International Journal of Software Engineering and Knowledge Engineering

The Metamodel: a Starting Point for Design Processes Construction

VALERIA SEIDITA

Dipartimento di Ingegneria Informatica - Università degli Studi di Palermo, Palermo, Italy
seidita@dinfo.unipa.it

MASSIMO COSSENTINO

Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Palermo, Italy
Systems and Transport Laboratory (SeT), Belfort, France
cossentino@pa.icar.cnr.it

VINCENT HILAIRE, NICOLAS GAUD, STEPHANE GALLAND, ABDER KOUKAM

Systems and Transport Laboratory (SeT), Belfort, France
vincent.hilaire@utbm.fr
nicolas.gaud@utbm.fr
stephane.galland@utbm.fr
abder.koukam@utbm.fr

SALVATORE GAGLIO

Dipartimento di Ingegneria Informatica- Università degli Studi di Palermo, Palermo, Italy
Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Palermo, Italy
gaglio@dinfo.unipa.it

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The construction of ad-hoc design processes following the Situational Method Engineering (SME) paradigm is currently carried out by adopting a set of phases for which, until now, no well defined techniques and guidelines had been established. The consequence is that organizations are very dependent on method designers' skills. In this paper we propose an approach based on SME for constructing customized agent oriented design processes. Our approach adopts the metamodel as the most important factor leading to the selection and assembly of method fragments and an algorithm for establishing the instantiation order of metamodel elements. The algorithm makes the proposed approach repeatable and usable even by not very skilled personnel thus proposing an improvement to the actual situation. The proposed approach and the algorithm are also experimented through the construction of a design process (ASPECS) for developing dynamic hierarchical societies of agents. The approach we created is general enough to be applied in other development contexts (not only agent-oriented).

Keywords: Metamodel; Design Process; Multi-agent system; Situational Method Engineering.

1. Introduction

Nowadays most researchers agree that it does not exist a one-size-fit-all design process [3][37][42] and this situation has been reported in a lot of application contexts. In the last years a great effort was spent by organizations for learning and reusing existing design processes and for customizing them for specific situations; the problem is that organizations generally are used to develop systems for one or at least few application domains.
Besides organizations are often composed of a limited number of people, with specific skills and competencies; hence, these organizations cannot use all kinds of design processes without spending too much time in personnel formation; all of that let the cost of producing a system increasing more and more.
A solution to this kind of problems can be to have ad-hoc created design processes for each situation and organization.
In this field, Situational Method Engineering (SME) [30][41], provides means for constructing ad-hoc Software Engineering Processes (SEP) following an approach based on the reuse of portions of existing design processes (often called *method fragments*).
Method Fragment is the core concept of SME; different well known approaches [3][27][31][16] present different definitions and descriptions of method fragment but all of them start from the assumption that whatever design process can be decomposed into (or it is composed of, if we use a bottom-up point of view) self contained components.
In order to create the new design process, the method designer needs to have access to a repository of components (or method fragments or simply fragments), probably coming from several existing design processes; from this repository he/she can retrieve and assemble the fragments in order to obtain the new SEP.
Note that although we are aware of the different definitions that can be found in literature according to personal point of views of researchers working in the field, in order to avoid entering in this debate, we will assume that the term Software Engineering Process (SEP) is synonym of: methodology, design process or simply process, and we will use all these words indifferently.

Today several approaches to Situational Method Engineering exist, each of them is based on the assumption that the knowledge on existing design processes has to be reused and they are mainly composed of three main phases: *process requirements analysis*, *fragments selection* and *fragments assembly*; none of these phases have been totally defined and described in these approaches; an interesting attempt to a generalization had been made in [40].
The difficulty to establish guidelines for the retrieval of method fragments from the repository and for assembling them is one of the most important problems, researchers in the field met in these years.
This is a relevant issue, because its solution is highly dependent from the particular adopted approach, the definition of fragment, and consequently from how the

repository is constructed. An optimal and general solution has not still been reached although some approaches are quite advanced on that [25][26].
The second important problem is that each approach is today too linked and too dependent from the skills and the knowledge of the method engineer that is constructing a specific design process; in fact the method engineer should deeply know the fragments stored in the repository, their inner activities, their deliverables, and he/she often selects and assemblies them on the basis of his/her own skills.
This situation causes the organization to be too bound to and dependent on the method engineer.
We can summary these arguments in two fundamental issues: (i) the lack of well defined techniques and guidelines for SME phases, and (consequently) (ii) the great dependency on the method engineer skills for properly enacting each phase. An approach including general or (even partially) automated techniques and guidelines would be highly desirable in order to encompass these problems. This is exactly the issues we worked on, and the solution we propose in this paper is based on the use of the metamodel as the central element for selecting and assembling fragments.
Our work is principally grounded on SME rationale and because our target is about multi-agent systems, this approach needs to be adapted /specialized. As a consequence, some differences exist between classical SME approaches and ours; the most significant one consists in the fact that we use the Multi Agent System (MAS) metamodel for defining the structure of the system we will build by adopting the new SEP.
Basing on MAS metamodel we establish a starting point also for the construction of the new SEP: the metamodel is used as the leading element for selecting and assembling fragments; more specifically, once a metamodel is constructed, an algorithm is adopted for choosing the order that should be used for instantiating the MAS metamodel elements in the process, in this way a first set of fragments can be retrieved from the repository.
Once these fragments are chosen, the same algorithm is used for assembling them thus producing a first prototype of the SEP under construction.
In other words, in our approach, the definition of a MAS metamodel entails the underlying design process thus becoming the key factor for the success of a new design process construction activity.

From the methodological point of view, our approach starts from the identification of the new process requirements in terms of development context, problem type and organization capabilities/existing processes maturity.
These requirements are used for defining an initial core version of the MAS metamodel. The elements of this metamodel are then ordered in a precedence list (with the already cited algorithm) and in this order we will retrieve the corresponding method fragments from the repository and assemble them in the new process.
If necessary, the core metamodel can eventually be extended with new elements (thus adopting an incremental/iterative approach towards the completion of the new design process) and the new corresponding method fragments can be included

in the process.

In this paper an experiment of creation of a new process (called ASPECS) is reported; this is not a classical toy problem but rather it deals with the construction of a full-size process for the design of agent-oriented systems. The scope of this process will be the design of dynamic hierarchical societies of agents; we aim at using the process and the related implementation platform (Janus [20]) for realizing open, dynamic holonic systems and solving complex problems requiring a huge number of agents.

The paper is organized as follows: section 2 provides a theoretical background on the issues the paper deals with (Situational Method Engineering, PASSI and CRIO metamodels); in section 3 the proposed approach is illustrated mostly detailing the construction of the core metamodel and the algorithm for selecting and assembling fragments. The case study on the construction of the ASPECS process is shown in section 4. Finally in Section 5 some discussions and conclusions are drawn.

2. Theorical Background

This section introduces the main concepts our work is based on. As regard the construction of the new design we refer to the SME approach that is discussed in subsection 2.1; whereas in the experiment described in this paper (the construction of ASPECS process) we largely reused elements from the PASSI and CRIO processes that are briefly introduced in subsections 2.2 and 2.3.

2.1. *Situational Method Engineering*

Several different approaches has been proposed from the beginning of the Situational Method Engineering discipline, each of them is based on different techniques and basic concepts [40].

All the existing approaches to Situational Method Engineering are mainly based on the reuse of portions of existing design processes (often called Method Fragments, Method Components or simply Fragments)[37]; these portions are usually stored in a repository called Method Base. Most of the reuse approaches are mainly composed of three phases [23]: the requirements specification of a project specific process, the selection of method fragments from a repository basing on the results of the previous phase and the assembly of the selected fragments.

The concept and definition of Method Fragment was firstly coined by Kumar and Harmsen [30] and by Brinkkemper [7]; during these latest years a lot of researchers based their work on this concept, sometimes with some differences but all of them, beyond the differences in the used terminology, shared the same fundamental aim: constructing a customized design process following an approach based on reuse of components of existing processes.

In [19][26] a huge repository of fragments is used for the composition of a new design process; the followed process is quite similar to that proposed in this paper in section 3 and is based on the so called deontic matrices. Deontic matrices contains

a set of values that gives indication for linking fragments in pairs.
The method designer has to elicit a set of requirements for the process under construction and then he has to identify the main activities composing the process development lifecycle; each activity can be linked through deontic matrices to tasks and then to techniques and work products thus obtaining a set of composable fragments.
Though the used repository contains a very large number of fragment this approach is mainly based on the experience a designer has about it and its content.

Another well known approach is proposed by Ralyté at al. in [33][34][31], here the formalism of maps with the triplet « source intention, target intention, strategy» is used to match the elicited process requirements to the method chunks stored in the method base. When the set of chunks that best fit the requirements is selected, the process designer can compose the new process following two strategies: integration and/or association, the first is used when the assembling chunks have similar aims, in this case modifications in the chunks are required after identifying the common elements whereas in the second case no overlapping in the chunks are present and one of them serves as an input of the second one.
These two assembly strategies can be also found in the proposed approach, we also can simply associate two fragments putting in sequence their work parts and joining the resulting work products or we can need some modifications thus making some kind of integration among fragments; this situation in the proposed approach is however discovered at a low level of work, looking at the MAS metamodel elements and their definition whereas in Ralyté et al. approach the smallest part of work that is considered is the chunk.

Finally, the first approach for applying SME is that proposed in [7][5][24] where the method engineer selects a set of method fragments that he considers as the best for fitting a particular situation and then modifies, adapts or takes from them the most reusable part basing on his own experience; from the assembly point of view this latest work is the most based on the designer skills.

What we propose in this paper, instead, aims at being as free as possible from the designer skills providing a set of reusable guidelines for fragments selection and assembly. In the case study proposed in section 4, we will reuse fragments coming from the PASSI and CRIO design processes; because of the particularities of the proposed reuse approach the importance of the multi-agent system (MAS) metamodel adopted in these processes is very relevant for our purposes and therefore, in the next subsections we will report a brief descriptions of these metamodels.

2.2. *PASSI*

Model driven engineering (MDE) [38] emphasizes the importance of artifacts in the development of software and after the standardization of the MDA architecture by OMG [1] it achieved a great diffusion among both researchers and practitioners. This approach proposes the use of some subsequent transformations for obtaining

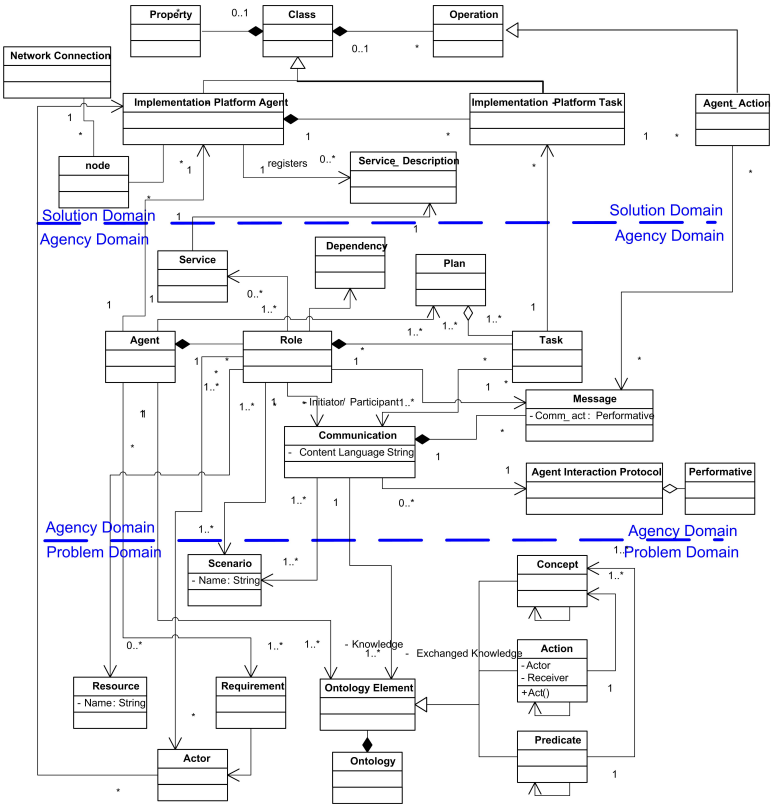

Fig. 1. The PASSI MAS meta-model

the system code. The first step of an MDA-based design approach, usually, consists in modelling the Problem domain thus producing the Platform Independent Model (PIM). This model is later transformed to the Platform Definition Model (PDM) that incorporates the constructs coming from the chosen implementation platforms and finally the Platform Specific Model (PSM) is generated as an executable solution for the studied problem.

As it is obvious, the availability of a good model of the Problem domain is one of the key success factors of transformational approaches.

The quality of this model is drastically influenced by the metamodel from which it is instantiated. A metamodel can be defined as a model of the model when looking at it from a purely structural point of view. This definition is probably too simplistic; a more interesting definition may probably be considered in order to introduce the perspective adopted in this paper: a metamodel defines the constructs and rules used to build a model. This definition emphasizes the semantic aspect of the metamodel (someone talks about ontological metamodelling [36][8] and it probably better fits the spread of the adoption of metamodels in the agent community in the last years.

When dealing with agent systems, metamodels are usually referred as Multi-Agent System (MAS) metamodels and they are very significant in this field since usually, different design processes underpin very different structures of the system to be.

In this and in the next subsection two MAS metamodels will be presented; the first is related to the PASSI process and the second to the CRIO approach.

The PASSI MAS meta-model [12] addresses three logical areas: (i) the Problem domain, (ii) the Agency domain and (iii) the Solution domain.
The Problem domain includes components coming from the world where the software is going to operate: these are directly related to the requirements analysis phase of the PASSI process.
Agency domain components are used to define an agent-based solution for the problem. Following this approach, we implicitly look at the agent paradigm as a problem decomposition and analysis instrument rather than a technological infrastructure for systems implementation.
Finally, in the PASSI MMM solution domain, agency-level components are mapped to the adopted FIPA-compliant implementation platform elements (we suppose the platform supports at least the concepts of agent and task); this represents the code-level part of the solution and it is the last refinement step. We will now only detail the Problem Domain as the other domains are not used in the case study presented in this paper.

The PASSI Problem Domain portion of the MAS metamodel (see Figure 1), deals with the user's problem in terms of scenarios, requirements, ontology and resources. Scenarios describe a sequence of interactions among actors and the system; they are used to identify the requirements that the system must fulfill. Requirements are represented with conventional UML use case diagrams. There is a strong point behind these choices: a lot of designers, already skilled with such an approach, are already present in different companies and can be more easily converted to the use of an agent-oriented methodology if they are already confident with some of the key concepts (and particularly the initial ones) used within it.
The ontological description of the domain is composed of concepts (categories of the domain), actions (performed in the domain and effecting the status of concepts) and predicates (asserting something about a portion of the domain, i.e. the status of concepts). Resources are the last element of the Problem domain. They can be accessed/shared/manipulated by agents. A resource could be a repository of data (like a relational database), an image/video or also a good to be sold/bought.

2.3. *CRIO*

CRIO is an organizational metamodel dedicated to model complex systems. It is based on the merge and the extension of two existing metamodels. The first, RIO [29], was conceived for the organizational and formal modelling of non-hierarchic multiagent systems, and the second is the generic framework for the modelling

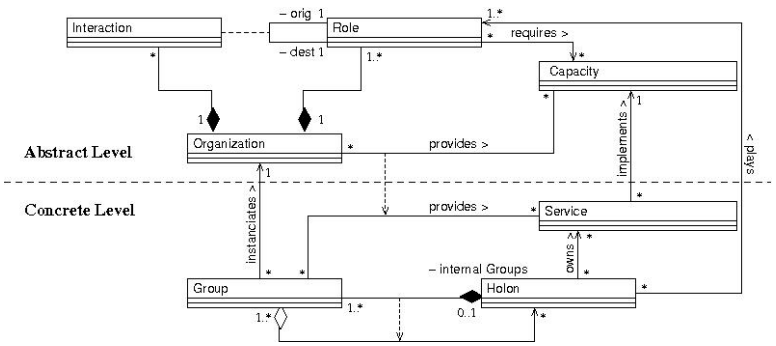

Fig. 2. The CRIO MAS meta-model

of holonic multiagent systems proposed by [35]. By considering organizations as blueprints that can be used to define a solution to a problem, we believe that an organizational approach encourages a the reusability of models.
CRIO is based on four main concepts: capacity, role, interaction and organization (see Figure 2). An Organization is defined by a set of roles, their interactions and a common context. These latter define a specific pattern of interaction. A Role is the abstraction of a behavior in a certain context and confers a status within the organization. The Role gives the playing entities the right to exercise its capacities. Roles may interact with other roles defined in the same organization. The role playing relationship between roles and agents is dynamic. In other terms, at any given time agents may request to play new roles and leave roles that they are currently performing. Roles may interact with other roles defined in the same organization. An Interaction is composed of the event produced by a first role, perceived by a second one, and the reaction(s) produced by the second role.

To obtain a generic model of organization, we need then to define a role without making any assumptions on the architecture of the holon which will play this role. Basing the description of these behaviors (Roles) on capacities, enables a modular and reusable modeling of holonic MAS. Indeed, capacities describe what the holon is capable of doing (Abstract Level), independently of how it does it (Concrete Level).
A role defines a behaviour based on what the agent/holon is capable of doing (i.e. the holon's capacities). Thus, a role requires that the role player has specific capacities. A holon/agent has to possess all capacities required by a role to play that role. On the other hand, a role confers to its player a certain status in the organization and the right to perform its capacities. A role thus confers holon the authorization to wield some of its capacities in the context defined by the organization.
At the Concrete Level we assign roles to agents. In this context, an agent is only specified as an active communicative entity which plays roles. An agent/holon may instantiate one or more roles and a role may be instantiated by one or more

agents/holons.

A holon is a whole-part construct that is composed of other holons, but it is, at the same time, a component of a higher level holon. In order to maintain the generality of the metamodel, it is necessary to distinguish between two aspects that overlap in a holon. The first is directly related to the holonic character of the entity, i.e. a holon (super-holon) is composed of other holons (sub-holons or members). This aspect is common to all the holons, thus it is often named the holonic aspect. This aspect considers how members organize and manage the super-holon, and it is modelled with a particular organization called Holonic Organization. This latter represents a moderated group [21] in terms of roles and their interactions.
The second aspect, is related to the problem the members are trying to solve; it is therefore specific to the application or domain of application, and often named production aspect. This aspect is modelled by using a set of production organizations. A holon may thus be composed of groups. A super-holon contains at least the holonic group and possibly a set of production groups, instances of production organizations.

3. The Proposed Approach for the Construction of a New Design Process

Several techniques and tools had been proposed in these latest years for the construction of new ad-hoc processes [19][6][4][30]. Each of them can usually be applied to a specific field (OO, IS, etc.); we adopted the principles of SME and created a process for defining new design processes (SEP) that are specific for the construction of ad-hoc agent-oriented systems.
The process we created is general enough to be applied to every kind of problem, we tested it upon the agent context because from years we work in this area but the obtained results can be easily generalized and the main contribution we propose (the adoption of the system metamodel as a guideline for selecting and assembling fragments) is valid also for non agent-oriented design processes. We called the proposed approach PRoDe (PRocess for the Design of Design PRocesses).

PRoDe is based on the classic situational method engineering main phases (see 2.1), on a specific definition of method fragment, that we call *Process Fragment* [11] and on what we consider a key point for the construction of a new process: the metamodel. The metamodel constitutes the pivot we adopted for applying SME principles; the focus on this element is one of the most important factors that makes our approach different from all the others proposed in literature.
PRoDe is organized in three main phases (see Figure 3):

- *Process Analysis*: where the process to be developed is analyzed and its requirements are elicited. The idea that requirements can be elicited for a design process is not new and directly descends from positions like the one proposed by Osterweil in [32]. Process Requirements Analysis, in the proposed approach, produces a set of elements, mainly a portion of the

system metamodel, deeply affecting the following phases.

- *Process Design*: during this phase the method engineer selects and retrieves, from a previously constructed repository, a set of reusable fragments that he assembles in the new process. In so doing he usually follows a set of guidelines based on the structure of the metamodel resulting from the previous phase.
- *Process Deployment*: here the new design process is enacted and used to solve a specific problem; it is evaluated in order to verify the achievement of the expected results and to gather further requirements (if any); if necessary, it is also possible to repeat the whole construction process in an incremental/iterative way.

All these phases are supposed to be performed under the assistance of specific tools: *i)* the process requirements analysis and process design is assisted by a Computer Aided Process Engineering (CAPE) tool that supports the method engineer in all the phases he performs during the process creation [14][15]; *ii)* the selection and retrieval of fragments from the repository is made with the aid of a Computer Aided Method Engineering (CAME) tool providing an interface for querying the fragments repository, a first prototype of CAME is already part of PRoDe [39]; finally *(iii)* a Computer Aided Software Engineering (CASE) tool is used during the process enactment phase. This latter is conceived as an instance of the CAPE tool that supports the designer during system design activities performed according to the specific new process.

In this section we describe how the new process is constructed. This process aims at supporting a method engineer who wants to construct a design process for developing multi-agent systems that solve a particular problem. It is reasonable to suppose that the first thing he does is to check if some kind of process is already existing in the organization and how the involved stakeholders use it.
The most common situation is that a kind of process is already in use, sometimes it is well documented and it only needs improvements, sometimes it needs to be created almost from scratch. In any case the method engineer has to analyze the organization development context, and the problem types faced within that, in order to gather the set of requirements new process has to fulfill. Further details about this activity will be provided in the next subsection.

3.1. *Process Analysis*

The main aim of this phase is to define the process lifecycle that gives a structure to the process to be built thus providing a guide for assembling fragments and to define the system metamodel; this phase is composed by the following activities (see Figure 3):

- **Process Requirements Analysis**. It has inputs coming from the maturity level of the organization, the development context (tools, languages,

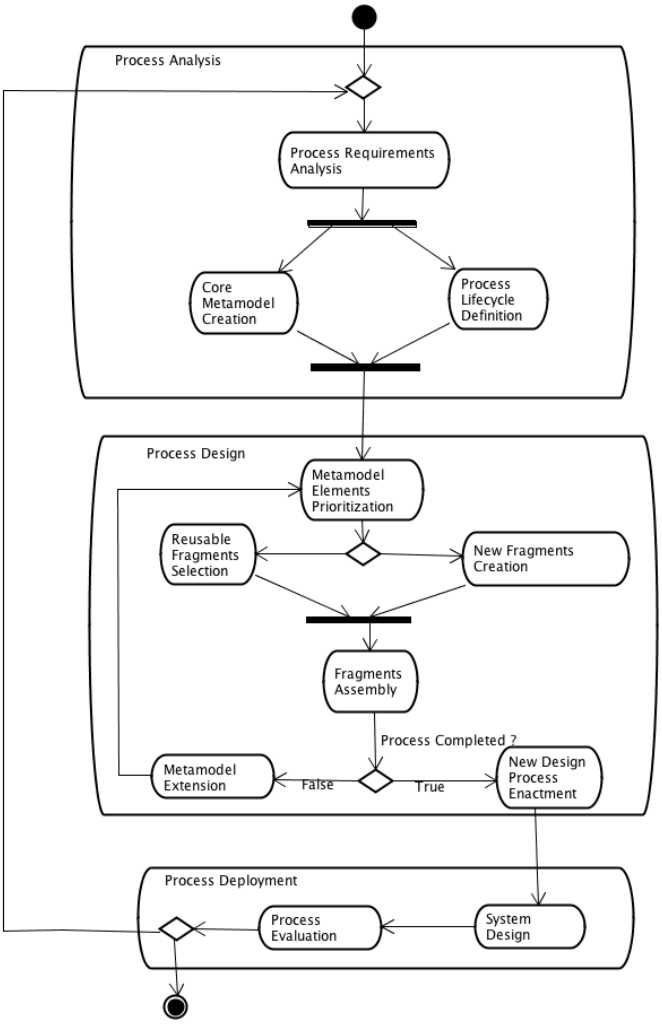

Fig. 3. The proposed Situational Method Engineering Process

available skills, etc.) and the type of problem to be solved. From these inputs the method engineer achieves the necessary information for identifying some fundamental requirements about the new design process.
These requirements define the domain of interest that the new process should take into account. For example, if the problem type deals with transportation of human beings and if someone in the development group has formal methods practice then some safety properties of the system may be proved.

- **Core Metamodel Creation.** The elicited requirements are useful for creating the system metamodel whose elements are used for selecting the proper fragments from the repository and for assembling them. Process requirements are also useful for describing the main process elements, for identifying the work that has to be done in order to produce a specific output and which stakeholder has to perform it.

- **Process Lifecycle Definition** A design process specifies when and how someone does something in order to reach a specific objective, so the process requirements analysis also affects the Process Life Cycle Definition activity: it is concerned with the decision about the process model (or life-cycle) to be adopted; this decision is influenced by several factors (for instance contract constraints imposed by the customer/commissioner on that). According to some studies it seems that the process life-cycle is not affected by the adoption of the agent paradigm and therefore classical life-cycles (waterfall, spiral, iterative/incremental, etc.) can be used for designing agents too [9].

The inputs of the Process Analysis phase are:

- *Process Capability.* It is the concept designed in the SEI Process Capability Maturity Model Integration (CMMI) for Development: "Software process capability describes the range of expected results that can be achieved by following a software process". The software process capability of an organization provides means for predicting the most likely outcomes to be expected from the next software project the organization undertakes. In this way it is well defined how to work for achieving fixed objectives. In our work the identification of these activities results in a well defined set of requirements on method fragments to be selected or on specific stakeholders to be involved in the process.
- *Problem Type.* The new process has to be tuned for a specific solution strategy to a class of problems. It is possible that, in a big company, different groups produce software for totally different areas (for instance business administration, and biological systems simulation). In this situation it should be expected that each of these groups adopts a different design process giving the right importance to the aspects that are more sensible in its target implementation domain.
- *Development Context.* It is a description of the available resources (both human and non human) and competencies that are available in the process enactment group. The development context is usually a sensitive aspect to be considered also because if the group is composed of people skilled with some specific approach or standard practice (for instance the use of UML in modelling the system), it is highly desirable to capitalize such an experience in order to lower training costs that always follow the introduction of a new design process. In the development context we also enumerate possible

constraints that could come from available developing tools.

The main outputs of the Process Requirements Analysis phase are:

- the *core metamodel*. It contains all the concepts that can be used to design and describe the system to be: it defines domain-specific concepts, solution concepts and all the concepts that specifically address the characteristic of the particular system a designer is developing, together with all of their relationships. For instance in the case of a MAS (Multi Agent System), a metamodel provides concepts such as agent, role, communication, agent task, and so on. Each concept of the metamodel must be designed/defined (that means instantiated) at least in one fragment of the process (whereas it can be cited in several other fragments).
- *Process Elements.* It is a list of elements composing the new design process. These elements can be activities (the work to be done), process roles (particular stakeholder performing the work) and work products (artefacts resulting from some activities) and they too can be used for the retrieval of method fragments [11][39].

Metamodel elements and process elements greatly affect the following phase since they are the main inputs for the selection of fragments from the repository. In fact, in the next phase (Process Design phase), the designer will use these inputs for building the new process. This can be done by firstly defining an ordered list of MAS metamodel elements and then using this list for retrieving the fragments from the repository.
At this stage it is likely that some fragments will need modifications and the initial metamodel some extensions. These operations will be performed in an incremental/iterative way until the new process is fully defined; further details about these activities are provided in the next section.

3.1.1. *Specific Issues with the Creation of the Core MAS Metamodel*

According to our previous experiences, we regard the construction of a metamodel as an iterative process. In this work, as well as in a previous one [13], we composed the new metamodel on the basis of portions of metamodels coming from other design processes (as prescribed by the results of the process requirements analysis activity reported in Table 2).

In so doing we are aware that defining the core MAS metamodel means defining a relevant part of the 'philosophy' that will be behind the new design process and for this reason we performed this activity during several meetings often involving stakeholders that would be employed in the usage of the new design process.

While defining the new MAS metamodel by starting from parts of metamodels coming from other processes, we can incur in three different situations:

(1) The different metamodels contribute to the new one with parts that are to-

tally disjointed (this happens when the selected metamodel elements combine together without overlapping each other). In this case a harmonization of concept definitions coming from different sources is probably advisable but it is not strictly required. An example of this situation can be the case when the resulting core metamodel includes the concept of agent coming from one design process and the concept of role coming from another; they are related in the resulting model but it is not strictly necessary to change their definitions, sometimes they are already congruent (a detailed check is anyway highly advisable).

(2) The different metamodels contribute to the new one with parts that overlap each other and overlapping elements have the same (or similar) definitions bounded to elements with different names. In this case a definition-oriented identification of concepts for the new metamodel assembly is advisable. This means that concepts should be addressed on the basis of their definitions and a partial or even global renaming activity is required in order to have a well composed metamodel with concept names and definitions put in harmony each other.

(3) The different metamodels contribute to the new one with parts that overlap each other and overlapping concepts have the same names bounded to elements with different definitions. This is the worst case since coherence of definitions of the new metamodel elements is not easily reached. Elements coming from different metamodels, when assembled in a new one, can originate non-sense situations that can only be solved by modifying the definition of the elements, in order to define a meaningful structure.

We already discussed a case study mainly belonging to the above described point 3 in [13]; in this paper we will mainly discuss the first case since parts coming from different metamodels will not significantly overlap. Of course, it is also possible that the above-discussed cases are all together present in a specific experiment.

3.2. *Process Design*

The aim of this phase, such as the aim of architectural design phase during software development, is to identify the main elements of the SEP and the general schema of their interactions; in our case these elements are the process fragments that have to be selected/retrieved from the repository and then assembled in the new SEP.
Process design phase is composed of five activities: *Metamodel Elements Prioritization*, *Reusable Fragments Selection*, *New Fragments Creation*, *Fragments Assembly*, *Metamodel Extension* and *New Design Process Enactment*.

- **Metamodel Elements Prioritization.** During this activity the method engineer identifies the order in which the MAS metamodel elements have to be instantiated during the development of the new SEP. This order constitutes the principal guide for the selection of fragment and their assembly.
  The metamodel is the main input of this activity; the method engineer ana-

lyzes its structure in order to establish the level of priority of each element. The output is a list used in the following activity.

- **Fragments Selection and New Fragments Creation.** The process fragment is the building block of process design; it is extracted from existing design processes, or created from scratch, and stored in a repository, called method base from which it is selected basing on the results of requirements analysis. A process fragment can be extracted from existing design processes or created/-modified to meet a specific requirement of the new process.
  The configuration of our actual fragments repository and how to use imt for selecting the fragments have been already discussed in [39].
- **Fragments Assembly.** This activity is still one of the most important unresolved points in the SME field and some proposal have been done in [33][25][26]. It is a very complex work where the method designer has to collate all the elements gathered in the previous activities and to merge them using his experience and skills.
  We think that in some cases a set of fragments can be directly associated each other, in other cases they need modifications of one (or more) constituting elements.
  For instance if two fragments adopt different semantics for the produced work products, it is necessary to change (or adapt) the elements of one work product kind to the other to allow a right assembly, or if the process part of two different method fragments overlap then one, or both of them, must be modified in order to create a unique consistent process.
- **Metamodel Extension.** This activity is carried out if after a first assembly activity there should be the necessity of extending the core metamodel. In fact it may happen that one or more of the selected fragments present elements (inputs and/or outputs) that do not belong to the core metamodel; in this case in order to use them an extension and a further analysis of the core MMM is needed.
- **New Design Process Enactment.** When no core MMM extension is needed then the new SEP is enacted and can be used by one or more system designer for developing a multi-agent system.

3.2.1. *The algorithm for selecting and assembling fragments*

In the proposed approach, the metamodel is supposed to be used for the selection and the assembly of fragments; the question at this point is: which is the starting point? Assuming that each element of the core metamodel has to be defined (instantiated) in at least one fragment, it is obvious to consider that the order of instantiation is relevant because of the elements' mutual dependencies.
This is exactly what happens, for instance, for the order of compilation of the files composing a complex software. We argue that the order itself should therefore descend from the structure of the metamodel and for this reason; in this subsection

we propose a prioritization algorithm.
According to this argument, once the core metamodel has been defined, the next step consists in the MAS metamodel elements (MMMEs) prioritization.

The order we will find is strictly related to the features the process will exhibit. For instance, a process with an early identification of roles and a late binding of these roles to the agents that will play them (like it happens in GAIA [44]) will obey to a philosophical design approach that is different from that of another process where agents are identified before roles (like it happens in PASSI [18]). It is worth to remember that, in our approach, the goal of the process design phase consists in selecting and assembling all the fragments defining the elements of the MMM according to a well established order.

The work can be represented as a cycle composed of three subphases: (i) prioritization of MMMEs; (ii) identification and assembly of process fragments defining the MMMEs; (ii) extension of the metamodel until the complete process is defined. The algorithm we adopted in performing this activity is reported in Table 1.

This algorithm is based on the following assumptions:

- MAS metamodel elements are organized in three domains: problem, agency, solution. In the first domain we put elements belonging to the model of the problem in terms or requirements, in the agency domain we collect elements defining an agent-based solution to the problem defined in the previous domain, in the solution domain we list elements related to the implementation of the solution in one or more available platforms (like JADE, JANUS, JACK and so on).
- The extension of the core MAS metamodel towards its completion (and the completion of the process obtained by composing fragments) is a crucial activity that should be strongly affected by the awareness of the new process requirements and the relationships among requirements and MMMEs. Despite this is the most important rule for completing the MAS metamodel (and its application strongly depends on the specific problem) another criterion should be considered as well:
- The opportunity of reusing some existing fragments leads to the introduction in the metamodel of all the MMMEs managed by them. The definition of new MMMEs that do not belong to the core metamodel is one of the most common consequences of that. This is a kind of bottom-up criterion that privileges the reuse of best known and tested fragments. Of course the acceptance of such new elements in the metamodel should be subjected to their participation in achieving the process requirements, otherwise a modification of the fragment (or the selection of another one) should be pursued.

Looking into the details of the proposed algorithm we can see that it starts with the selection of one of the three MMM domains (problem, agency, solution); after that three lists are initialized:

- The first list (*List_elements1*) is used for storing all the elements that can be defined by reusing fragments from the repository; a priority p is associated to

Table 1. The algorithm proposed for defining the instantiation priority of MAS metamodel elements

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 L_i (i=1,...,n) that: (i) can be instantiated by fragments of the repository and (ii) have less relationships with other elements
      i. Insert L_i (i=1,...,n) in List_elements1;
      ii. Remove elements L_i (i=1,...,n) from the core MMM;
      iii. p = p+1;
6. While the core MMM is not empty
   (a) select the leaves L_i (i=1,...,m) that can not be instantiated by fragments of the repository;
      i. Insert L_i (i=1,...,m) in List_elements2;
      ii. Remove L_i (i=1,...,m) from the core MMM;
7. For each element $E1_i$ of List_elements1 select an instantiating fragment from the repository *(verify the correspondence among fragment rationale and the process requirements/strategies)*
   (a) If one fragment corresponds to process requirements and strategies then:
      i. insert the fragment in the new process composition diagram
      ii. analyze inputs I_i (i=0,...,n) and outputs O_j (j=0,...,m) of the fragment
         A. If some I_i or O_j does not belong to the core MMM then add it to List_elements3; mark the fragment as "To be modified"
         B. remove $E1_i$ from List_elements1;
      iii. For each element $E2_i$ in List_elements2 analyze if there is a similarity with the elements defined in this fragment
         A. if yes delete $E2_i$ from List_elements2 and I_i/O_i from List_elements3
   (b) else (if no fragment correspond to requirements and strategies) then
      i. remove $E1_i$ from List_elements1 and insert it in List_elements2
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.

each element thus defining the prioritization order.

- The second list (*List_elements2*) is used for storing all the core MMMEs for the instantiation of whom we have no fragments in the repository; in this case it is not necessary to assign a level of priority to them as it will be discussed later.
- Finally, the third list (*List_elements3*) is used for storing the elements that are defined by the reused fragments but are not present in the core metamodel; this

list is used for establishing if the fragment introducing a specific MMME has to be modified or if it is better to extend the core MMM with this new element. For instance let us suppose a fragment for defining the MMME x has been reused; the same fragment could have another output MMME (suppose this is y) that does not belong to the core MMM; in this case two possibilities are available: *(i)* the core metamodel has to be extended by inserting y in it; *(ii)* the definition of element y could be similar to that of an element stored in *List_element2* (one for which no fragments are available in the repository). In any of these cases some modifications in the selected fragment are needed (the case study in subsection 4.3 will later show an example of that).

After the creation of these lists (step 5 of the algorithm) we start analyzing the core metamodel. The list of elements with priority p=1 let us identify the first element of the core metamodel to be instantiated in the new SEP thus giving us a base for selecting the first fragments from the repository.
These fragments have to be chosen among all those satisfying the process requirements and their associated strategies. A strategy is a set of actions, decisions, assumptions that are adopted in order to fulfill one of the process requirements. Table 2 will later report some of the strategies associated to the ASPECS process requirements.

A process component diagram (see the following subsection 4.4) is drawn in order to keep trace of all the selected fragments, their inputs, outputs and in order to facilitate their positioning in the previously selected life-cycle.

As it can be seen the algorithm entails some iterations, in fact it could be possible that after a first assembly an extension of the metamodel is needed; more specifically, this happens when the *List_elements3* is not empty (this is the list including the elements that are inputs or outputs of the adopted fragments).
Another opportunity for iterations occurs when the MAS metamodel alone does not lead to the completion of the process. This is something we have not directly experienced but we cannot exclude that the proposed approach ensures the definition of a complete process; it is possible to imagine that after some iterations the process is not definitive and it still needs some fragments in order to introduce activities or artefacts that are related to the process requirements but have not been connected to any MMME.

In the next subsection we will discuss the final phase of the new process construction approach presented in Figure 3.

3.3. *Process Deployment*

Once the new SEP has been created and enacted the method engineer evaluates it during the work of a system designer that adopts, during the *System Design* activity, the new process with the aid of a CASE tool for solving a specific problem.
After that the designed system is used and experimented; a results evaluation activity occurs (during *Process Evaluation*) in order to measure and evaluate the new

process, also according to the CMMI model. Gathered information can be used as new process requirements for a next iteration (if necessary).

In the following section we move from the presentation of the theoretical description of our approach to the experiment we used to prove that. We will present (some of) the requirements under which the ASPECS design process has been developed and the resulting process itself.

4. The ASPECS Process: From Initial Requirements to the Result

The ASPECS design process has been created to deal with the construction of large multi-agent systems used in problems where an hierarchical decomposition could be advisable. We also tried to reuse the six years of experiences done with the PASSI design process [10] and the social aspects of the CRIO approach [14].

In the following subsections we detail the identified requirements for the ASPECS process, the created core metamodel, the definition of the precedence order for the instantiation of metamodel elements, the selection/assembly of method fragments and the extension of the metamodel with the consequent selection of new fragments in an iterative process. This process is the instantiation of the general process described in section 3 and complements the theoretical part of this paper with the experiment we did in composing ASPECS.

4.1. *Requirements for the construction of ASPECS*

The design of the ASPECS methodology has been constrained by a set of requirements that according to the inputs of the process requirements analysis phase presented in subsection 3.1, can be classified as follows:

(1) Problem Type: the scope of the new design process was defined to be the development of very large MASs for the solution of problems suitable for an hierarchical decomposition. This requirement was inspired by previous experiences of some project members with holonic social structures.

(2) Development context: the development of the ASPECS methodology can be seen as a joint work of people coming from two different experiences: people working at the SET laboratory who had a strong background in the design and implementation of holonic systems with a strong accent on the organizational aspect of the system (CRIO process) and one new lab member who was the main author of a process (PASSI) for the design of MASs where agents are mostly peers and important features were: the use of ontologies, a requirements-driven agent identification, the adoption of patterns and tools for supporting the design/coding activities.
Participants to this project soon agreed to maintain the key elements of their background and skills in order to allow an easier adoption of the new design process. This implied reusing distinguishing elements from PASSI and CRIO. As regards agents implementation, in the SET lab, the development of a new

implementation platform (Janus) was undergoing and its adoption in the new design process was, of course, highly desirable.

(3) Organization maturity: several experiments for the development of holonic systems have been previously performed in the lab but each single project adopted a different implementation solution or design strategy so that a unique consolidated design process was not available.
Conversely, as regards the experiences coming from PASSI people, a complete documentation of the process was available, a large number of projects have been already developed and a large experience of usage of the process and the related guidelines/tools was available.
From previous PASSI design experiences, authors gained the conviction that an early adoption of an ontological description of the Problem domain could help in improving problem understanding and design quality.

These requirements concurred to the definition of the core metamodel we describe in the next subsection.

4.2. *The ASPECS core metamodel*

From the above described process requirements, we obtained the elements that compose the core metamodel.
A detailed description of the techniques and guidelines for relating process requirements with some MMMEs is out of the scope of this paper but we can briefly say that we adopted the following process: initially we identified one or more possible strategies for fulfilling each process requirement and after we selected the MMMEs that best fit these strategies. This latter selection, for the scope of this paper, can be considered the result of the method engineer experience.

In the specific case study this means that we selected the elements that were directly related to the adopted strategy according to our knowledge of the fragments in our repository as well as the structure of other well known design processes.

Table 2 reports the list of process requirements for the ASPECS process (column 1), the strategy (sometimes more than one) we selected for fulfilling this requirement (column 2) and finally the consequences of the selected strategy (columns 3-5 of the Table). Consequences are expressed in terms of MAS MetaModel Elements (MMMEs) coming from both the PASSI (column 3) and CRIO (column 4) metamodels or finally they can be specific guidelines to be adopted in building the process (column 5).

Just to consider a few example from Table 2, we can note the effect of the first ASPECS requirement on the final result. In order to face the development of very large MASs for the solution of hierarchically decomposable problems we decided to adopt a holonic decomposition of the problem. Holonic societies were part of the background of people working with CRIO and therefore it was a logical choice to adopt them for such an objective.
Moreover, holons perfectly fit this kind of problems and they allow the construction

Table 2. ASPECS process requirements, fulfillment strategies and related MAS MetaModel Elements (MMMEs), partial list.

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
Reuse of experiences done with CRIO	Adoption of Capacity for abstracting agent's behaviour and enabling service exchanges	Service	Capacity	
	Adoption of role as a primitive concept		Role	
Organization maturity	Enabling the adoption of design tools supporting the introduction of design patterns and automatic code generation			Supporting tools should be built ensuring pattern reuse and code production
	Early adoption of ontological description of problem domain			Ontology is defined early in the process

of very large MAS because during holon design it is sufficient to look at one specific abstraction level at a time thus lowering the complexity of this activity.
The consequences of this strategy are: (i) the adoption of the key elements of the CRIO metamodel supporting the holonic structure, and the decision to put the concept of organization at the center of the design process philosophy (in some way, this moved a little apart the importance of Agent concept).

The reuse of experiences done with PASSI and CRIO was another important requirement for the new process. This requirement generated several strategies; for

instance the decision to maintain the semantic structure of PASSI communications (that are FIPA-compliant) implied the adoption of concepts like Communication, Message, Interaction Protocol, Ontology.
Another strategy related to that implied the adoption of Role as a primitive concept; this introduced a great change in the philosophy of the new process. Agents were no more the basic composition elements of the new application, they were replaced by Roles (as it was in the CRIO approach).

Finally, an interesting discussion arose from the assessment we made of our design experiences in order to evaluate the maturity of our organization and the needed improvement: PASSI designers found interesting their experience in adopting an ontological description of the problems for modeling agents' knowledge and communication contents.
We concluded that this experience, although positive, had a limit in the fact that it did not explicitly used the ontological model for improving problem understanding and requirements capturing. For this reason we decided to perform an early modeling of Problem domain in form of an ontology that could be reused in several different aspects of the design.

From the methodological point of view an interesting point concerned the decision of adopting a transformational approach inspired by PASSI but coherent with the MDE [38] theories. This suggested the idea of organizing the metamodel in 3 domains: problem, agency and solution, each one corresponding to a different level of abstraction and design refinement from problem requirements capture to the final solution delivery.

Resuming the results of this part of the work, in Table 2 it is possible to see that the following MMMEs have been identified as parts of the new core metamodel:

- For the Problem Domain: Scenario, (Functional) Requirement, Organization, Role, Ontology (composed of Concept, Predicate, Action), Interaction, Capacity.
- For the Agency Domain: Message, Interaction Protocol, Communication, Holon, Service.

It is worth to note that we here report only a part of the overall set of ASPECS process requirements and corresponding consequences. For this reason, for instance, we are not dealing at all with the elements belonging to the Solution domain MAS metamodel. In the following we will only focus on the Problem domain part of the metamodel.

From this list of elements we initially defined a core metamodel whose Problem domain portion can be seen in Figure 4. Relationships among elements largely come from the original metamodels (PASSI and CRIO) but some interesting issues raised in the composition of these elements in the new core metamodel:

- Elements coming from CRIO have been integrated in the new model with only minor changes in their definitions (that is what happened to the Organization

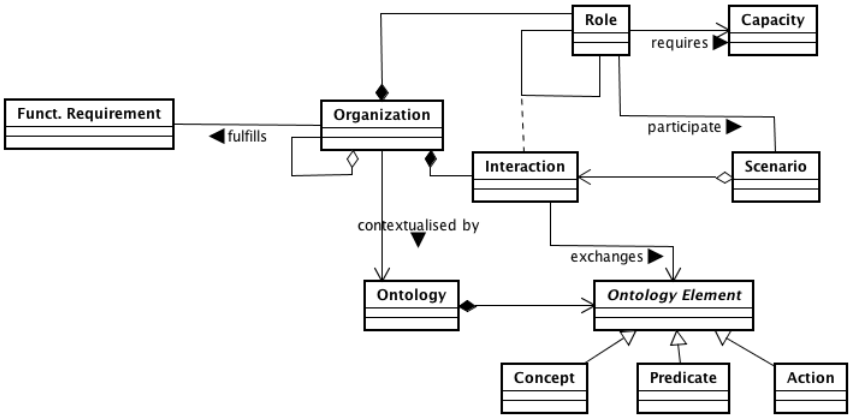

Fig. 4. The ASPECS Problem domain core metamodel.

and Interaction relationships with other elements)

- Two different concepts (Role in PASSI and CRIO) had the same name but different definitions. Essentially, as it is easy to see, the CRIO Role concept is an analysis level concept while the PASSI Role one is mostly a design abstraction. This is an example of the situation described in point 3 on the previously reported subsection on the core MAS metamodel composition (elements with same names and different definitions).
  In this specific occurrence we found an easy solution of the problem, by positioning each of the two elements in the domain it belonged to in its original approach, and introducing a relationship between them (i.e. the CRIO Role is transformed in the PASSI Role when moving from the Problem to the Agency domain).
- PASSI Requirement is usually related to the agent concept. This represents the fact that in PASSI agents are responsible for satisfying requirements. In the ASPECS process this responsibility is given to organizations as it comes from the CRIO process and the choices reported in Table 2. As a consequence the two concepts (Requirement and Organization) have been related and their definitions have been consequently modified.
- While in PASSI Roles participate to Scenarios, in the new model this situation has been replaced by the more detailed integration of the CRIO and PASSI contributions: Role interact each other (Interaction is an attribute class of the Role self-relationship) within Scenarios.
- Ontology has the same structure as in PASSI but it is now positioned in the Problem domain. This is the consequence of a precise choice: adopting ontological exploration of the Problem domain as a tool for deepening the understanding of the problem to be solved.

- Capacity has been introduced as a Problem domain abstraction for representing what each role is capable to do (it does not describe how to achieve the objective, for instance the adopted algorithm, it only describes what can be done).
- A new relationship has been introduced between Interaction and Ontology Element (an abstract class that generalizes the ontology components for purely commodity purposes). This relationship enriches the description of the information exchanged in interactions thus fully exploiting the possibilities offered by an early modeling of problem ontology.
- The original CRIO definition of Organization has been enriched by the adoption of an ontological description of the organizational context (again a consequence of the early availability of an ontology model).

From these and other similar considerations we built the core metamodel for the ASPECS process. It has not been an easy and short activity but rather it has been performed during several meetings, it involved debates with other people not directly belonging to the team of ASPECS developers but skilled in the usage of agent-oriented methodologies and related platforms.

In the next subsection we discuss the prioritization of the MMMEs that represents the order, we expect to adopt, for instantiating these elements in the fragments that will compose the new design process.

4.3. *Prioritization of MAS metamodel elements*

The priority order of the MMMEs has been defined by applying the already discussed algorithm (see Table 1).
In steps 1-4 we select the Problem domain part of the core MMM (shown in Figure 4), and we create the three lists.
In step 4, by analyzing the core MMM we see that *Capacity* complies to condition 5.a of the algorithm and it has only one relationship with the other elements; therefore we assign p=1 to *Capacity*, we introduce it in the first list (List_element1 = (*Capacity*, 1)), we remove this element from the core MMM and finally we increment p.
Since the metamodel is not empty, we continue the loop reported at step 5 and we find that the following elements have only two relationships with the remaining part of the metamodel and they also satisfies the condition at step 5.a: *Functional Requirement*,*Concept*, *Predicate* and *Action* (that are specializations of the abstract concept *Ontology Element*), and *Scenario*. As already done for *Capacity*, these elements are now introduced in the first list, removed from the core metamodel and the priority p is incremented.

At the end of the core MMM analysis and the application of steps 5 and 6, we have the following situation:

- List_elements1 = (*Capacity*, 1), (*Func. Req.*, 2), (*Concept*, 2), (*Predicate*, 2),

(*Action*, 2), (*Scenario*, 2), (*Ontology*,3), (*Role*, 4)
- List_elements2 =(*Organization*),(*Interaction*)
- List_elements3 =NULL.

As regard *List_elements2*, we introduced *Organization* and *Iteration* in it since we have not available fragments in our repository for instantiating them; in the following subsection it will be explained how they are managed. *List_elements1* and *List_elements2* cover all the elements of the Problem domain and the choice done reflects the ASPECS design process requirements.
Similarly we obtained a priority order list for the MMMEs of the following domains (Agency and Solution). This part of the case study application is omitted because of space concerns.

After these steps, it is possible to start the selection of fragments (step 7) from the repository or the construction of new ones in order to define the elements according to the prescribed order. This process will be discussed in the next subsection.

4.4. *Definition of an initial draft of the process*

In this subsection it will be explained how a first prototype of the new SEP is drawn following the second part of the algorithm presented in Table 1 (from step7).

In performing the fragments selection activity, we refer to our repository of fragments [39] that includes fragments extracted from PASSI[10], Agile PASSI [17], TROPOS [22], Adelfe [2] and some others coming from CRIO. Since several of the MMMEs required by this novel approach (for instance holon) are not present in the repository, we expect to produce several new process fragments, hoping of reusing and modifying some existing ones when possible.

According to what prescribed at step 7.a.i, the first process fragment to be considered for the construction of the new process is devoted to the definition of *Capacity*.

In order to draw the initial sketch of the process we adopt what we call a *process component diagram*. Two examples, at different levels of refinement, of this diagram are reported in Figure 5. In this diagram, each process fragment is reported as a component with some input and output MMMEs. If a component defines an element that is an input for another one, than a dashed line is drawn in order to show this dependency. Inputs of the process are here reported too (this is the case of the *Text Scenario* element that is an input of the Domain Requirements Description fragment reported in Figure 5.b).

In executing step 7.a.ii we consider the input/output MMMEs of the *Capacity Identification* fragment and we find that they all belong to the core MMM so that this fragment can be reused as it is and therefore marked as "Reused" (besides we fill it in green). The resulting process component diagram is reported in Figure 5.

During the second iteration of the loop reported at step 7 we deal with the second element of List_elements1 (*Functional Requirements*) and we select (step 7.a) the PASSI Domain Requirements Description fragment to instantiate that. A

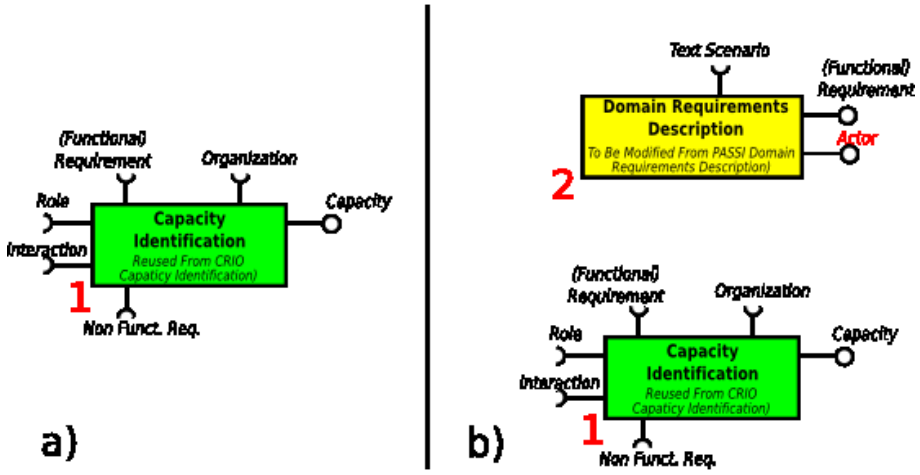

Fig. 5. The first and the second step of process component diagram construction

detailed list of the operations performed on that is now reported:

- *Step 7.a.i.*: the PASSI *Domain Requirements Description* fragment is introduced in the process component diagram, see Figure 5.b.
- *Step 7.a.ii.*: this fragment has an input (*Text Scenario*) that does not belong to the core MMM but it is considered as an input for the whole process and therefore we can avoid inserting it in *List_elements3* (no fragment will be necessary to define that).
- *Step 7.a.iii.*: no similarity with the elements of *List_elements2* is found.

The fragment has two outputs: the (obvious) *Functional Requirement*, and *Actor*; this latter element is not, actually, present in the core MMM and therefore it is to be inserted in *List_elements3*. The fragment is then marked as *To be Modified* and the Actor element is red coloured in the diagram.

By iterating the process again for two times, we obtain the process component diagram reported in Figure 6. About this result, it is interesting to note that:

- the third element (*Problem Ontology Description*) can be totally reused form PASSI as it descends from already done considerations.
- the fourth fragment,(*Scenario Description*), presents some elements to be inserted in the *List_elements3*, they are: *Actor*, *Agent* and *Message*.
- for the *Message* element, by following the step 7.a.iii, we find that the definition of *Message* is quite similar to that of *Interaction* and therefore we can remove that respectively from *List_elements2* and *List_elements3*. *Interaction* and *Message* will be considered as the same element from now on (the name *Interaction* will be used for both).

Now in *List_elements2* only the element “Organization” remains; by analyz-

ing it (step 8), we see that the fragment that should define it aims at creating a relationship between each organization and the requirements it is responsible to accomplish. This is very near to the work done in the PASSI Agent Identification fragment that can therefore be easily adapted to cope with this new situation (the resulting fragment will be labelled Organization Identification).

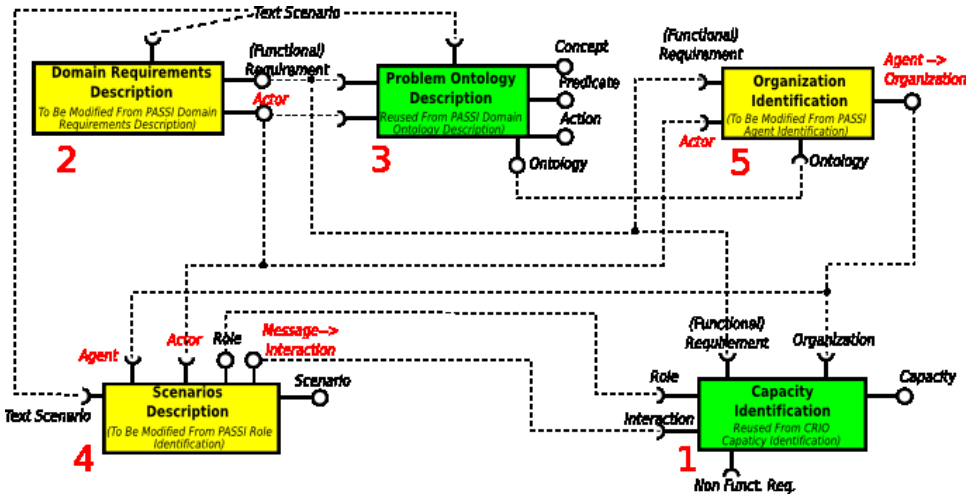

Fig. 6. The process component diagram obtained by the core MAS metamodel

In a similar way, the remaining part of the process has been defined but the corresponding description has been omitted because of space concerns. In the next section we discuss some examples of extension of the initial core MAS meta-model done in order to refine the initial sketch of the process.

4.5. *Completion of the process and extension of the core metamodel*

The construction of a new design process can be regarded as an iterative-incremental activity that can be decomposed in the following steps:

(1) Construction of a process stub (including several fragments, for instance up to reach the phase size);
(2) Evaluation of results;
(3) Next iteration planning (in terms of new process requirements to be addressed, changes to be done in the existing process stub, new parts of the metamodel to be included in the process).

In the previous sections we discussed the construction of the new process stub as reported at the beginning of the previous list. In the case of the ASPECS methodol-

ogy, we performed the first significant results on evaluation activity after completing the System Requirements phase.

This test, in our case, consisted in using the new process stub for designing a couple of simple applications. This allowed us familiarizing with the process and appreciating its qualities. Because of the high number of reused fragments and the good level of coherence and cohesion exposed by the process stub we only proposed a few changes. The first was the explicit introduction of non functional requirements in the early stages of the process. This has been done by introducing the Non Functional Requirement element in the MAS metamodel (directly related to the existing functional Requirement element) and then modifying the Domain Requirements Description fragment in order to complement the adopted use-case based description of requirements with a textual description of non functional requirements constraining the designed use cases.

Another change regarded the Actor element that is an output of the same Domain Requirements Description fragment, but it was not in the core MMM (it can be found in *List_elements3*). This is an element that could reasonably be considered as a part of the metamodel although not explicitly deducted from requirements. For this reason, it has been decided to include it in the final metamodel.

Whereas the above-described changes regarded the introduction of new elements in the metamodel, other modifications have been done in other aspects of the process. For instance, it has been decided to fully exploit the advantages of an ontological modelling of the Problem domain by using that for the identification of organizations. For this reason the Ontology input has been introduced in the Organization Identification fragment (it was not an input in the original PASSI Agent Identification fragment).

Finally, a totally new fragment has been introduced in this part of the process because of a simple consideration: the Scenarios Description fragment reports a dynamic view of scenarios as they are enacted by organization roles; it was considered convenient to introduce a structural view of these roles in order to facilitate the management of these elements; the *Interactions and Roles Identification* fragment was therefore introduced, its aim is to produce a class diagram reporting Roles as classes and Interactions as relationships among them. Finally, the elements detailing role plans (and the corresponding fragment) have been introduced. At this stage the first phase of the process (the one regarding the Problem domain) is completed.

The order in which the fragments are listed in the process is the consequence of their dependencies reported in the process component diagram (Figure 6 depicts the realization of the core metamodel without the above reported extensions of it). The final Problem domain metamodel of the ASPECS process is reported in Figure 7, whereas resulting ordered list of fragments is:

- Domain Requirements Description;
- Problem Ontology Description;
- Organization Identification;

- Interactions and Roles Identification;
- Scenarios Description;
- Role Plan;
- Capacity Identification.

After that, according to the 3-steps iterative process discussed at the beginning of this subsection, we designed a new portion of the metamodel, more specifically, the core part of the Agency domain metamodel. We are not now going to detail the work done for building the remaining phases of ASPECS since it is essentially the same done for the first phase.

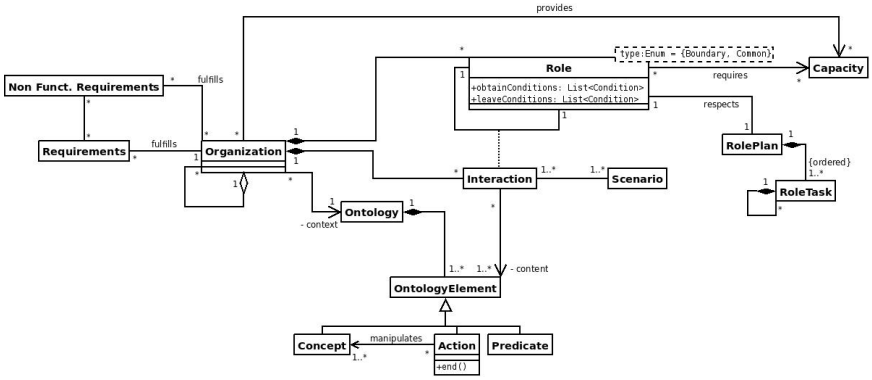

Fig. 7. A part of the ASPECS Problem domain core metamodel.

5. Discussion and Conclusions

In this paper an approach for the construction of customized agent-oriented design processes on the basis of the well known Situational Method Engineering paradigm has been presented; this approach is named PRoDe meaning Process for the Design of Design Processes).

PRoDe is composed of three phases: (*i*) *Process Analysis*, devoted to the process requirements elicitation; this phase starts from the maturity level of the organization, the problem type and the development context; it produces a set of elements useful for the construction of the core metamodel and the structure (the process model) to be adopted by the under construction process; *(ii) Process Design*, whose aim is to establish which process fragments have to be selected from the repository and how they can be assembled; *(iii) Process Deployment* where the new design process is used and evaluated. The construction process is iterated, if necessary.

Each PRoDe phase is composed of a set of activities (see Figure 3); in this paper we only gave a brief description of some of them, while we detailed the following: *Core Metamodel Creation*, *Reusable Fragments Selection* and *Fragments Assembly*.

This choice descends from the specific focus of this paper (the exploitation of the system metamodel in the new process construction). Our aim was, in fact, to explore how the core metamodel underpinned by the new process and its elements are used for the selection and assembly of fragments after its construction. In this sense we invert the classical point of view according to which the system metamodel is a consequence of the process and we adopt a MDE perspective where the metamodel is the focus of the development process.

One of the most relevant problems we face is to find a criterion for establishing the starting point for process fragments assembly. This starting point is expressed in terms of elements of the MAS metamodel. In our approach, each metamodel element is strongly connected to a process fragment since each fragment is devoted to produce a deliverable where, usually, at least one metamodel element is instantiated or related to one or more other elements. This assumption let us identify a list where metamodel elements are ordered according to a priority; this priority is a measure of the mutual dependency of MAS metamodel elements (number of relationships among them). In this way, for each element we can establish the order of instantiation during the process fragments selection phase. MAS metamodel elements with less relationships with the others are selected at first and corresponding process fragments (that instantiate them) are the first to be introduced in the new process. At this aim, we created the presented algorithm that constitutes a useful guide for method engineers during the construction of a new process. It is worth to note that the fact that a specific fragment is the first to be selected does not necessarily mean that this fragment will be positioned at the beginning of the process. This position is determined by considering mutual dependencies among fragments as described in Figure 6.

The proposed approach has been adopted in the construction of a new design process devoted to the development of holonic multi agent systems (ASPECS); in this paper we illustrated the work we did for that and namely: the construction of the core metamodel, the use of the prioritization algorithm, the selection and assembly of fragments.

We regard the use we did of the system metamodel, and the prioritization algorithm we proposed, as the first steps towards a solution to the problem of the lack of guidelines for some SME activities and for reducing the great dependency on the method designer personal skills. Another approach in literature faces the selection and assembly of fragments in a kind of formal way: this is based on the deontic matrices proposed in [27] and [25]; it provides the possibility to link fragments in pairs thus facilitating their assembly. This approach, although useful, is anyway directed to people skilled in using the OPF framework; this because the structure of deontic matrix itself, still requires a deep knowledge of the used repository; in this sense, we think our solution offers a viable opportunity to organizations where a not very experienced method engineer is available.

Our approach is surely to be refined and improved, it is still linked to the method

designers skills in the step 7 (of the algorithm), when the designer has to check if the selected fragment corresponds to process requirements and strategy; this point needs further guidelines and for now it could not be automated, whereas all the other steps offer the advantages of being a well defined guide also for not skilled people; it is only necessary to have a well documented repository of fragments.

This approach was at first conceived for being used in the agent-oriented software engineering context, but the use of system metamodel as the basis of it, allows us to affirm that it can be considered general, since it can be used in all the contexts where a system metamodel is defined (this actually means all the approaches based on the MDE paradigm). We found some similarities and some possibilities of interrelation, with the approach shown in [43] where a *process-data diagram* is used to relate activities of the process under construction with the elements (data) they produce; data is then used by the method designer for choosing, on the basis of his/her knowledge, which fragment has to be used.

This approach is quite similar to the one we proposed in this paper, it is not focussed on the metamodel but it can be easily brought back to that thus strengthening our conviction about the generality of our work.

As a future work we are planning to extend the algorithm, best detailing the procedure for matching selected fragments to the process requirements and also we aim at studying the best way for representing and realizing the modifications required from fragments during assembly.

Acknowledgements

Part of this work makes use of results produced by the PI2S2 Project managed by the Consorzio COMETA, a project co-funded by the Italian Ministry of University and Research (MIUR) within the Piano Operativo Nazionale "Ricerca Scientifica, Sviluppo Tecnologico, Alta Formazione" (PON 2000-2006).

References

[1] The Model Driven Architecture. http://www.omg.org/mda/.

[2] Carol Bernon, Valérie Camps, Marie-Pierre Gleizes, and Gauthier Picard. Engineering adaptive multi-agent systems: the adelfe methodology. In *Agent Oriented Methodologies*, chapter VII, pages 172–202. Idea Group Publishing, 2005.

[3] S. Brinkkemper. Method engineering: engineering the information systems development methods and tools. *Information and Software Technology*, 37(11), 1996.

[4] S. Brinkkemper, K. Lyytinen, and R. Welke. Method engineering: Principles of method construction and tool support. *International Federational for Information Processing 65*, 65:336, 1996.

[5] S. Brinkkemper, M. Saeki, and F. Harmsen. Meta-modelling based assembly techniques for situational method engineering. *Information Systems, Vol. 24*, 24, 1999.

[6] S. Brinkkemper, M. Saeki, and F. Harmsen. A Method Engineering Language for the Description of Systems Development Methods. *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, pages 473–476, 2001.

[7] S. Brinkkemper, R.J. Welke, and K. Lyytinen. *Method Engineering: Principles of Method Construction and Tool Support*. Springer, 1996.
[8] Atkinson C. and Kuhne T. Model-driven development: A metamodeling foundation. *IEEE Software*, 20(5):36–41, September/October 2003.
[9] L. Cernuzzi, M. Cossentino, and F. Zambonelli. Process models for agent-based development. *Engineering Applications of Artificial Intelligence*, 18(2):205–222, 2005.
[10] M. Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies* [28], chapter IV, pages 79–106.
[11] M. Cossentino, S. Gaglio, A. Garro, and V. Seidita. Method fragments for agent design methodologies: from standardisation to research. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 1(1):91–121, 2007.
[12] M. Cossentino, S. Gaglio, L. Sabatucci, and V. Seidita. The passi and agile passi mas meta-models compared with a unifying proposal. In *In proc. of the CEEMAS'05 Conference*, pages 183–192, Budapest, Hungary, Sept. 2005.
[13] M. Cossentino, S. Gaglio, and Seidita V. 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)*, 2007.
[14] M. Cossentino, L. Sabatucci, V. Seidita, and Gaglio S. An agent oriented tool for new design processes. In *Proceedings of the Fourth European Workshop on Multi-Agent Systems (EUMAS'06)*, 2006.
[15] M. Cossentino, L. Sabatucci, V. Seidita, and Gaglio S. An expert system for the design of agents. In *Proceedings of International Workshop on Agent Supported Cooperative Work at the IEEE The Second International Conference on Digital Information Management (ICDIM'07)*, 2007.
[16] M. Cossentino and V. Seidita. Composition of a New Process to Meet Agile Needs Using Method Engineering. *Software Engineering for Large Multi-Agent Systems*, 3:36–51, 2004.
[17] M. Cossentino and V. Seidita. Composition of a New Process to Meet Agile Needs Using Method Engineering. *Software Engineering for Large Multi-Agent Systems*, 3:36–51, 2004.
[18] Massimo Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies* [28], chapter IV, pages 79–106.
[19] D.G. Firesmith and B. Henderson-Sellers. *The OPEN Process Framework: An Introduction*. Addison-Wesley, 2002.
[20] N. Gaud, S. Galland, V. Hilaire, and Koukam A. An organisational platform for holonic and multiagent systems. In *In Proceedings of Programming Multi-Agent Systems (PROMAS) workshop*, 2008.
[21] C. Gerber, J.H. Siekmann, and G. Vierke. Holonic multi-agent systems. Technical Report DFKI-RR-99-03, DFKI - GmbH, 1999.
[22] P. Giorgini, J. Mylopoulos, and R. Sebastiani. Goal-oriented requirements analysis and reasoning in the Tropos methodology. *Engineering Applications of Artificial Intelligence*, 18(2):159–171, 2005.
[23] D. Gupta and N. Prakash. Engineering Methods from Method Requirements Specifications. *Requirements Engineering*, 6(3):135–160, 2001.
[24] A.F. Harmsen, M. Ernst, and U. Twente. *Situational Method Engineering*. Moret Ernst & Young Management Consultants, 1997.
[25] B. Henderson-Sellers. Process Metamodelling and Process Construction: Examples Using the OPEN Process Framework (OPF). *Annals of Software Engineering*, 14(1):341–362, 2002.
[26] B. Henderson-Sellers. Creating a comprehensive agent-oriented methodology-using

method engineering and the OPEN metamodel. *Agent-Oriented Methodologies*, pages 368–397, 2005.

[27] B. Henderson-Sellers. Method engineering: Theory and practice. In D. Karagiannis and editors Mayr, H. C., editors, *Information Systems Technology and its Applications.*, pages 13–23, 2006.

[28] Brian Henderson-Sellers and Paolo Giorgini. *Agent Oriented Methodologies.* Idea Group Publishing, Hershey, PA, USA, June 2005.

[29] V. Hilaire, A. Koukam, P. Gruer, and J.P. Müller. Formal specification and prototyping of multi-agent systems. In *ESAW*, number 1972 in LNAI, 2000.

[30] K. Kumar and R.J. Welke. Methodology engineering: a proposal for situation-specific methodology construction. *Challenges and Strategies for Research in Systems Development*, pages 257–269, 1992.

[31] I. Mirbel and J. Ralyté. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11(1):58–78, 2006.

[32] L. Osterweil. Software processes are software too. In *Proceedings of the 9th international conference on Software Engineering*, pages 2–13, 1987.

[33] J. Ralyté. Towards situational methods for information systems development: engineering reusable method chunks. *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education*, pages 271–282, 2004.

[34] J. Ralyté and C. Rolland. An Approach for Method Reengineering. *Conceptual Modeling-ER 2001: 20th International Conference on Conceptual Modeling, Yokohama, Japan, November 2001: Proceedings*, 2001.

[35] S. Rodriguez, V. Hilaire, and K. Koukam. Formal specification of holonic multi-agent system framework. In *Intelligent Agents in Computing Systems, ICCS(3)*, number 3516 in LNCS, pages 719–726, 2005.

[36] Shavrin S. Ontological multilevel modeling language. *International Journal Information Theories & Applications*, 14, 2007.

[37] M. Saeki. Software specification & design methods and method engineering. *International Journal of Software Engineering and Knowledge Engineering*, 1994.

[38] Douglas C. Schmidt. Model-driven engineering. *Computer*, 39(2):25–31, Feb. 2006.

[39] V. Seidita, M. Cossentino, and S. Gaglio. A repository of fragments for agent systems design. *Proc. Of the Workshop on Objects and Agents (WOA06)*, 2006.

[40] V. Seidita, J. Ralyté, B. Henderson-Sellers, M. Cossentino, and N. Arni-Bloch. A comparison of deontic matrices, maps and activity diagrams for the construction of situational methods. In *CAiSE'07 Forum, Proceedings of the CAiSE'07 Forum at the 19th International Conference on Advanced Information Systems Engineering.*, pages 85–88, Trondheim, Norway, 11-15 June 2007.

[41] ter Hofstede A.H.M. and Verhoef T.F. On the feasibility of situational method engineering. *Information Systems.*, 22(6/7):401–422, 1997.

[42] Juha-Pekka Tolvanen. Incremental method engineering with modeling tools: Theoretical principles and empirical evidence (ph.d. thesis). *Jyvskyl Studies in Computer Science*, page 301, 1998.

[43] van de Weerd I., Brinkkemper S., Souer J., and Versendaal J. A situational implementation method for web-based content management system-applications: Method engineering and validation in practice. *In Software Process: Improvement and Practice, John Wiley & Sons, Ltd.*, 2006.

[44] Franco Zambonelli, Nicholas Jennings, and Michael Wooldridge. Multiagent systems as computational organizations: the gaia methodology. In *Agent Oriented Methodologies* [28], chapter VI, pages 136–171.