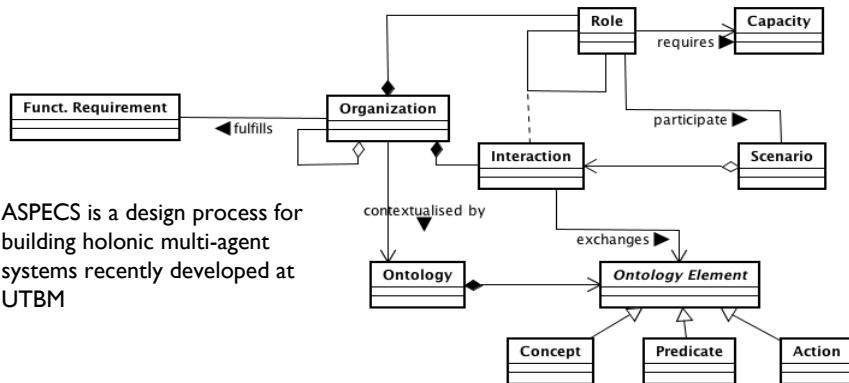
Process Analysis and Design in PRODE



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

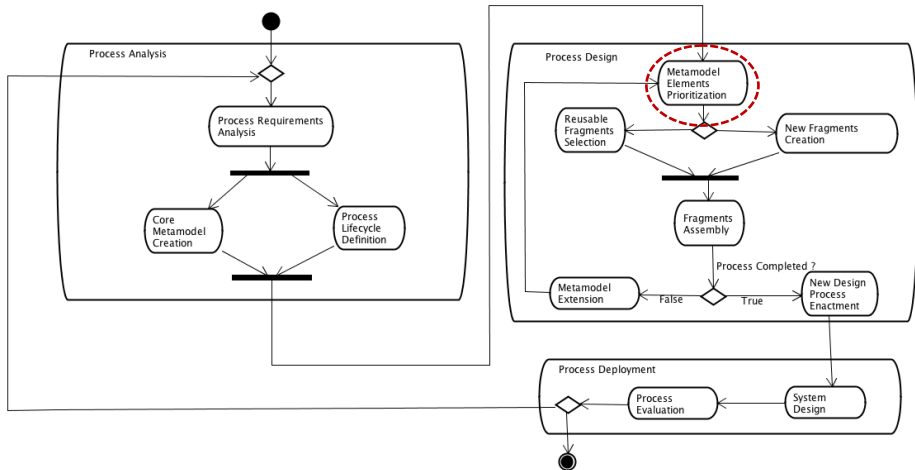
Example: Aspecs core meta-model



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

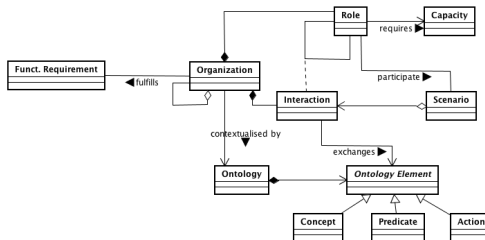
- A detailed description of ASPECS in [Cossentino et al., 2010]

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 ($i=1, \dots, n$) that: (i) can be instantiated by fragments of the repository and (ii) have less relationships with other elements
 1. Insert Li ($i=1, \dots, n$) in List elements1;
 2. Remove elements Li ($i=1, \dots, n$) from the core MMM;
 3. $p = p+1$;
6. While the core MMM is not empty
 - a) Select the leaves Li ($i=1, \dots, m$) that can not be instantiated by fragments of the repository;
 1. Insert Li ($i=1, \dots, m$) in List elements2;
 2. Remove Li ($i=1, \dots, m$) from the core MMM;

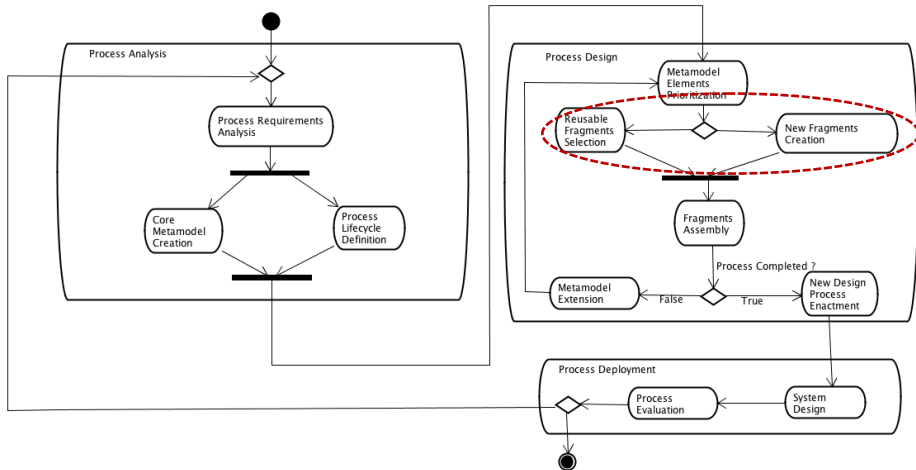
The Prioritization Algorithm (2 of 3)

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 ($i=0, \dots, n$) and outputs O_j ($j=0, \dots, m$) 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

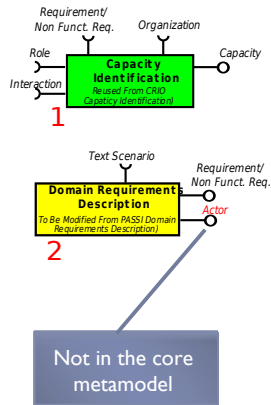
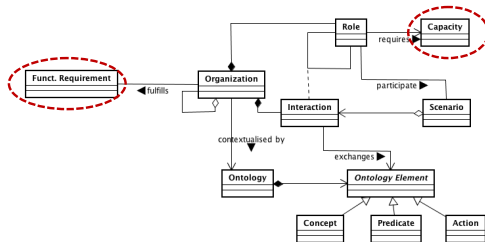
The Prioritization Algorithm (3 of 3)

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.

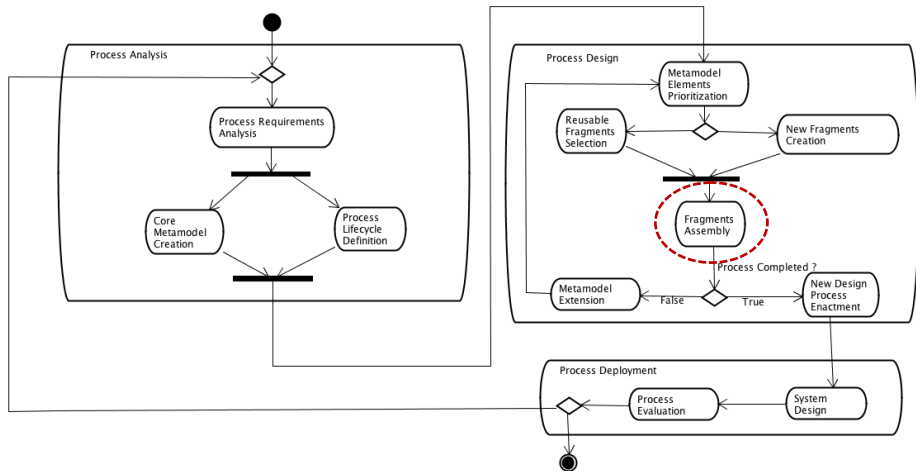
Process Analysis and Design in PRODE



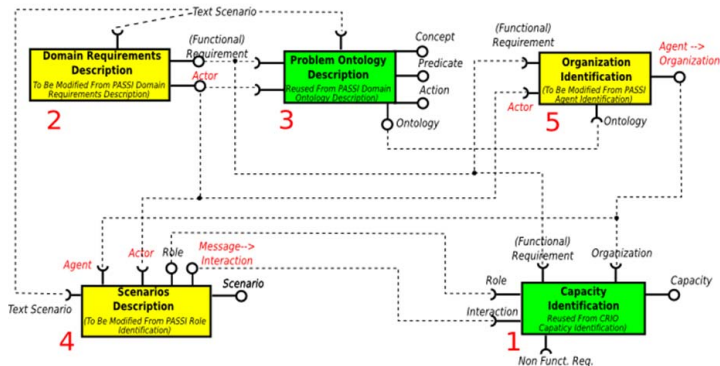
Example: the first two fragments in Building the ASPECS Process



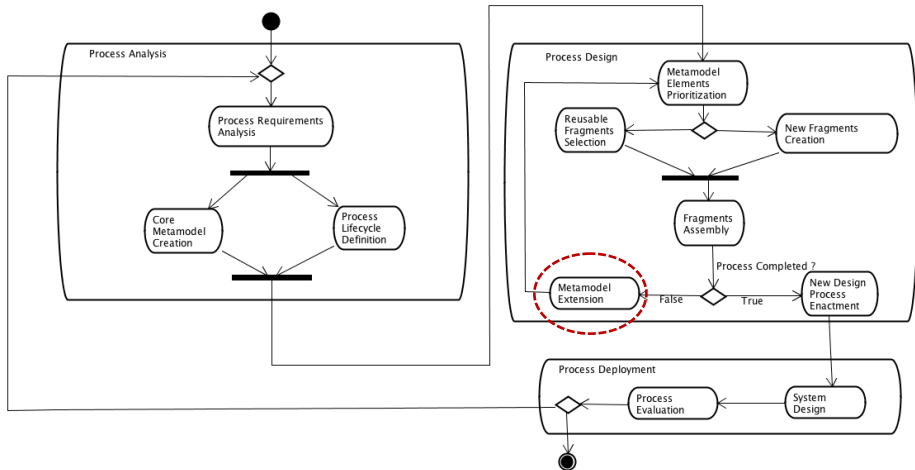
Process Analysis and Design in PRODE



Example: Aspects process component diagram



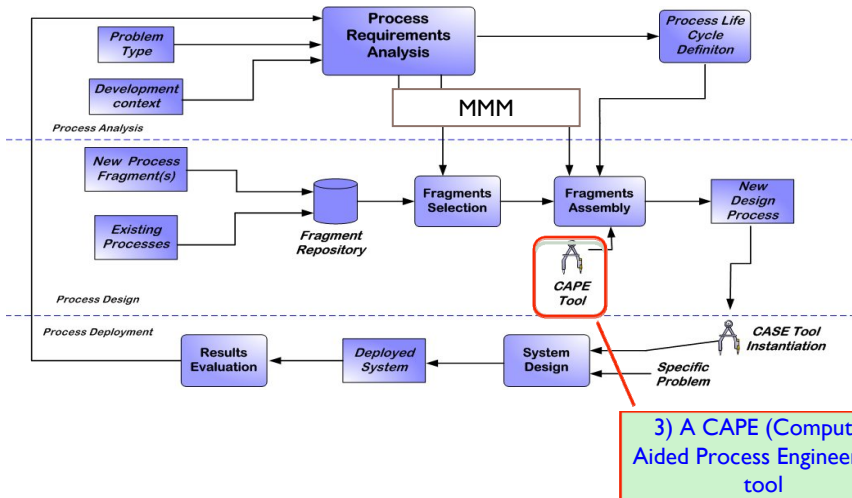
Process Analysis and Design in PRODE



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 MMEs 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

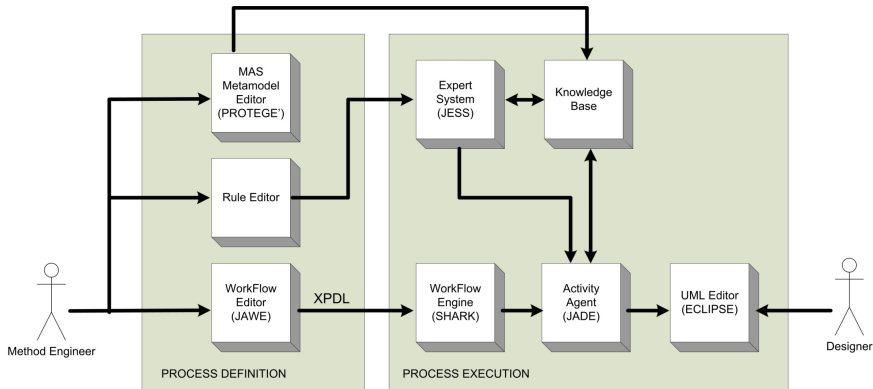
Supporting Tool in PRODE



Metameth

- **Metameth** is an (open-source) agent-oriented tool we built to support our experiments in methodologies composition and their application in real projects.
- Metameth is:
 - ▶ a CAPE tool: since it supports the definition of the design process life-cycle and the positioning of the different method fragments in the intended place
 - ▶ a CAME tool: since it allows the definition of different method fragments
 - ▶ a CASE tool: since it supports a distributed design process, it offers several (by now UML) graphical editors and an expert system for verifying the resulting system

Metameth tool architecture



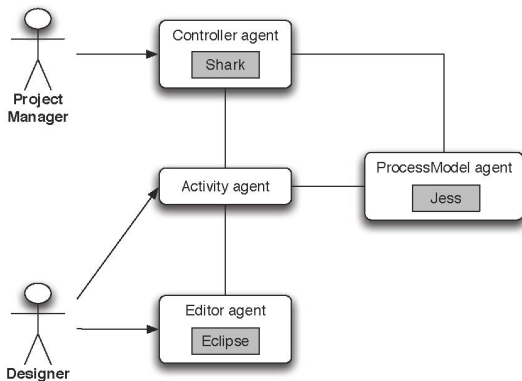
Supporting design activities

- 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)

The expert system

- 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 expert system



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

The rules

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

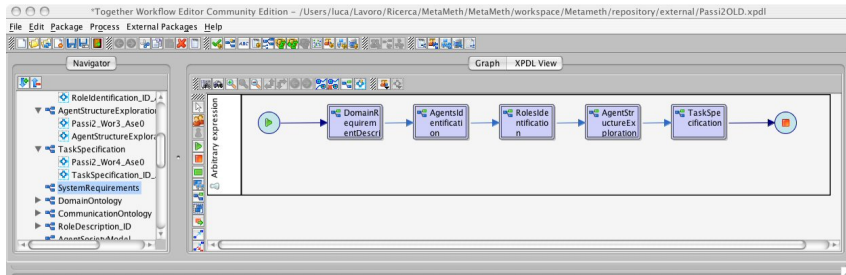
[– Validation
– Semantic interpretation] → **Rule Check**

[– 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 Protégè
- Services offered by the expert system:
 - ▶ syntax checks: it verifies the abundance to modelling language rules
 - ▶ semantic checks: it verifies the abundance 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



Outline

8 Method Engineering in traditional SE

- Method Fragment Representation
- Method Fragment Assembly

9 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

The need for a taxonomy

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

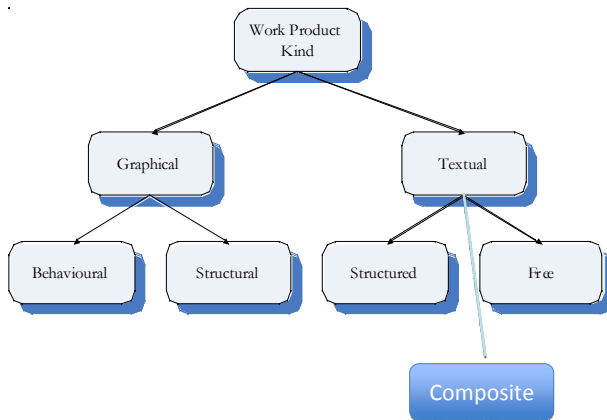
Phases

- Any kind of design process can be decomposed in phases
 - High level of abstraction for phases resulting from 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



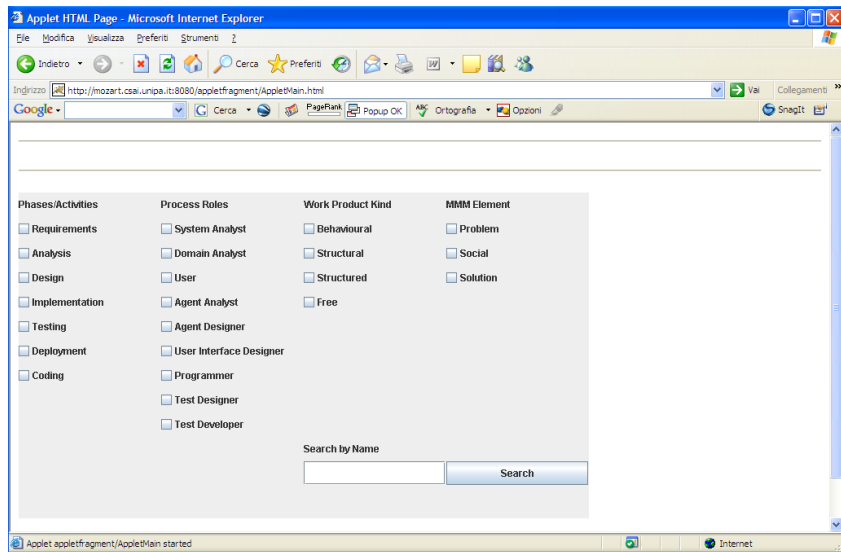
The need for a taxonomy

- 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

Repository content

Repository																																					
<table><tr><th colspan="2">Phase</th></tr><tr><td>Requirements</td><td>8</td></tr><tr><td>Analysis</td><td>16</td></tr><tr><td>Design</td><td>11</td></tr><tr><td>Implementation</td><td>4</td></tr><tr><td>Testing</td><td>1</td></tr><tr><td>Deployment</td><td>1</td></tr><tr><td>Coding</td><td>3</td></tr></table>	Phase		Requirements	8	Analysis	16	Design	11	Implementation	4	Testing	1	Deployment	1	Coding	3	<table><tr><th colspan="2">Process Role</th></tr><tr><td>System Analyst</td><td>14</td></tr><tr><td>Domain Analyst</td><td>7</td></tr><tr><td>User</td><td>1</td></tr><tr><td>Agent Analyst</td><td>6</td></tr><tr><td>Agent Designer</td><td>11</td></tr><tr><td>User Interface Designer</td><td>1</td></tr><tr><td>Programmer</td><td>2</td></tr><tr><td>Test Designer</td><td>1</td></tr><tr><td>Test Developers</td><td>1</td></tr></table>	Process Role		System Analyst	14	Domain Analyst	7	User	1	Agent Analyst	6	Agent Designer	11	User Interface Designer	1	Programmer	2	Test Designer	1	Test Developers	1
Phase																																					
Requirements	8																																				
Analysis	16																																				
Design	11																																				
Implementation	4																																				
Testing	1																																				
Deployment	1																																				
Coding	3																																				
Process Role																																					
System Analyst	14																																				
Domain Analyst	7																																				
User	1																																				
Agent Analyst	6																																				
Agent Designer	11																																				
User Interface Designer	1																																				
Programmer	2																																				
Test Designer	1																																				
Test Developers	1																																				
<table><tr><th colspan="2">WorkProduct Kind</th></tr><tr><td>Behavioural</td><td>9</td></tr><tr><td>Structural</td><td>24</td></tr><tr><td>Struc tured</td><td>6</td></tr><tr><td>Free</td><td>14</td></tr></table>	WorkProduct Kind		Behavioural	9	Structural	24	Struc tured	6	Free	14	<table><tr><th colspan="2">MMM Element</th></tr><tr><td>P roblem</td><td>12</td></tr><tr><td>S ocial</td><td>45</td></tr><tr><td>S olution</td><td>2</td></tr></table>	MMM Element		P roblem	12	S ocial	45	S olution	2																		
WorkProduct Kind																																					
Behavioural	9																																				
Structural	24																																				
Struc tured	6																																				
Free	14																																				
MMM Element																																					
P roblem	12																																				
S ocial	45																																				
S olution	2																																				

Method fragment retrieval



Outline

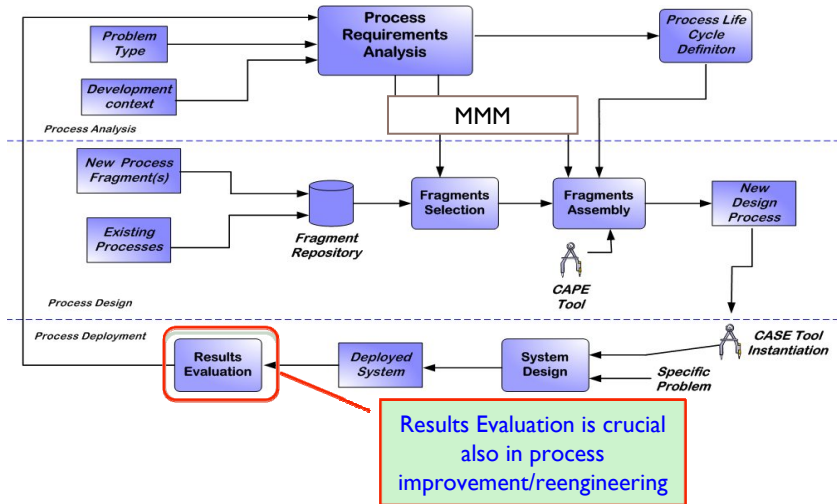
8 Method Engineering in traditional SE

- Method Fragment Representation
- Method Fragment Assembly

9 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**

Results Evaluation: an open problem?

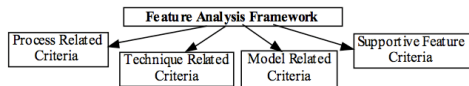


AO Design Process Evaluation

- 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-COMMONAKADS	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, GIS, 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

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.

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

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 organisation provides a way to predict an organisation's performance in a given discipline or set of disciplines
- A **maturity level** is a defined evolutionary plateau for organisational 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

Outline

10 Research directions and visions

11 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)

AOSE Observation Levels

- Micro scale
 - ▶ small- medium-size MASs
 - ▶ control over each component (limited complexity single stakeholder)
 - ▶ this is the (only) focus of mainstream AOSE
- Macro scale
 - ▶ very large scale distributed MASs
 - ▶ no control over single components (decentralization, multiple stakeholders)
 - ▶ the kingdom of “self-organisation” people
- Meso scale
 - ▶ micro scale components deployed in a macro scale scenario
 - ▶ my own system influence and is influenced by the whole
- Very rarely a fully fledged study can be limited to a single level of observation
 - ▶ most MASs (even small scale) are open
 - ▶ deployed in some sort of macro scale system
 - ▶ dynamically evolving together with the system

Micro-Level Challenges (1)

- Assessing AOSE Advantages
 - ▶ AO has clear advantages. What about AOSE?
 - ▶ methodologies, methodologies, methodologies... ;-(
 - ★ qualitative work
 - ▶ we need to show that AOSE
 - ★ helps saving money and human resource
 - ★ leads to higher quality software products
 - ★ quantitative comparison of AOSE vs. non-AOSE complex software development
- Pay Attention to the Software Process
 - ▶ most methodologies assume a “waterfall” model
 - ★ either implicitly or explicitly
 - ★ with no counterpart in industrial software development
 - ▶ Need for:
 - ★ agile processes
 - ★ agent-specific flexible processes
 - ▶ Cf. Knublauch 2002 “Extreme programming for MAS”, Cossentino 2006: “Agile PASSI”
 - ▶ Can meta-models be of help in that direction?

Micro-Level Challenges (2)

- Agent-specific notations
 - ▶ AUML is ok to spread acceptance but...
 - ★ is it really suited for MASs?
 - ★ and for complex systems in general?
 - ★ even the mainstream SE community doubts about that...
 - ▶ do more suitable notations exist?
 - ★ agent-specific ones to be invented
 - ★ other non-UML approaches
 - ▶ Cf. Sturm et al. 2003: OPM/MAS
 - ▶ AML by Whitestein [Cervenka et al., 2005]
 - ▶ A proposal of unified notation by L. Padgham, M. Winikoff, S. De Loach, M. Cossentino [Padgham et al., 2009]
- Intelligence engineering
 - ▶ selling AI has always been difficult
 - ★ lack of engineering flavor...
 - ▶ agents can help with that
 - ★ embodied, modular, intelligence
 - ★ observable rationality
 - ▶ our role should be that of:
 - ★ exploiting scientific results from the AI-oriented MAS community
 - ★ turn them into usable engineered products

Macro-Level Challenges (1)

- The macro level deals with complex collective behavior in large scale MASs
 - ▶ some say this is not AOSE...
 - ★ scientific activity
 - ★ observing and reproducing biology
 - ▶ but it must become an engineering activity
 - ★ challenging indeed
- Universality in MASs
 - ▶ can general laws underlying the behavior of complex MASs be identified?
 - ★ as they are starting being identified in the “complex systems” research community
 - ★ phase transitions, edges of chaos, etc.
 - ▶ letting us study and engineer complex MAS
 - ★ abstracting from the specific characteristics of agents (from ants to rational BDI agents)
 - ★ abstracting from the specific content of their interactions
 - ▶ Cf. Van Parunak 2004: “Universality in MAS”

Macro-Level Challenges (2)

- Measuring Complex MASs
 - ▶ how can we characterize the behavior of large-scale MASs?
 - ★ when we cannot characterise the behavior of single components
 - ▶ macro-level measures must be identified
 - ★ to concisely express properties of a system
 - ★ Cf. Entropy, Macro-properties of complex network, etc
 - ▶ and tools must be provided to actually measure systems
 - ▶ but measuring must be finalised
- Controlling Complex MASs
 - ▶ given a measurable property of a MASs
 - ▶ software engineers must be able to direct the evolution of a system, i.e., to tune the value of the measurable property
 - ★ in a fully decentralised way
 - ★ and with the possibility of enforcing control over a limited portion of the MAS
 - ▶ software engineering will become strictly related to control systems engineering
- Emergent behaviors, physics, biology, etc
 - ▶ Cf. The activity of the “SELF ORGANIZATION” Agentlink group

Meso-Level Challenges (1)

- It is a problem of deployment
 - ▶ engineering issues related to deployment of a MAS (typically engineered at a micro level of observation). . .
 - ▶ . . . into a large scale system (to be studied at a macro-level of observation)
- Impact Analysis
 - ▶ how will my system behave when it will be deployed in an existing open possibly large scale networked system?
 - ▶ how I will influence the existing system?
 - ▶ micro-scale aspects:
 - ★ tolerance to unpredictable environmental dynamics on my system
 - ★ internal handlings
 - ▶ macro-scale aspect:
 - ★ can my “small” MAS change the overall behavior of the global system?
 - ★ “butterfly effect”?

Meso-Level Challenges (2)

- Identifying the Boundaries

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

- Trust

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

- Infrastructures for Open Systems

- ▶ are configurable context-dependent coordination infrastructure the correct answer?
- ▶ are normative approaches the correct ones?
 - ★ We know what we gain but we do not know what we lose
- ▶ Cf. Incentives in social and P2P networks

Research directions and visions: conclusions

- There is not a single AOSE
 - ▶ depends on the scale of observation. . .
- The micro scale
 - ▶ overwhelmed by research
 - ▶ often neglecting very basic questions. . .
- The macro scale
 - ▶ some would say this is not AOSE
 - ▶ but it must become indeed. . .
- The meso scale
 - ▶ fascinating. . .
 - ▶ very difficult to be tackled with engineering approaches. . .
- What else?
 - ▶ there is so much to engineer around. . .
 - ▶ emotional agents, mixed human-agent organisations, interactions with the physical world. . .

Outline

10 Research directions and visions

11 Conclusions

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, *Organisational 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 on AOSE

- ▶ F. Zambonelli, A. Omicini, *Challenges and Research Directions in Agent-Oriented Software Engineering* [Zambonelli and Omicini, 2004],
- ▶ C. Bernon, M. Cossentino, J. Pavòn *An Overview of Current Trends in European AOSE Research* [Bernon et al., 2005c],
- ▶ C. Bernon, M. Cossentino, J. Pavòn, *Agent-oriented software engineering* [BERNON et al., 2006]
- ▶ C. Iglesias, M. Garijo, J. C. Gonzales, *A Survey of Agent-oriented Methodologies* [Iglesias et al., 1999]

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]

References on MAS Infrastructures

• 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 CArAgO* [Ricci et al., 2009]

Open Research Directions & Visions

- F. Zambonelli, V. Parunak, *From Design to Intention: Signs of a Revolution* [Zambonelli and Parunak, 2002]
- V. Parunak, S. Bruekner, *Entropy and Self-Organization in Agent Systems* [Van Dyke Parunak and Brueckner, 2001]
- R. Albert, H. Jeong, A. Barabasi, *Error and Attack Tolerance of Complex Networks* [Albert et al., 2000]
- G. D. Abowd, E. D. Mynatt, *Charting Past, Present and Future Research in Ubiquitous Computing* [Abowd and Mynatt, 2000]
- H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman and R. Weiss, *Amorphous Computing* [Abelson et al., 2000]
- M. Mamei, F. Zambonelli, L. Leonardi, *A Physically Grounded Approach to Coordinate Movements in a Team* [Leonardi et al., 2002]
- A. Roli, F. Zambonelli, *What Can Cellular Automata Tell Us About the Behavior of Large Multiagent Systems?* [Zambonelli et al., 2002]
- J. Doyle, J. M Carlson, *Highly Optimized Tolerance: A Mechanism for Power Laws in Designed Systems* [Carlson and Doyle, 1999]

Bibliography I



Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, Jr., T. F., Nagpal, R., Rauch, E., Sussman, G. J., and Weiss, R. (2000). Amorphous computing.
Commun. ACM, 43(5):74–82.



Abowd, G. D. and Mynatt, E. D. (2000). Charting past, present, and future research in ubiquitous computing.
ACM Trans. Comput.-Hum. Interact., 7(1):29–58.



Albert, R., Jeong, H., and Barabasi, A.-L. (2000). Error and attack tolerance of complex networks.
Nature, 406(6794):378–382.



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.

Bibliography II



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*

Bibliography III



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., COSENTINO, M., and PAVÓN, J. (2006).

Agent-oriented software engineering.

The Knowledge Engineering Review, 20(02):99–116.

Bibliography IV



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.

Bibliography V



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., Gómez-Sanz, J., Pavòn, J., Kearney, P., and Massonet, P. (2004).
The MESSAGE methodology.
In [Bergenti et al., 2004], chapter 9, pages 177–194.

Bibliography VI



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.



Carlson, J. M. and Doyle, J. (1999).

Highly optimized tolerance: A mechanism for power laws in designed systems.

Physical Review E, 60(2):1412–1427.

Bibliography VII



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.

Bibliography VIII



Chess, D. M., Harrison, C. G., and Kershbaum, 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.

Bibliography IX



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.

Bibliography X



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)*.

Bibliography XI



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).

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

International Journal of Autonomous Agents and Multi-Agent Systems (IJAAMAS).



DeLoach, S., Wood, M., and Sparkman, C. (2001).

Multiagent Systems Engineering.

International Journal of Software Engineering and Knowledge Engineering, 11(3):231–258.

Bibliography XII



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.

Bibliography XIII



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.



Grasia Group (2009).

Ingenias meta model.

<http://grasia.fdi.ucm.es/main/?q=en/node/135>.

Bibliography XIV



Henderson-Sellers, B. (2003).

Method engineering for oo systems development.

Commun. ACM, 46(10):73–78.



Henderson-Sellers, B. (2005).

Creating a comprehensive 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.

Bibliography XV

 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.

Bibliography XVI



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.

Bibliography XVII



Leonardi, L., Mamei, M., and Zambonelli, F. (2002).

A physically grounded approach to coordinate movements in a team.
In ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems, pages 373–378, Washington, DC, USA. IEEE Computer Society.



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.

Bibliography XVIII



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.

Bibliography XIX

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.

Bibliography XX



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.

Bibliography XXI



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.

Bibliography XXII



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>.

Bibliography XXIII



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.

Bibliography XXIV



Parunak, H. V. D. (1997).

“go to the ant”: Engineering principles from natural agent systems.
Annals of Operation Research, 75:69–101.



Pavòn, J., Gómez-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>.

Bibliography XXV



Ralyté, J. and Rolland, C. (2001a).

An approach for method reengineering.

In Kunii, H. S., Jajodia, S., and Sølvsberg, 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.

Bibliography XXVI



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 Carnegie 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 Carnegie Mellon University.

Bibliography XXVII



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. (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).

Bibliography XXVIII



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.

Bibliography XXIX



Tolvanen, J.-P. (1998).

Incremental method engineering with modeling tools: Theoretical principles and empirical evidence.

PhD thesis.



Van Dyke Parunak, H. and Brueckner, S. (2001).

Entropy and self-organization in multi-agent systems.

In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 124–130, New York, NY, USA. ACM.



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.

Bibliography XXX



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.

Bibliography XXXI



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.

Bibliography XXXII



Zambonelli, F., Mamei, M., and Roli, A. (2002).

What can cellular automata tell us about the behavior of large multi-agent systems?

In Garcia, A. F., Pereira de Lucena, C. J., Zambonelli, F., Omicini, A., and Castro, J., editors, *SELMAS*, volume 2603 of *Lecture Notes in Computer Science*, pages 216–231. Springer.



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.



Zambonelli, F. and Parunak, H. V. D. (2002).

From design to intention: signs of a revolution.
In *AAMAS*, pages 455–456. ACM.

Agent Oriented Software Engineering

Ambra Molesini¹ Massimo Cossentino²

¹ALMA MATER STUDIORUM – Università di Bologna (Italy)
`ambra.molesini@unibo.it`

²Italian National Research Council - ICAR Institute - Palermo (Italy)
`cossentino@pa.icar.cnr.it`

11th European Agent Systems Summer School
Torino, Italy
September 2009