Process Engineering and AOSE

Massimo Cossentino, Marie-Pierre Gleizes, Ambra Molesini, Andrea Omicini

Processes Engineering and AOSE

Massimo Cossentino[1], Marie-Pierre Gleizes[2],
Ambra Molesini[3], and Andrea Omicini[3]

[1] ICAR CNR, Viale delle Scienze, ed. 11, 90128 Palermo, Italy
cossentino@pa.icar.cnr.it
[2] SMAC team, IRIT, University of Toulouse,
118 Route de Narbonne, F-31062 Toulouse Cedex 9, France
Marie-Pierre.Gleizes@irit.fr
[3] Alma Mater Studiorum – Università di Bologna
viale Risorgimento 2, 40136 Bologna, BO, Italy
via Venezia 52, 47521 Cesena, FC, Italy
ambra.molesini@unibo.it, andrea.omicini@unibo.it

Abstract. Agent-oriented methodologies like ADELFE, ASPECS, INGENIAS, MaSE, PASSI, Prometheus, SODA, or Tropos propose development formulae with their own specificities. Analyzing them is the responsibility of the Process Engineering discipline, which is currently one hot research line in software engineering. The analysis makes it possible to construct a catalogue of current processes, assessing their utility and enabling their reuse. Additionally, the study may lead to the modification or improvement of existing development processes, perhaps combining fragments from solutions coming from the different methodologies.

In this paper, we first provide a general view over the area of Software Process Engineering (SPE), then focus on the most recent developments of SPE in the Agent-Oriented Software Engineering (AOSE) field.

Keywords: Agent-Oriented Software Engineering, Methodologies, Software Process Engineering, Fragment.

1 Software Processes

In Software Engineering (SE), developers expects to find concrete "instructions" and methods to complete development before a deadline, delivering a quality product, and with a cost meeting the initial budget of the project. Such instructions are typically expressed in the form of a software *development process*. Development processes have often appeared in Agent Oriented Software Engineering (AOSE) methodologies as a simple enumerated list of steps. While this can be considered effective for small developments – e.g. one person during a short period of time –, it is hardly applicable to medium-to-big developments— e.g. a development team with a project several years long. The literature suggests using more complex solutions, like Cernuzzi et al. [1] or Fuggetta [2].

Cernuzzi et al. [1] define the development process as "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". Fuggetta [2] proposes an interesting definition from the organizational point of view of *software development process* (or simply *software process*) as "the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain (evolve) a software product". Software processes can then be (and typically are) composed by a set of stages, each specifying which phases/activities should be carried on and which roles (i.e.: client, analyst, software architect, programmers, etc.) and resources are to be involved in them.

Such concepts are seldom found in existing agent-oriented methodologies. Typically, new AOSE methodologies get proposed without explicitly relating them to any process models and, at the same time, being implicitly suitable only for a limited set of (or a single) process models. This contrasts with the fact that designing multi-agent systems is a complex software development, and that developers need to be guided and helped in this task. Therefore, agent-oriented methodologies should also take into account the fact that the product is not merely developed but also: (i) conceived, often relying on unstable or incomplete requirements; (ii) deployed, i.e., put to work in an operational environment; (iii) maintained and evolved, depending on novel requirements or changes in the operational environments. Besides, intuitively, different needs call for different processes. For example, it is quite clear that the way a multi-agent system (MAS) dedicated to solve a specific problem such as a timetabling [3] and a MAS which has to simulate a natural phenomenon [4] involve different development approaches.

Another important aspect to be considered is that a software process is not something static that, once adopted, should never be changed. Instead, a process can be improved with the experience of applying it in a concrete development. For instance, PASSI has been modified in order to also handle multi-agent simulation [5], while Tropos and ADELFE are combined in order to take into account self-adaptation in Tropos [6]. As a consequence, any agent-oriented software engineer should assume the development process evolves over time. It evolves together with the increased awareness of the involved people, towards a maturity level[1] that ensures the repeatability of a process in terms of the quality, cost and time of the produced software. This is a fundamental evaluation criterion for any organization that would aim at adopting the agent-based paradigm in its development process.

As a conclusion, the development process is an important element of any software engineering methodology—and is essential within any agent-oriented methodology; it attends the needs of a concrete development problem and evolves when the change brings benefits. Therefore, the study of the many aspects of the software development process from a scientific perspective is mandatory.

Along this line, in this paper we adopt *Software Process Engineering* (SPE) as the conceptual and technical framework for such a study, then discuss its application to AOSE. In short, SPE provides the tools for analyzing, decomposing, and

[1] CMMI, `http://www.sei.cmu.edu/cmmi/tools/cmmiv1-3/`

building software development processes. Accordingly, Section 2 briefly describes SPE under the classical SE perspective, along with the main concepts used in process composition. Then, Section 3 presents SPE from the AOSE point of view, and expounds the recent works done on fragments and process engineering customization.

2 Software Processes Engineering

Like most processes concerning human activities, the software development process is inherently complex and hard to standardize, as demonstrated by the difficulties reported in the literature of automating the software process. Despite the great acceptance of development processes based on the Unified Process (UP [7]), researchers and practitioners in the area agree that there is no single software process that could fit the need of all [8]. UP itself is essentially a highly-customizable development framework rather than a well-established and fixed process.

Among the current method engineering approaches in the literature, Situational Method Engineering (SME) is undoubtedly one of the most promising: there, the construction of ad-hoc processes is based on reusing portions of existing or newly-created processes, called *fragments*. So, in the reminder of this section we first discuss the basis of SME (Subsection 2.1), then we sketch the diverse fragment definitions (Subsection 2.2) and fragment repositories (Subsection 2.3), finally we briefly introduce some tools supporting SME (Subsection 2.4).

2.1 Situational Method Engineering

Situational Method Engineering [9,10] is the discipline that studies the composition of new, ad-hoc software engineering processes for each specific need. This is based on the assumption that the "situation" is the information leading to the identification of the right design approach. This paradigm provides tools for the construction of ad-hoc processes by means of the reuse of existing process fragments (called method fragments), stored in a repository (Subsection 2.3). In order to be effective, this approach requires the process and its fragments to be suitably modeled.

Within the concept of situation authors usually include: requirements of the process, development context, the specific class of problem to be solved. Several approaches to SME have been presented in the last years with no specific reference to the OO context [8,11,12,13,14].

The most complete approach in the field is probably represented by the OPEN Process Framework (OPF [15]). OPF is based on a large number of method fragments stored in a repository along with a set of construction guidelines that are considered to be parts of existing methodologies and can be used to construct new methodologies. The OPF meta-model is composed of five main meta-classes [16]: Stages, Producers, Work Units, Work Products and Languages. When instantiated, each meta-class produces a method fragment.

2.2 Fragment Definition

Different approaches to process composition may be also rely on different definitions (and labels) for the building blocks. We may attempt a rough categorization of process reusable parts defined in classical SE:

Method Fragment — A method fragment is a piece of an information systems development process. According to Brinkkemper et al. [11,17], there are two kinds of method fragments: the product fragment – concerning the structure of a process product, representing deliverables, diagrams, ... – and the process fragment—describing the stages, activities and tasks to be performed to produce a product fragment.

Method Chunk — According to J. Ralyte et al. [12,18], a method chunk is a consistent and autonomous component of a development process. The method chunk integrates two aspects of the method fragment, the product and the process, so it represents a portion of process together with its related product(s).

OPF Method Fragment — An OPF method fragment is an instance of one of the following classes: Stages, Producers, Work Units, Work Products and Languages. According to D. Firesmith and B. Henderson-Sellers [15,19], fragments of different types need to be composed in order to provide the different features of a process.

2.3 Fragment Repository

A *fragment repository* (or *method library*) is an essential component of any SME approach. In spite of this, only few repositories are currently available, due to the great effort required to build such a resource as well as to the lack of a widely-accepted standard in the field.

A repository could be used as a general purpose software engineering processes container. It could even contain sets of reusable elements (content and process) that do not belong to a specific software engineering processes, but could be used as building bricks (fragments/chunks). The largest available repository is part of OPF [15][2]. The support for a method library is also present in a widespread software tool, the Eclipse Process Framework[3] (EPF) plugin, which supports the OMG Software Process Engineering Metamodel (SPEM) v. 2.0 [20], and provides all the capabilities needed to define method plugins into library as well as to tailor new software engineering processes from those plugins (method configuration). A project such as OpenUP[4] provides an open-source, common and extensible base for iterative incremental software engineering processes by using the modularity and re-usability skills of SPEM 2.0 through the EPF plugin.

[2] `http://www.opfro.org/`
[3] `http://epf.eclipse.org/`
[4] `http://epf.eclipse.org/wikis/openup`

2.4 Tools

When developing a software system, several tools are required to support the different stages of the process. Areas of application for design tools spread from requirements elicitation to design, testing, validation, version control, configuration management and reverse engineering. In this section, not all tools available during the engineering process are described. We rather would like to focus on the different kinds of tools linked to apply or adapt a method. Tools can be classified in three different categories: CASE, CAME, and CAPE tools.

The CASE acronym stands for Computer Aided Software Engineering, and it addresses the large category of tools that could be used in any software engineering process employment. CASE tools support the modeling activities and constrain the designer just in the choice of the system modeling language (for instance UML), and, when code generation is possible, on the coding languages. The main limit of these tools is that they are not aware of the adopted method – in terms of work to be done – and are instead only concerned with the representation of some (often uncoordinated) views of the system.

Today, in the field of aided software development, meta-CASE tools achieved large significance, being able to provide an either automated or semi-automated process for the creation of CASE tools, based on a meta-model used to describe the language, the concepts and the relationships of a specific design methodology. In the field of Method Engineering a meta-CASE tool – called CAME (Computer Aided Method Engineering) – is used to describe and to represent a set of methods. CAME tools are conceived to support methods rather then design. They do not adopt any specific software development process model – they are not even aware of its existence because they are only concerned with the drawing of the different aspects of the model separately –, therefore the designer could freely work on the different views, even violating the prescribed process without any warning from the tool.

Several existing CAME tools (Mentor [21], Decamerone [22], MetaEdit+ [23], INGENME[5] and MethodBase [24]) are also based on Meta-CASE technology allowing the construction or the automatic generation of CASE tools specific for given methods. In particular, MetaEdit+ seems the most complete CAME tool, even if it presents several limitations. MetaEdit+ makes it possible to implement a domain specific modeling language, and to use it in ad-hoc created CASE tool. MetaEdit+ is at the same time a CAME and a CASE tool, and is the only tool that allows the instantiation of a CASE tool starting from the definition of a given modeling language. However, MetaEdit+ does not provide any support for managing the development processes, and the generated CASE tools always adopt the same UML editor for the software product design. As a result, the generated methods are static, and there is no way to reuse portions of existing tools supporting the method fragments used. A possible alternative to MetaEdit+ is INGENME, a tool for producing self-contained visual editors for languages defined using an XML file. INGENME can be used to test visual languages and also to produce customized editors.

[5] `http://ingenme.sourceforge.net/`

One of the intrinsic limits of CAME tools – the lack of process awareness – is overcome by CAPE (Computer Aided Process Engineering) tools that are instead aware of the adopted process model – or, that could be used to design it –, and coordinate the different stages of the process in order to respect its prescriptions.

3 AOSE Software Processes Engineering

Putting the focus of process engineering on the development of multi-agent systems moves the attention of the designer to the study of specific topics such as:

- the definition of the agent concept (that is often specific to each single approach),
- the definition of the MAS meta-model (composed of the already cited agent as well as many other entities like role, communication, group, and so on),
- the agent's reasoning technique (goal-oriented, expert system-based, rule based, ...),
- the implementation of autonomy, self-organization and similar system features.

Such topics affect the conception of the AOSE design process. Just to provide an example, it is nowadays very uncommon to find the adoption of formal languages in the development of object-oriented system. Conversely, this is quite frequent in conception of MAS where such languages provide a good support for the design of some reasoning techniques. This obviously influences the developing process since there is the need of several different new activities such as model checking and verification.

Besides, the absence of a standardized and widely accepted MAS meta-model has deeply influenced the introduction of SME techniques in the AOSE field. As a consequence, several authors centered their approach on the influence that the definition of a specific MAS type can have on the process that have to be followed in order to analyze and design such a MAS type. More details about such a variant of the classical SME approach are provided in the next subsection.

3.1 AOSE Situational Method Engineering

As far as the approaches specifically related to the agent-oriented context are concerned, an initial reference framework has been provided by the work of the FIPA Methodology Technical Committee (TC)[6] devoted to the study of fragments' definition and composition [25]. A standard specification is expected by the successor of that committee, the IEEE FIPA Design Process Documentation and Fragmentation (DPDF) working group[7]. The DPDF WG approach is based

[6] `http://www.fipa.org/activities/methodology.html`
[7] `http://www.fipa.org/subgroups/DPDF-WG.html`

on the adoption of SPEM 2.0 with the inclusion of minor extensions motivated by the specific needs of a multi-agent system design process [26].

SME in AOSE shares the same objective with SME researchers in proposing the most relevant process for a given situational context of development. The objective is to provide CAPE tools enabling to build the most convenient process and possibly to adapt it during the development. In fact, the most relevant fragments must be retrieved and used among all the available fragments. Consequently, some of the works in the AOSE community currently focus on how to automate the software process construction such as the three following examples reported here.

PRoDe: A Process for the Design of Design Processes. PRoDe [27] is an approach for new AOSE design process composition based on two pillars: the process fragment [28] and the MAS meta-model. These two elements are both defined and considered under a specific agent-oriented perspective thus creating a peculiar approach. PRoDe is based on the classic situational method engineering and it is organized in three main phases: Process Analysis, Process Design and Process Deployment. They clearly resemble the software development phases thus realizing the parallelism proposed by Osterweil in his well-known paper *Software Processes are Software too* [29].

During Process Analysis, the process to be developed is analyzed and its requirements are elicited. Process Requirements Analysis delivers a portion of the MAS meta-model, whose structure will decisively affect the following steps of process development. During Process Design the method engineer selects, from a previously constructed repository, a set of fragments that he/she assembles in the new process. Finally, in the Process Deployment phase, the new process is used to solve a specific problem. From this employment some feedbacks are received and this is used to further enhance the process towards its maturity. It is also possible to repeat the whole construction process in an incremental/iterative way.

The most relevant contribution coming from PRoDe to the state of the art consists in some guidelines driving the method designer through the most difficult steps of the work. This is a very relevant aspect of the approach since skills required to a method engineering are very high and such a professional profile is not frequently found in the field. The most relevant guideline is realized by an algorithm used to prioritize the retrieval of fragments from the repository. The problem solved by the algorithm is: given a MAS meta-model and the set of fragments able to instantiate the elements of the meta-model, which is the first fragment that should be selected? And the following? This problem is relevant because the selection of one fragment (the first) introduces new constraints in the design: its inputs are to be satisfied and its outputs should be all used otherwise the fragment needs an adaptation (an adjunctive cost).

Currently the process has been already adopted in several case studies [30,31] and it is at the basis of the approach proposed in the next subsection.

MEnSA Project. In the MEnSA project[8] the authors decided to directly link the new process requirements to the fragments they were going to select. The composition of the new methodology was inspired by the PRoDe approach [27], which proposes to use the MAS meta-model as a central element for selecting and assembling fragments. It is worth to note that while in the PRoDe approach requirements are used to compose an initial draft of the MAS meta-model, which is then used to retrieve fragments from the repository, in the novel MEnSA approach process requirements are used to select fragments, and their outcomes are used to define the MAS meta-model.

The approach adopted is organised in a few steps. The first step of the work consists in collecting process requirements, currently some methodologies have been decomposed in fragments: PASSI, Gaia, Tropos and SODA. Then, fragments are retrieved from the repository according to the requirements they contribute to fulfill.

The *fragments selection* activity makes available the set of fragments used to produce a first draft of the MAS meta-model. Thus, each fragment contributes to define a portion of the meta-model.

Once the meta-model has been polished, the initial set of fragments finds its position in a proper life-cycle, therefore a proper process model has to be chosen. Classically-available life-cycles (waterfall, iterative/incremental, spiral, etc.) are here considered, and the best fitting is used to host the selected fragments.

Now fragments can be positioned in the life-cycle placeholders, and a first version of the new process is almost ready. The last activity is *fragments adaptation*, which aims at solving incompatibility issues arising fragments from different processes. Such fragments should then be adapted to properly support the new MAS meta-model and to comply with all input/output constraints.

At this stage, an initial version of the process is finally available. This could be either complete or incomplete according to the number and refinement of the initial process requirements, as well as to other factors—fragment repository dimension, assembly issues, process phases coverage, ...

In the MEnSA process composition, when the process needs to be completed, the authors follow up with an iteration of the proposed composition approach, given its smaller granularity and its MAS-meta-model-based approach that perfectly fits the needs of the new process final refinement.

Self-Organisation-based SPE Design. The aim of this approach is to provide a system able to combine existing fragments to build a software process adequate to the expertise level of designers and to the applications features as well. The designer may interact with the system in order to modify the software process proposed by the system. The SPE is co-constructed by the system and the designer.

The system continuously self-adapts to these new perturbations and proposes a new software process taking into account the designer's wishes. Each fragment

[8] Methodologies for the Engineering of complex Software systems: Agent-based approach. For more details, see the project website at:
`http://www.mensa-project.org/`

is *agentified* following the Adaptive Multi-Agent Systems (AMAS) theory [32]. The adaptive MAS automatically designs an adaptive software engineering process [33]. The resulting MAS is composed of (i) the fragments, which are the sole agents of the MAS (called fragment-agents), and (ii) the resources of the MAS, which are the MMMEs (MAS Meta-Model Elements), the MMME repository and the fragments repository.

The first MAS prototype is developed using a repository containing the already-defined fragments of three processes from three methodologies: ADELFE [34], INGENIAS[9], and PASSI [35].

In this system, fragments are autonomous agents able to find other relevant fragments to interact (workproduct exchanges). The fragment-agents cooperate with each other in order to find their right location in the software process. The agentification of fragments is realized in a general way to enable the adding or removal of a fragment in the set of all fragments. In the first version of the system, agentification must ensure a mutual understanding between what is required by a fragment and what is produced by it—i.e., the agent-fragments understand each other.

The behavior of a fragment is represented by a finite state automaton with three states: *Inactive*, *Unsatisfied* and *Satisfied*. The fragment-agent switches states according to its perception of the environment and behaves according to its current state. Following the AMAS theory, the design of agents focuses in particular on the Non Cooperative Situations (NCS) [32]. Encountering a NCS is one of the possible reasons for a fragment-agent to change its state. For a fragment-agent, three NCS are identified. The first is when none of the MMMEs that a fragment-agent can produce are needed by the other fragments (the agent is *useless*). The second is when a fragment-agent needs some MMMEs that no other fragment-agent can produce (the agent *cannot satisfy its preconditions*). The third is when two fragment-agents of the same process produce the same MMMEs (agents are in conflict, and must determine which one of them is *useless*). To react to such NCS, a fragment-agent can execute three main actions. First, it can use existing MMMEs to produce new ones and register them to the repository. Or, it can stimulate other fragments able to produce their needed MMMEs by sending a stimulation value. This value depends on the amount of stimulation it received. So, the stimulation value of a fragment-agent can be assimilated to a measure of its criticality, since the more it is important to the system, the more a fragment-agent will be stimulated. Finally, it can observe other agents and determine its membership to a process.

3.2 Fragment Definition

As far as the agent-oriented approaches are concerned, some contributions for the adoption of the OPF framework [36] to agent design have been proposed such as in PASSI [37] and Tropos [38]. A different solution was proposed by the FIPA Methodology TC for the adoption of a kind of method chunk, called *process fragment*, which is a portion of a development process defining deliverables

[9] `http://grasia.fdi.ucm.es/main/node/241`

(workproducts), guidelines, preconditions, sets of system meta-model elements and key-words characterizing it in the scope of the method in which it was defined. The main difference with respect to the OPF lays in the focus of the MMMEs managed by the process fragment. The definition and use of a MAS meta-model enables the adoption of Model-Driven Engineering practices, and overcomes typical problems due to the confusing definition of agent-oriented concepts in most AOSE methodologies. Besides, considering fragments through such a rough definition eases the adoption of SPEM 2.0 concepts as well as a high level of compliance to that standard. However, [25] presents an enhanced version of the fragment meta-model originated by the FIPA Methodology TC; one of the proposals is that fragments should be considered from different points of view whether the designer is interested in their reuse, storing, implementation, or in the definition of the process they represent. So, it seems obvious that the choice of a process fragment depends on the point of view and the requirements it has to fulfil.

Process fragments do not match a single SPEM 2.0 concept, instead they should rather be considered as matching several ones depending on the granularity or concern. A process fragment is a portion of a process, but process element definitions in SPEM 2.0 are divided into two categories: definition and use. Thus, such a separation should be taken into account while mapping fragments to SPEM 2.0 concepts. Furthermore, also fragments granularity should to be considered, as it implies a different mapping.

It is worth to note that although the use of SPEM is widespread in the SE community, a new standard has been recently published by ISO [39,40], which models both the design and enactment of a process by using a multilayered architecture. The peculiarity of that work lays in the adoption of *powertypes*, an innovative mechanism allowing a class to assume value for its attributes not necessarily when instantiated in the next abstraction level but, if required, at the second level [41]. A similar standardization effort, in the AOSE field, is currently ongoing within the IEEE FIPA DPDF working group. The first step has been the identification of the most suitable process meta-model and notation: (i) for the representation of the existing design processes from which the fragments have to be extracted, and (ii) for the representation of fragments themselves. An important contribution on the subject might come from the SPEM 2.0 specification.

The second step has been consisted in the definition of a proper template for the description of agent-oriented design processes. Such a template refers to the selected process meta-model and suggest the adoption of good practices in documenting existing processes as well as defining new ones. The availability of such a specification would have several benefits, the first is that adopting the same documentation template would enable an easier comprehension (and comparison) of existing and new processes. This goes in the direction initially drawn by UML creators (Booch, Jacobson, and Rumbaugh) when they first removed all the specific notation issues cluttering their own approaches and then easily found commonalities that inspired their new (and successful) modeling

language [42]. The specification for process documentation has been already proposed for adoption as an IEEE FIPA standard.

After that, the group is going to define the process fragment structure and according to that a procedure for extracting fragments from processes documented according to the adopted template. The final result will consist in a set of fragments that are compliant to the fragment specification and are documented according to the same style. This would enable their composition and the production of a consistent documentation of the new process.

3.3 Fragment Repository and Tools

Some extensions of the OPEN framework [15] aiming at including the support for agent-oriented methodologies have been recently presented in [36,37,38,43,44,45]. Another (explicitly agent-oriented) repository[10] has been developed according to the specifications proposed by the FIPA Methodology TC [46], used as a starting point by the IEEE FIPA DPDF working group.

The proposed repository structure is an XML-based repository storing a collection of XML documents, each one representing a method fragment, validated by a Document Type Definition (DTD) or an XML Schema [25]. The validation process ensures that the method fragment was extracted and defined according to the prescribed meta-model (Subsection 3.2). The repository is oriented towards a MAS meta-model-based classification of fragments; each one of them is in fact labeled with the MAS meta-model components that are defined or refined during its activities. Each activity has some inputs and produces some outputs in terms of defined/refined components of the MAS meta-model [25]. At the moment, the repository contains the fragments coming from PASSI, ADELFE and Tropos. Also, AOSE could benefit from projects such as OpenUP by defining specific AO method plugins and reusing predefined ones.

As far as tools are concerned, some of the CAME tools introduced in Subsection 2.4 – such as INGENME – are general enough to be effectively reused in the AOSE context. In addition, an example of CAPE tool specifically developed in the agent field is represented by Metameth [47], which allows the composition of a set of fragments stored in a specific repository. Metameth seems the only tool considering interactions with existing external tools for the creation of a CASE tool based on the characterization of the interaction between Metameth and the external tools. At the moment, these kinds of interactions are rather limited because of the complexity of the composition of the existing tools' portions—which are typically based on different development principles and specific APIs, to be taken into account in order to compose at the best the different tools.

3.4 AOSE Meta-model

In their most general acceptation, meta-models have been addressed from different points of view. Just to cite some of them we can list what follows:

[10] `http://www.pa.icar.cnr.it/passi/FragmentRepository/fragmentsIndex.html`

Bernon et al. — The process of designing a system (object or agent-oriented) consists of instantiating the system meta-model that the designers have in their mind in order to fulfill the specific problem requirements. In the agent world this means that the meta-model is the critical element because of the variety of methodology meta-models [48].

Gonzalez-Perez et al. — A meta-model is a model of a methodology or, indeed, of a family of related methodologies [49].

Brian Henderson-Sellers — A meta-model describes the rules and constraints of meta-types and meta-relationships. Concrete meta-types are instantiated for use in regular modeling work. A meta-model is at a higher level of abstraction than a model. It is often called *a model of a model*. It provides the rules/ grammar for the modeling language itself. The modlling language consists of instances of concepts in the meta-model [50].

Although it is possible to describe a methodology without an explicit meta-model, formalizing the underpinning ideas of the methodology in question is valuable when checking its consistency or when planning extensions or modifications. The importance of meta-model becomes clear when it is necessary to study the completeness and the expressiveness of a methodology, and when comparing different methodologies. Consequently, there is the need to study the different AOSE methodologies, to compare their abstractions, rules, relationships, and the process they follow, all of that would lead to a more comprehensive view of this variety of methodologies. Different works have been devoted to the study [51,52,53,54,55,56,57], and the unification of MAS meta-models [48,50,58,59], that it has been one of the more important topic in the work done by the agent community within the Agentlink Agent-Oriented Software Engineering Technical Forum Group (AOSE TFG) meetings[11].

The importance of meta-modeling is not only for having a precise view of agent-oriented methodologies as a way to check their completeness and expressivity, or to compare them, but it is also useful to clarify the distance between agent-oriented methodologies and infrastructures [51]. Expressing agent-oriented methodologies and infrastructures through formal meta-models is an initial step to reduce the conceptual and technical gap amongst these two research areas[12].

The situation up-to-date is that the lack of a unique MAS meta-model leads each methodology to deal with its own concepts and system structure, even if currently there are two kind of standardization efforts. The first effort is the recent standard "Software Engineering Metamodel for Development Methodologies" ISO/IEC 24744 [13] [60]. This standard is not specific for the agent-oriented field, rather it is very general. It is based on two powerful but someway complex concepts: powertype, which was already introduced, and clabject, i.e. a class/object hybrid concept. These tow concepts could easily lead to reduce the understandability of the standard.

[11] See `http://www.pa.icar.cnr.it/cossentino/al3tf3/` for more details.

[12] `http://www.mensa-project.org/`

[13] See `http://www.iso.org/iso/catalogue_detail.htm?csnumber=38854`

The second effort is represented by the OMG Agent Platform Special Interest Group[14] (Agent PSIG) that tries to identify and recommend new OMG specifications in the area of agent technology, with particular attention to:

- recommend agent-related extensions to existing and emerging OMG specifications;
- promote standard agent modeling languages and techniques that increase rigor and consistency of specifications;
- leverage and interoperate with other OMG specifications in the agent area.

The work of the Agent PSIG is still in its infancy and at the time of writing there are no specific results nor public documents.

3.5 Agent-Oriented Design Processes

In this section, we will provide a quick survey on some processes from literature. In order to provide a schematic view we will compare them according to a list of features. The list of compared AOSE processes includes: ADELFE [61], ASPECS [30], GAIA [62], INGENIAS [63], MaSE [64], PASSI [65], Prometheus [66], SODA [67,68].

A lot of approaches have been published about processes comparison (for instance see: [69,70,71,72,73,74]) and we decided to adopt a plain one based on a the consideration of a minimal set of process features. For each feature, we examine if the process exhibits it or not; the list of features is described below:

Coverage of the Entire Lifecycle — The methodology designer should be interested in developing a detailed and complete methodology from existing analysis till software deployment and maintenance. It is now widely accepted that the core workflows of a methodology are requirements collection, analysis, design, development (also called implementation), deployment and testing. Some methodologies cover the entire process of software engineering while some others are more focused on a part of that.

Problem Type — The challenge of agent or multi-agent design processes is to help the designers in building complex systems such as multi-agent systems usually are. Two categories of works could be found: those which are general high-level methodologies, and those which a focus on a specific application context.

Underlying Agent Architecture — Different design processes often refer to different agent architecture. Some processes look at BDI agents, some others refer to the IEEE FIPA specifications and finally there are processes that are not necessarily related to a specific architecture.

Origin — Some agent-oriented design processes are based on concepts and theories developed in the object-oriented field. Sometimes they even explicitly refer to an existing OO process (such as the Unified Process, UP).

[14] `http://agent.omg.org/`

Notation — Several design processes propose a proprietary notation to be adopted in the work products they specify. In some cases notation is an application or an extension of well known modeling languages (mainly UML), in other situations notation is a specific one, in one case there even is the proposal of a shared adoption of the same notation (as it happens in [75]).

Lifecycle — While modern object-oriented processes like UP are nowadays flattered to the adoption of the incremental/iterative lifecycle, agent-oriented approaches are still variegate and exhibit different solutions. Actually old paradigms like waterfall are still adopted as well as the last iterative/ incremental and even agile ones (not adopted by any of the here discussed processes but by some others).

Support for Model Transformations — The acceptance of a model-driven engineering paradigm [76] is growing in popularity in the agent-oriented landscape. According to that, (portions of) design models are obtained by transformation of preceding ones thus reducing design effort and at the same time increasing design quality. Several agent-oriented design process have been conceived to support that at a different extent.

Design Support Tools — The availability of a specific support tool allows for an easier enactment of the design process and usually increases the resulting design quality thanks to a set of check on notation and semantic aspects of the models.

Results of the comparison conducted on the basis of these criteria are reported in Table 1. Other design processes are reported in literature although they have not been compared with the previous ones. Among the others we may list: Agent-PriME [77], ASEME [78], AOR [79], Gormas [80], MESSAGE [81], O-MaSE [57], Tropos [82].

3.6 Roadmap

Multi-agent systems are increasingly used in different kinds of applications. Designing such systems requires handling different points of view about the system to be done. Sometimes, the correct way to handle these perspective have just to be discovered. For instance, ant-based simulation requires to take into account different environments: the simulation environment composed in the simulation participants; the client who provides the data and has to analyze and observe the running system; and the MAS environment which is composed of resources accessible by agents. Instead of building new development methods to deal with these three environments, it seems more logical to investigate how one existing method can be modified to take them into account, such as: PASSI which is slightly modified to take into account simulation requirements[5].

Fragments represent a paradigm which enables to diminish the cost of method definition by reusing existing ones. Although a couple of repositories already exist, much work is still to be done in the field. Little experience exists on widespread design processes composition and there is an obvious relationship between the quality of fragment repository and the assembled process. Moreover, the opposite is true as well. The more processes will be composed by using

Table 1. A comparison of some agent-oriented design processes. $^{(a)}$ Unified Notation proposed in [75].

Criterion	ADELFE	ASPECS	GAIA	INGENIAS	MASE	PASSI	PROMETHEUS	SODA
Entire Life-cycle	Yes	Yes	No	Yes	No	Yes	No	No
Problem Type	Open systems, Dynamic environment	Hierarchical decomposable problems	Open systems	Not specified	Robotic Systems	Information Systems, Robotic Systems	General Purpose	Open systems
Agent Architecture	Cooperative Agents	Holonic agents	Not specified	BDI	Not specified	FIPA	BDI	Not specified
Origin	UP	PASSI, RIO	Not specified	UP	UP	UP	Not specified	Not specified
Notation	UML Extension	UML adaptation	Non specific	Specific notation	UML adaptation$^{(a)}$	UML adaptation$^{(a)}$	UML adaptation$^{(a)}$	Tabular
Life-cycle Model	Iterative	Iterative / Incremental	Waterfall	Iterative	Iterative	Iterative / Incremental	Iterative	Iterative
MDE Support	Yes	Yes	No	Yes	No	Yes	No	Yes
Tools	Yes	No	No	Yes	Yes	Yes	Yes	No

fragment reuse, the more experience will be available on the field thus enabling an improvement of fragment definitions and repository structures. The main future research axis has to focus on three main elements: a language for fragments description; the means to ensure the interoperability between fragments; and tools to facilitate the fragments composition in order to produce the relevant software processes. A language is needed that guarantees interoperability of fragments developed by different designers and for different purpose. A main obstacle towards this interoperability is the MAS meta-model. Fragments of processes are associated with fragments of MAS specifications. Hence, a fragment depends on a MAS meta-model. The lack of a unified MAS meta-model topic has been long discussed in the AOSE community (just think about the debates held during the Agentlink AOSE TFG events[15]) and nonetheless it is still an unsolved issue. Besides the underlying MAS meta-model, the fragments must work with each others and the problem here is quite closed to component architecture. In any case, the work on languages for fragments description requires necessarily the assistance of CAPE tools which validate the devised solutions. CAPE tools permit designers to build the most relevant software processes regarding the application and the designers team expertise. Several steps can be followed: from the hand-made static built process to the self-design process; and from process determined at the beginning of the software development to a dynamic process adapted to the development status.

Other open research issues will briefly discussed below:

- The definition of application specific processes. Even considering the amount of existing processes as a good starting point for several custom products, there still is the need for specific approaches related to specific domains.
- The integration of agents with services and the related (web service) technology. Web services are nowadays widely spread and represent a good and affordable solution for the development of highly distributed systems. Where can agents contribute in a development scenario dominated by services? Probably the answer lays in the essence of agency: agents are autonomous, proactive, and social. These properties are not necessarily shared by services and they provide an invaluable support in the creation of a new abstraction and design layer.
- Agent reasoning techniques. They are not a new issue but they are always an important topic. Too many times agents are realized as simple statecharts. Introducing advanced reasoning capabilities in agents is a complex task but this is at the same time one of the crucial factors for distinguishing agent-oriented systems from traditional ones.
- Common pitfalls in design processes. Testing, deployment and formalized design techniques (patterns) issues have not received, in the agent-oriented field, the same attention they received in classic software engineering despite they are worth to. Testing techniques are often taken from object-oriented systems and adapted with minor changes. What about testing the successful implementation of a certain degree of autonomy or self-organization? No

[15] `http://www.pa.icar.cnr.it/cossentino/al3tf3/`

definitive answer still exist. Deployment is totally neglected by most of agent-oriented approaches but this should be one of the strength points of MAS (because of their easy distribution). In the era of cloud-computing, the agent community has a great opportunity. Agents may take profit of the elaboration infrastructure proposed by this paradigm and conversely may offer to cloud-computing new ideas for load balancing and distribution. Finally very few agent designers accept the use of design patterns as a daily practice. Several papers have been written on the matter and some pattern repositories exist in literature but the diffusion of them in process employment (but also conception) is still limited.
- Agent modeling languages. This is another long lasting issue. It is strictly related to the research about MAS meta-model but the perspective may be different. Defining a MAS meta-model means defining what are the elements that will be instantiated in the design of a new MAS. Defining a MAS modeling language means defining how the instances will be represented at design time. The link between the two is tight but there is not (necessarily) a one to one link.

4 Conclusion

SPE can bring a number of benefits to AOSE. One is the capability of critically analyzing our own development process by decomposing them into fragments. Another one is enabling the construction of new or altered development process as our knowledge of the needs of concrete domain problems and the performance of applied development processes grow.

While the research on SPE is lively in the general area of software engineering, the complexity of the processes to be modeled and built make the agent-oriented framework possibly the most suitable place for new approaches and solutions. In the AOSE field, current research in SPE aims at providing more flexible software processes taking into account the work already done by AOSE methodology designers, by re-using the most relevant parts (fragments) of any methodology.

At mid-term, research on SPE will likely be concerned with the development of tools supporting SPE, whereas the long-term perspective looks towards a self-designed software processes. Along these lines several difficulties should be overcome, among which the interoperability of fragments and the situational context representation and use seem to be the main ones.

References

1. Cernuzzi, L., Cossentino, M., Zambonelli, F.: Process models for agent-based development. Journal of Engineering Applications of Artificial Intelligence 18(2), 205–222 (2005)
2. Fuggetta, A.: Software process: a roadmap. In: ICSE 2000: Proceedings of the Conference on The Future of Software Engineering, pp. 25–34. ACM Press, New York (2000)

3. Picard, G., Bernon, C., Gleizes, M.P.: Etto: Emergent timetabling organization. In: [83]
4. George, J.P., Peyruqueou, S., Regis, C., Glize, P.: Experiencing self-adaptive mas for real-time decision support systems. In: Int. Conf. on Practical Applications of Agents and Multiagent Systems (PAAMS 2009). Springer, Heidelberg (2009)
5. Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., Russo, W.: Passim: A simulation-based process for the development of multi-agent systems. International Journal on Agent Oriented Software Engineering, IJAOSE (2008)
6. Morandini, M., Migeon, F., Penserini, L., Maurel, C., Perini, A., Gleizes, M.P.: A goal-oriented approach for modelling self-organising MAS. In: Aldewereld, H., Dignum, V., Picard, G. (eds.) ESAW 2009. LNCS, vol. 5881, pp. 33–48. Springer, Heidelberg (2009)
7. Jacobson, I., Booch, G., Rumbaugh, J.: The unified software development process. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
8. Cockburn, A.: Selecting a project's methodology. IEEE Software 17(4), 64–71 (2000)
9. Kumar, K., Welke, R.: Methodology engineering: a proposal for situation-specific methodology construction. In: Challenges and Strategies for Research in Systems Development, pp. 257–269 (1992)
10. ter Hofstede, H.A.M., Verhoef, T.F.: On the feasibility of situational method engineering. Information Systems 22(6/7), 401–422 (1997)
11. Brinkkemper, S.: Method engineering: engineering the information systems development methods and tools. Information and Software Technology 37(11) (1996)
12. Ralyté, J., Rolland, C.: An assembly process model for method engineering. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 267–283. Springer, Heidelberg (2001)
13. Henderson-Sellers, B.: Process Metamodelling and Process Construction: Examples Using the OPEN Process Framework (OPF). Annals of Software Engineering 14(1), 341–362 (2002)
14. Hamsen, A.: Situational Method Engineering. Moret Ernst & Young (1997)
15. Firesmith, D., Henderson-Sellers, B.: The OPEN Process Framework: An Introduction. Addison-Wesley, Reading (2002)
16. Gonzalez-Perez, C., McBride, T., Henderson-Sellers, B.: A metamodel for assessable software development methodologies. Software Quality Journal 13(2), 195–214 (2005)
17. Brinkkemper, S., Saeki, M., Harmsen, F.: Meta-modelling based assembly techniques for situational method engineering. Information Systems 24 (1999)
18. Ralyté, J.: Towards situational methods for information systems development: engineering reusable method chunks. In: 13th International Conference on Information Systems Development. Advances in Theory, Practice and Education, pp. 271–282 (2004)
19. Henderson-Sellers, B., Serour, M., McBride, T., Gonzalez-Perez, C., Dagher, L.: Process construction and customization. Journal of Universal Computer Science 10(3) (2004)
20. OMG: Software process engineering metamodel. Version 2.0. Object Management Group (2007)
21. Si-Said, S., Roland, C., Grosz, G.: Mentor: A computer aided requirements engineering environment. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) CAiSE 1996. LNCS, vol. 1080, pp. 22–43. Springer, Heidelberg (1996)
22. Harmsen, A., Ernst, M., Twente, U.: Situational Method Engineering. Moret Ernst & Young Management Consultants (1997)

23. Brinkkemper, S., Saeki, M., Harmsen, F.: A method engineering language for the description of systems development methods. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 473–476. Springer, Heidelberg (2001)
24. Saeki, M., Iguchi, K., Wen-yin, K., Shinohara, M.: A meta-model for representing software specification & design methods. In: Proceedings of the IFIP WG8.1 Working Conference on Information System Development Process, pp. 149–166. North-Holland Publishing Co., Amsterdam (1993)
25. Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardisation to research. International Journal of Agent-Oriented Software Engineering (IJAOSE) 1(1), 91–121 (2007)
26. Seidita, V., Cossentino, M., Gaglio, S.: Using and extending the SPEM specifications to represent agent oriented methodologies. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 46–59. Springer, Heidelberg (2009)
27. Seidita, V., Cossentino, M., Galland, S., Gaud, N., Hilaire, V., Koukam, A., Gaglio, S.: The metamodel: A starting point for design processes construction. International Journal of Software Engineering and Knowledge Engineering (IJSEKE) 20(4), 575–608 (2010)
28. Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardisation to research. International Journal of Agent Oriented Software Engineering 1(1), 91–121 (2007)
29. Osterweil, L.: Software processes are software too. In: 9th International Conference on Software Engineering (ICSE 1987), pp. 2–13. IEEE CS Press, Los Alamitos (1987)
30. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: ASPECS: an agent-oriented software process for engineering complex systems. International Journal of Autonomous Agents and Multi-Agent Systems (IJAAMAS) 20(2), 260–304 (2010)
31. Cossentino, M., Gaglio, S., Seidita, V.: Adapting PASSI to support a goal oriented approach: a situational method engineering experiment. In: 5th European Workshop on Multi-Agent Systems, EUMAS 2007 (2007)
32. Gleizes, M.P., Camps, V., George, J.P., Capera, D.: Engineering systems which generate emergent functionalities. In: Weyns, D., Brueckner, S., Demazeau, Y.E. (eds.) EEMMAS 2007. LNCS (LNAI), vol. 5049, pp. 58–75. Springer, Heidelberg (2008)
33. Jorquera, T., Bonjean, N., Gleizes, M.P., Maurel, C., Migeon, F.: Combining methodologies fragments using self-organizing MAS. Technical Report IRIT/RR-2010-4-FR, IRIT (2010)
34. Jorquera, T., Maurel, C., Migeon, F., Gleizes, M.P., Bonjean, N., Bernon, C.: ADELFE fragmentation. Technical Report IRIT/RR-2009-26-FR, IRIT (2009)
35. Cossentino, M., Sabatucci, L., Seidita, V.: Method fragments from the PASSI process. Technical Report RT-ICAR-21-03, Istituto di Calcolo e Reti ad Alte Prestazioni - Consiglio Nazionale delle Ricerche (2006)
36. Debenham, J., Henderson-Sellers, B.: Designing agent-based process systems – extending the OPEN process framework. In: Intelligent Agent Software Engineering, pp. 160–190. Idea Group Publishing, USA (2003)
37. Henderson-Sellers, B., Debenham, J., Tran, Q.-N.N., Cossentino, M., Low, G.: Identification of reusable method fragments from the PASSI agent-oriented methodology. In: Kolp, M., Bresciani, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS 2005. LNCS (LNAI), vol. 3529, pp. 95–110. Springer, Heidelberg (2006)

38. Henderson-Sellers, B., Giorgini, P., Bresciani, P.: Evaluating the potential for integrating the OPEN and Tropos metamodels. In: Al-Ani, B., Arabnia, H.R., Mun, Y. (eds.) International Conference on Software Engineering Research and Practice, SERP 2003, USA, June 23-26, vol. 2, pp. 992–995. CSREA Press, Las Vegas (2003)
39. ISO/IEC: Software Engineering — Metamodel for Development Methodologies. Fdis 24744 edn. (2006)
40. Gonzalez-Perez, C., Henderson-Sellers, B.: Metamodelling for Software Engineering. Wiley, Chichester (2008)
41. Gonzalez-Perez, C., Henderson-Sellers, B.: On the ease of extending a powertype-based methodology metamodel. In: 2nd Workshop on Metamodelling, WoMM 2006 (2006)
42. Rumbaugh, J.E.: Notation notes: Principles for choosing notation. JOOP 9(2), 11–14 (1996)
43. Tran, Q.-N.N., Henderson-Sellers, B., Debenham, J.: Incorporating the elements of the MASE methodology into agent OPEN. In: 6th International Conference on Enterprise Information Systems, ICEIS 2004 (2004)
44. Henderson-Sellers, B., Debenham, J., Tran, Q.-N.N.: Adding Agent-Oriented Concepts Derived from Gaia to Agent OPEN. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 98–111. Springer, Heidelberg (2004)
45. Henderson-Sellers, B., Tran, Q.-N.N., Debenham, J.: Incorporating Elements from the Prometheus Agent-Oriented Methodology in the OPEN Process Framework. In: Bresciani, P., Giorgini, P., Henderson-Sellers, B., Low, G., Winikoff, M. (eds.) AOIS 2004. LNCS (LNAI), vol. 3508, pp. 140–156. Springer, Heidelberg (2005)
46. Seidita, V., Cossentino, M., Gaglio, S.: A repository of fragments for agent systems design. In: Workshop From Objects To Agents, WOA 2006 (2006)
47. Cossentino, M., Sabatucci, L., Seidita, V.: A collaborative tool for designing and enacting design processes. In: Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M. (eds.) 24th Annual ACM Symposium on Applied Computing (SAC 2009), Honolulu, Hawai'i, USA, vol. 2, pp. 715–721. ACM, New York (2009)
48. Bernon, C., Cossentino, M., Gleizes, M.P., Turci, P., Zambonelli, F.: A study of some multi-agent meta-models. In: Odell, J.J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 62–77. Springer, Heidelberg (2005)
49. Gonzalez-Perez, C., McBride, T., Henderson-Sellers, B.: A metamodel for assessable software development methodologies. Software Quality Journal 13(2), 195–214 (2005)
50. Henderson-Sellers, B., Gonzalez-Perez, C.: A comparison of four process metamodels and the creation of a new generic standard. Information & Software Technology 47(1), 49–65 (2005)
51. Molesini, A., Denti, E., Omicini, A.: From AO methodologies to MAS infrastructures: The SODA case study. In: Artikis, A., O'Hare, G.M.P., Stathis, K., Vouros, G.A. (eds.) ESAW 2007. LNCS (LNAI), vol. 4995, pp. 300–317. Springer, Heidelberg (2008)
52. Molesini, A., Denti, E., Omicini, A.: MAS meta-models on test: UML vs. OPM in the SODA case study. In: [83], pp. 163–172
53. Grupo de Investigación en Agentes Software: Ingeniería y Aplicaciones (2009), Home page, `http://grasia.fdi.ucm.es/ingenias/metamodel/`
54. Beydoun, G., Low, G.C., Henderson-Sellers, B., Mouratidis, H., Gómez-Sanz, J.J., Pavón, J., Gonzalez-Perez, C.: FAML: A generic metamodel for MAS development. IEEE Transactions on Software Engineering 35(6), 841–863 (2009)

55. Dam, K.H., Winikoff, M., Padgham, L.: An agent-oriented approach to change propagation in software evolution. In: Australian Software Engineering Conference, pp. 309–318. IEEE Computer Society, Los Alamitos (2006)
56. Giorgini, P., Mylopoulos, J., Perini, A., Susi, A.: The Tropos metamodel and its use. Informatica 29, 401–408 (2005)
57. Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H., Valenzuela, J.L.: O-maSE: A customizable approach to developing multiagent development processes. In: Luck, M., Padgham, L. (eds.) Agent-Oriented Software Engineering VIII. LNCS, vol. 4951, pp. 1–15. Springer, Heidelberg (2008)
58. Cossentino, M., Gaglio, S., Sabatucci, L., Seidita, V.: The PASSI and Agile PASSI MAS meta-models compared with a unifying proposal. In: [83], pp. 183–192
59. Beydoun, G., Gonzalez-Perez, C., Low, G., Henderson-Sellers, B.: Synthesis of a generic MAS metamodel. In: 4th International Workshop on Software Engineering for Large-scale Multi-Agent Systems (SELMAS 2005), pp. 1–5. ACM, New York (2005)
60. Henderson-Sellers, B., Gonzalez-Perez, C.: Standardizing methodology metamodelling and notation: An iso exemplar. In: Kaschek, R., Kop, C., Steinberger, C., Fliedl, G. (eds.) UNISCON. Lecture Notes in Business Information Processing, vol. 5, pp. 1–12. Springer, Heidelberg (2008)
61. Bernon, C., Camps, V., Gleizes, M.P., Picard, G.: Engineering adaptive multi-agent systems: the ADELFE methodology. In: Agent Oriented Methodologies, pp. 172–202. Idea Group Publishing, USA (2005)
62. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM) 12(3), 317–370 (2003)
63. Pavòn, J., Gòmez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. In: [84], ch. IX, pp. 236–276
64. DeLoach, S.A., Kumar, M.: Multi-agent systems engineering: An overview and case study. In: [84], ch. XI, pp. 317–340
65. Cossentino, M.: From requirements to code with the PASSI methodology. In: [84], ch. IV, pp. 79–106
66. Padgham, L., Winikoff, M.: Prometheus: A methodology for developing intelligent agents. In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585, pp. 174–185. Springer, Heidelberg (2003)
67. Omicini, A.: SODA: Societies and infrastructures in the analysis and design of agent-based systems. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 185–193. Springer, Heidelberg (2001)
68. Molesini, A., Zhang, S.-W., Denti, E., Ricci, A.: SODA: A roadmap to artefacts. In: Dikenelli, O., Gleizes, M.-P., Ricci, A. (eds.) ESAW 2005. LNCS (LNAI), vol. 3963, pp. 49–62. Springer, Heidelberg (2006)
69. Cernuzzi, L., Rossi, G., Plata, L.: On the evaluation of agent oriented modeling methods. In: Workshop on Agent Oriented Methodology, pp. 21–30 (2002)
70. Bernon, C., Gleizes, M.P., Picard, G., Glize, P.: The ADELFE methodology for an intranet system design. In: Giorgini, P., Lespérance, Y., Wagner, G., Yu, E. (eds.) 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2002), Toronto, Canada. CAiSE 2002, CEUR Workshop Proceedings, vol. 57 (2002)
71. Sturm, A., Shehory, O.: A framework for evaluating agent-oriented methodologies. In: Giorgini, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS 2003. LNCS (LNAI), vol. 3030, pp. 94–109. Springer, Heidelberg (2004)

72. Dam, K.H.: Comparing agent-oriented methodologies. In: Giorgini, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS 2003. LNCS (LNAI), vol. 3030, pp. 78–93. Springer, Heidelberg (2004)
73. Luck, M., Ashri, R., D'Inverno, M.: Agent-Based Software Development. Artech House, Norwood (2004)
74. Tran, Q.N.N., Low, G.C.: Comparison of ten agent-oriented methodologies. In: Agent Oriented Methodologies, pp. 341–367. Idea Group Publishing, USA (2005)
75. Padgham, L., Winikoff, M., DeLoach, S., Cossentino, M.: A unified graphical notation for AOSE. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 116–130. Springer, Heidelberg (2009)
76. Schmidt, D.C.: Model-driven engineering. Computer 39(2), 25–31 (2006)
77. Miles, S., Groth, P.T., Munroe, S., Luck, M., Moreau, L.: AgentPrIMe: Adapting MAS designs to build confidence. In: Luck, M., Padgham, L. (eds.) Agent-Oriented Software Engineering VIII. LNCS, vol. 4951, Springer, Heidelberg (2008)
78. Spanoudakis, N., Moraitis, P.: Development with ASEME. In: 11th International Workshop on 11th International Workshop on Agent Oriented Software Engineering, AOSE (2010)
79. Wagner, G.: Agent-oriented analysis and design of organizational information system. In: 4th IEEE International Baltic Workshop on Databases and Information Systems (2000)
80. Argente, E., Botti, V., Vincente, J.: GORMAS: An organizational-oriented methodological guideline for open MAS. In: Proc. of Agent-Oriented Software Engineering (AOSE) Workshop (2009)
81. Garijo, F.J., Gòmez-Sanz, J.J., Massonet, P.: The MESSAGE methodoly for agent-oriented analysis and design. In: [84], ch. VIII, pp. 203–235
82. Castro, J., Kolp, M., Mylopoulos, J.: A requirements-driven development methodology. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 108–123. Springer, Heidelberg (2001)
83. Pěchouček, M., Petta, P., Varga, L.Z. (eds.): CEEMAS 2005. LNCS (LNAI), vol. 3690. Springer, Heidelberg (2005)
84. Henderson-Sellers, B., Giorgini, P.: Agent Oriented Methodologies. Idea Group Publishing, Hershey (2005)