Auton Agent Multi-Agent Syst (2014) 28:430–473
DOI 10.1007/s10458-013-9228-y

Medee Method Framework: a situational approach for organization-centered MAS

Sara J. Casare · Anarosa A. F. Brandão · Zahia Guessoum · Jaime S. Sichman

Published online: 15 June 2013

Abstract This paper presents a situational approach, called Medee Method Framework, which allows the development of organization-centered MAS in a disciplined way, even though some agent organizational (AO) models are not currently incorporated into agent-oriented software engineering (AOSE) methods. In order to do that, such a method framework proposes the composition of MAS situational methods out of method fragments according to a given project situation, by applying the principles proposed by situational method engineering. The proposed approach provides a high degree of reuse and flexibility, allowing the composition of new methods as well as the reengineering of AOSE methods based on the standards proposed by SPEM. Furthermore, it allows the user to leverage advantages of both AOSE methods and AO models in order to develop organization-centered MAS. The Medee Method Framework offers a method repository that covers different development phases, such as requirements, analysis, design, implementation, as well as the main components of a MAS application, like agents, environments, interactions, and organizations. This repository has been sourced from several AOSE methods and AO models, as Gaia, Tropos, Ingenias, PASSI, MOISE, and OperA.

Keywords Organization-centered MAS · AOSE method · Method fragment · Situational method

S. J. Casare · A. A. F. Brandão · J. S. Sichman (✉)
Laboratório de Técnicas Inteligentes (LTI), Escola Politécnica (EP), Universidade de São Paulo (USP), Av. Luciano Gualberto, 158 trav. 3, São Paulo, SP 05508-970, Brazil
e-mail: jaime.sichman@poli.usp.br

S. J. Casare
e-mail: sjcasare@uol.com.br

A. A. F. Brandão
e-mail: anarosa.brandao@poli.usp.br

Z. Guessoum
Equipe SMA, Laboratoire d'Informatique de Paris 6 (LIP6), Université Pierre et Marie Curie (UPMC), 4 place Jussieu, 75005 Paris, France
e-mail: zahia.guessoum@lip6.fr

1 Introduction

Multiagent systems (MASs) provide a new paradigm for conceptualizing, designing, and implementing software systems, ranging from manufacturing to process control, air traffic control, and information management. They are particularly attractive for creating software that operates in distributed and open environments, such as the Internet, and which simulates scenarios that serve as basis to create public policies and strategies to deal with complex problems, such as rescue after natural disasters and evacuation of public facilities.

Nevertheless, in order to be adopted by the software industry, a controlled and disciplined way to conduct software development projects related to the aforementioned domains is needed. Despite the research community efforts while proposing methods for structuring and guiding the development of MAS [3,6,15,19,44,55,56], agent-oriented software engineering (AOSE) methods are still at an early stage, mainly being applied in the context of academic projects. Moreover, the development of complex systems using MAS requires specific methods and then situational method engineering [7,30] for MAS seems to be a good solution for it.

Most AOSE methods adopt an agent-centered MAS approach, focusing on agent behavior, such as Tropos [6], MaSE [55], Adelfe [3], and PASSI [15]. Nonetheless some of them—such as Gaia [56], Ingenias [44], and O-MaSE [19]—propose developing MAS based on the notion of agent organizations.

Moreover, several agent organizational (AO) models have been proposed in the MAS literature for developing organization-centered MAS (OC-MAS) [42] beyond AOSE methods, such as MOISE+ [33] and OperA [22]. Some of these AO models encompass aspects that are not currently covered by AOSE methods. For instance, they allow the specification of organizational characteristics during the development of the MAS application and possibly changing at runtime. Such changes are accomplished by organizational acts, e.g., agent actions that can modify the organization, such as adding roles or changing the organizational structure [45].

Thus, let us consider a class of problems whose solutions depend on organization and coordination and are suitable for adopting the MAS paradigm, and could benefit from using an organization-centered approach. Currently, a project team that looks for a disciplined way to develop a MAS application involving such organizational characteristics will not find a method ready for use. An example of such a real application could be an information system to support the adoption of strategies for evacuating huge facilities under bomb threats.

In addition, using AOSE methods or AO models separately may cause some project drawbacks. On one hand, AOSE methods offer a development cycle but may not support the required organizational aspects. On the other hand, most AO models do not provide a structured MAS development cycle in terms of phases, tasks, and work products, as extensively accepted by the software industry. In order to overcome this issue, the possibility of reusing parts of existing AOSE methods and AO models to build methods that suit specific project characteristics is very interesting. In fact, the AOSE and Method Engineering research communities are also addressing issues related to building methods on demand [1,7,16,23,29,30,32,40,46–51].

In this direction, this paper introduces the Medee Method Framework, a framework for building methods on demand that supports the development of organization-centered MAS in a controlled and disciplined way. Such a framework allows the user to build methods that may cover the main MAS development phases and components, by combining the advantages of both AOSE methods and AO models. Nevertheless, it also suits building methods to support the development of agent-centered MAS.

The Medee Method Framework is part of a broad approach—the Medee Framework [8]—that aims to offer situational method composition embedded in a MAS method improvement cycle. Nonetheless, this paper is focused on the method framework itself.

It is important to note that Medee users are method engineers since our ultimate goal is that it can be adopted by the software industry. Nowadays, these engineers are playing strategic roles in the software industry. For instance, IBM adopts the role of "Method Champion", who is responsible for building and/or choosing appropriate methods to support software projects considering their characteristics. Having this in mind, in order to define and develop the framework we adopted the principles proposed by situational method engineering [7,30,40], combined with software and system process engineering meta-model (SPEM) [43]—the de facto standard for modeling software processes—and the open source tool built upon it, the Eclipse Process Framework Composer (EPF Composer) [31].

This paper is organized as follows. Section 2 briefly presents background and related work. Sections 3 and 4 describe theoretical foundations to the method framework definition, while Sects. 5 and 6 describe the application of such foundations to implement and use the method framework. A comparison between the proposed framework and other approaches is presented in Sect. 7, outlining the advancements achieved with the Medee Method Framework. Finally, conclusions and future work are presented in Sect. 8.

2 Background and related work

In this section, we present some basic notions that are essential for understanding the main aspects of the Medee Method Framework, related to Situational Method Engineering, SPEM, EPF Composer, and Organizational-centered MAS. Moreover, we present several approaches concerning Situational Method Engineering notions proposed in the AOSE field.

2.1 Situational Method Engineering

The word *'method'* comes from the Greek—*methodos*—that means way of investigation. Roughly speaking, a software development method encompasses a set of integrated procedures, techniques, and product descriptions, which provide a consistent support for the software development [7].

Kruchten [41] suggests that a software development method should not be used before being customized according to current project characteristics. Otherwise, the project risks wasting work already done and producing artifacts of little added value. On one hand, the method might be made as lean as possible and, on the other hand, it must fulfill the objective of producing high quality software. A suitable method for a small project, for example a 3 months project, may not fit a larger project, such as a 3-year project, given that during a longer time period, the project environment (e.g. the problem to be solved and people involved) will probably change.

Situational Method Engineering is the discipline concerned with a controlled construction of software development methods according to a given *project situation*. In this context, a situation may encompass specific aspects of the problem to be solved, expected project deliverables, project team, and many other factors. Therefore, a *situational method* consists of a method built according to a project situation by combining reusable parts of methods, usually called *method fragments* [1,7,30].

The Situational Method Engineering research community has proposed several approaches to deal with the notions related to method fragment and situational method building [1,7, 23,30,40,46]. Among these approaches there are the Method Fragment approach [7,30], the Method Component approach [40], the Process Component approach [23], and the Method Chuck approach [46].

The Medee Method Framework is strongly based on the first two approaches: the Method Fragment and Method Component ones. The former proposes the specification of method fragments into several layers of granularity and a bottom–up reuse mechanism for building situational methods, by assembling the selected fragments. The Method Component approach proposes the specification of exchangeable and reusable parts of the method that can be viewed from two perspectives: an internal and an external view. Moreover, this approach proposes a procedure to build situational methods—called configuration procedure—which involves the notion of a *base method*: a method chosen as a starting point for such a procedure. The configuration procedure offers a top–down reuse mechanism for building situational methods that consists of eliminating method components from the base methods, as well as adding and/or exchanging method components from it, using those captured from other methods.

A general iterative procedure for building situational methods usually encompasses five main steps: (i) management of the method repository, (ii) characterization of the project situation, (iii) selection of method fragments, (iv) situational method building, and finally (v) project execution [7,30,40,46].

The management of the method repository consists of elaborating method fragments based on existing methods and techniques, as well as updating those fragments to take into account lessons learned from previous projects. Therefore, the management of such a repository presupposes the definition of method fragment. After characterizing the project situation and then selecting method fragments appropriate to such a situation, the next step consists of building the situational method out of selected method fragments, by adopting a reuse mechanism for combining them (e.g. assembling, configuring). Finally, the last step concerns the project execution itself, based on the situational method. Moreover, this step involves gathering lessons learned from the adopted method, which are taken into account to update the method fragments already stored, closing the iterative procedure.

The Medee Method Framework covers steps (i)—(iv) and adopts the term *Situational Method Composition* for representing distinct approaches for building such methods, as such assembling and configuration. The step (v), concerned with project execution and lessons learned, is covered by the Medee Improvement Cycle, described in [8].

2.2 SPEM

Along with the notions proposed by the Situational Method Engineering discipline, several meta-models, frameworks, and tools supporting software development methods have emerged in the last decades. Among them we can cite the SPEM, the *Software Engineering Meta-model for Development Methodologies* (SEMDM) [38], and the *EPF Composer*.

SPEM consists of a meta-model that provides the concepts for modeling, documenting, managing, and enacting development methods and processes. It is the *de facto* standard for managing methods in the Software Engineering field, and it was adopted as method meta-model for the Medee Method Framework. Consequently, the EPF Composer has been used for developing this framework.

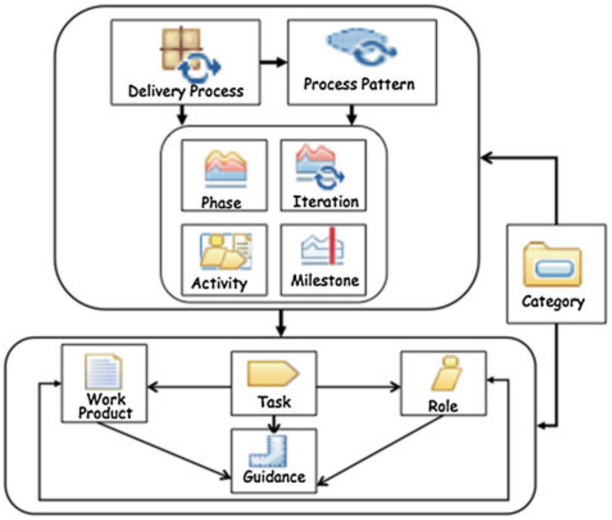


Fig. 1 SPEM main elements and their relationship

2.2.1 SPEM in a nutshell

The goal of SPEM is to allow the representation of a broad range of development methods of different styles and cultural backgrounds, distinct levels of formalism, and different lifecycle models (e.g. waterfall, iterative).

Figure 1 (lower frame) depicts SPEM's method elements— task[1] work product, role, and guidance—as well as the relationships between them (depicted through the arrows). A task represents an assignable unit of work described step-by-step, generally executed from a few hours up to a few days. A work product represents pieces of work that are used, modified, and produced by a task. Roles are used to define who performs the tasks. Guidance represents specific descriptions related to any SPEM concepts, such as work products, tasks, and roles. SPEM offers several types of guidance, among them guidelines, reusable assets, whitepapers, and concepts.

Furthermore, tasks are grouped to form several types of work breakdown structures, such as activities, iterations and phases (Fig. 1, upper frame). An activity represents the basic unit of work within a process, which can be nested into phases and iterations. A phase represents a significant period in a project, during which several deliverables are usually produced, while iteration represents a set of nested activities that are repeated more than once during a project. Category (Fig. 1, right) represents classification structures used to group elements based on the user's criteria.

Moreover, SPEM proposes two types of processes: process pattern and delivery process. The former represents a reusable cluster of activities that provides a consistent development approach to common problems, offering a building block for assembling processes, while the latter represents a complete end-to-end project lifecycle. It is important to observe that these two concepts are the backbones in specifying Medee method fragments and Medee situational methods respectively, as described in detail in Sect. 3.

[1] In order to improve readability, the Comic Sans font is used in this paper from hereon in to designate SPEM concepts.

SPEM also offers management and reusability capabilities through concepts such as method plugin, method library, method configuration, and variability. A method plugin provides a physical storage for modularization, extension, packaging, and deployment of tasks, work products, activities, phases, process pattern and so on. A method configuration provides a visibility space of method plugins, offering a logical view that allows the filtering of their elements, while a method library is a physical container for method plugins and method configuration. Finally, variability provides mechanisms for tailoring method elements without directly modifying their original content: it allows the definition of differences—like replacements and extensions—relative to the original method element, such as an original task or work product. As explained in Sect. 5, the Medee Method Framework applies variability in order to standardize method fragments.

2.2.2 EPF Composer overview

EPF Composer is the open source tool developed by the Eclipse Foundation, which implements SPEM. It consists of a content system that provides a common management structure for SPEM elements. Therefore, the EPF Composer offers features for defining, tailoring, and managing software development methods as well as their portions. Finally, the content managed in the EPF Composer can be exported (or imported) as a set of method plugins, as well as published on HTML pages and deployed to Web servers for distributed usage.

Our approach to define method fragment elements is fully based on SPEM, and using the EPF Composer, without the introduction of any further element or association. Therefore, the Medee Method Framework is totally SPEM and EPF Composer-compliant. Thanks to this compliance, the Medee Method Framework concrete implementation consists of a set of method plugins that can be imported in the EPF Composer,[2] and Medee situational methods can be published on HTML pages, as illustrated in Sect. 6.

2.3 Organization-centered MAS

Lemaıtre and Excelente [42] suggest that MAS research can be divided into two classes according to the approach adopted for representing social aspects: agent-centered and organization-centered MAS approaches.

The agent-centered MAS approach proposes representing the social aspects of MAS through concepts focused on the agents' behavior as a social entity: as social commitments [13], as joint intentions [14], and using social reasoning [52]. This approach is strongly focused on the agent notion. Moreover, it encompasses research on formalisms for representing individual agent knowledge. However, this approach does not define explicitly organizations.

Research concerning organization-centered MAS adopts a sociological and organizational vision for modeling these systems, encompassing the specification of distinct types of agent groups, such as organizations and teams. These groups establish rules and norms to constrain agent behavior, as well as to specify agent's rights and duties, independently of a particular agent model.

The basic conceptual entity in the organization-centered MAS approach is the agent organization as a whole. It is composed of a set of goals, norms, and functionalities, as well as an internal structure of components, like subsystems [42]. Moreover, MAS development

[2] This is made by creating a new method library using the following sequence of commands from the EPF Composer toolbar: File>Import>Library Configuration. See *EPF Composer Help* and *EPF Composer Tutorial* for more information about this feature.

approaches that deal with organizations can be classified according to the evolution of the organization during the MAS life cycle. They are divided into two categories: *agent-oriented engineering* and *organization-oriented MAS* [45].

The first category encompasses those approaches that deal with organizational specification during MAS application design time, by involving an explicit model for representing organizations. However, these approaches do not allow agents to modify core aspects of their organizations during runtime, such as creating or eliminating roles, modifying organization hierarchy or goals. Examples of such approaches are mainly found among the AOSE methods, such as Gaia, Ingenias, and O-MaSE. For instance, Gaia suggests analyzing and designing the MAS application based on organizational aspects, as roles, structure, and norms. However, Gaia does not specify how to add new roles or change the organizational structure dynamically, after the MAS implementation.

The organization-oriented MAS category classifies those MAS development approaches that allow both specifying organizational aspects during MAS application development and possibly changing them over the course of the MAS application execution. Such changes are made through agent actions—called *organizational acts*—that can modify the organization, for instance by changing the organizational structure and adding roles. Examples of these approaches, found among the AO models, are MOISE+ and OperA.

Since some fundamental characteristics of AO models are not currently incorporated into AOSE methods, like reorganization during application runtime, someone who adopts an organization-oriented MAS approach may not take advantage of AOSE methods and AO models together.

2.4 Situational Method Engineering for MAS

Several approaches concerning Situational Method Engineering for MAS have been proposed in the last decade [16,17,19,24,32,47–51]. Most of these approaches are related to at least one of the five steps of a general procedure for building situational methods—covering from the method repository management to the situational method building and project execution—as described in the following paragraphs.

As mentioned in Sect. 2.1, a clear definition of MAS method fragments is required whenever one defines an approach to manage such fragments in a method repository. In the AOSE field, some method fragment notions had already been proposed. Among them, we can cite the approach proposed by Seidita et al. [50], as well as the fragment description for adaptive methods proposed by Rougemaille et al. [47]. In general, such works propose the use of SPEM as a common meta-model for representing MAS method fragments.

The approach presented in [50] proposes the term *process fragment* for designating a part of method, instead of method fragment. In such an approach a MAS meta-model is in the core of the process fragment definition, which involves notions like the MAS meta-model elements and the relationships between them. Moreover, a process fragment encompasses elements as such activities, work products, roles, guidelines, and can be defined in three levels of granularity: phase, composed, and atomic. These granularity levels are based on the work products delivered by a process fragment: (i) a phase fragment aims to deliver a set of work products generated during a development phase; (ii) a composed fragment delivers a work product; and (iii) an atomic fragment delivers part of a work product. Therefore, although involving SPEM concepts (e.g. activities, roles, work products) a process fragment cannot be represented using only SPEM elements, since the MAS meta-model concepts are out of SPEM scope.

Based on such a process fragment definition, the approach presented in [51] proposes guidelines for extracting fragments from AOSE methods, while Process for the Design of Design Processes (PRoDe) [49] consists of a procedure for building situational methods, covering from the method repository management to project execution. Thus, these three approaches, along with that presented in [16], are strongly based on one broad meta-model that underpins a MAS application. Moreover, MAS meta-model elements are the starting point for fragments extraction in [50,51], and for fragments selection and situational method building in [49].

As discussed in Sect. 7, presupposing the existence of a common MAS meta-model for defining fragments sourced from several MAS development approaches might postpone (or even prevent) the use of Situational Method Engineering principles in AOSE, since there is no consensus in the MAS research field concerning multiagent key concepts, and notations to represent the agency notion [18].

Moreover, other approaches have been proposed in the AOSE field, such as the one presented in [48], which contains fragments sourced from some AOSE methods (e.g. PASSI, Tropos, Adelfe) and the OPEN Process Framework (OPF) [23,32]. The latter is a framework for building software development methods initially proposed for dealing with the object-oriented paradigm and later extended to manage agent-oriented development methods. Thus, some methods—among them Tropos, PASSI, MaSE, and Gaia—have been incorporated in this framework. Such methods were represented in terms of *process components*, which is the notion used by OPF to designate a part of method. However, OPF is not based on SPEM.

Furthermore, there are some work in the literature concerning the extension of specific AOSE methods in order to incorporate organizational notions, as ASPECS [17] and O-MaSE [19,24]. The latter extends MaSE in order to support organization-centered MAS development, while ASPECS consists of a method for the development of MAS based on a holonic organizational meta-model, which has been built following PRoDe and involving, among others, fragments captured from PASSI.

Finally, IEEE-FIPA has recently proposed a template [37] for designing and documenting full AOSE methods, which suggests describing methods using SPEM elements such as process, phase, activity, role, task, work product, and guideline. The use of this template for documenting AOSE methods may facilitate the extraction of fragments from these methods. However, this template does not deal with method fragment management nor addresses the standardization of AOSE methods elements by, for instance, stating a common set of MAS development roles (e.g. developer, architect, tester), although suggesting that this step should be done.

In the next section, we present the Medee Method Framework that allows you to take advantage of both AOSE methods and AO models characteristics for building MAS based on a well-established development method.

3 Medee Method Framework

3.1 Overview

As previously mentioned, the Medee Method Framework covers four steps of a general procedure for building situational methods, from managing the method repository to building methods according to a given project situation.

Figure 2 depicts how a method engineer could use the Medee Method Framework: she/he could start analyzing the existing AOSE methods as well as the AO models in order to

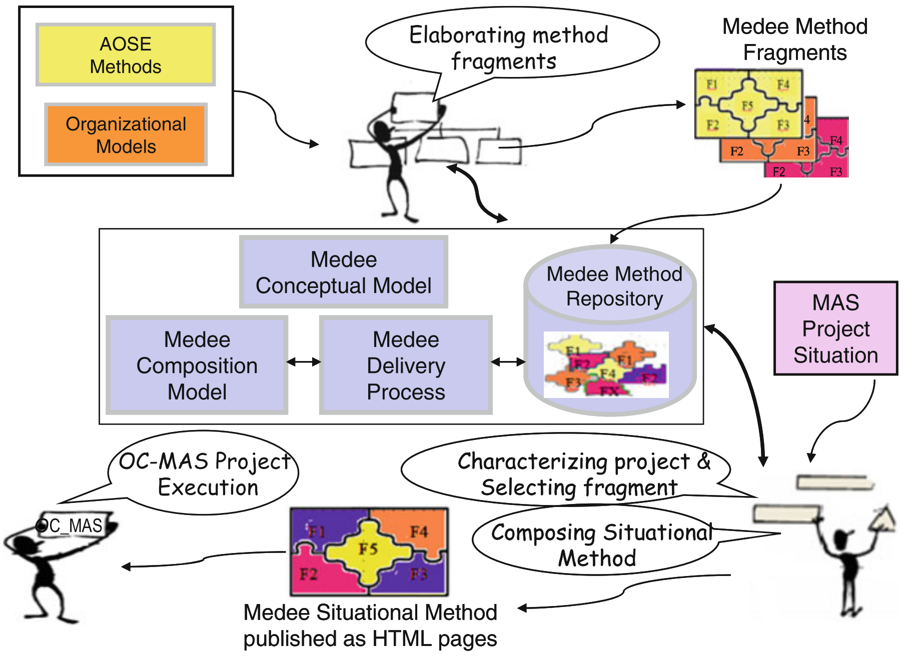


Fig. 2 Using the Medee Method Framework for composing organization-centered MAS methods on demand

describe them as a collection of Medee method fragments and store these fragments in the method repository, i.e. elaborating method fragments. Having this repository, the method engineer could then select the appropriate fragments based on a given MAS project situation characterization and compose the Medee situational method.

In order to offer such a situational method composition based on AOSE methods and AO models, the Medee Method Framework encompasses four components, as depicted in Fig. 2 (central frame). Firstly, it underpins the *Medee Conceptual Model* that defines MAS method fragments and MAS situational methods. Secondly, it provides the *Medee Method Repository* for managing method fragments and situational methods. Thirdly, it provides the *Medee Composition Model* that aims to support the identification of appropriate fragments during the situational method composition. Finally, the *Medee Delivery Process* specifies in a step-by-step manner the whole process from the Medee method fragments elaboration to the deployment of the Medee situational method to Web servers in order to be used during the project execution.

The four components of the Medee Method Framework are represented in terms of SPEM elements. Moreover, excepting the Medee Conceptual Model, these components were built upon the EPF Composer. For example, the Medee Delivery Process is itself described using SPEM, built upon the EPF Composer, and published on HTML pages currently available at the Medee website.[3]

Next section presents the Medee Conceptual Model, which is an improved version of [11,12]. The remainder Medee components are presented in Sects. 4 and 5.

[3] http://www.medee.poli.usp.br/.

3.2 Medee Conceptual Model

3.2.1 Definitions

In order to build MAS situational methods, we propose a definition for a MAS Method Fragment[4] and another one for a MAS Situational Method. These definitions are strongly based on the Method Fragment approach [7,30] and the Method Component approach [40], briefly described in Sect. 2.1.

Moreover, we used SPEM 2.0 as meta-model for describing MAS method fragments, and adopted some concepts proposed by Jacobson et al. [39], which is among the most popular software development processes. Thus, we propose the following definitions:

A Medee MAS Method Fragment is a standardized building block that represents a coherent part of a MAS development approach, in the sense that it is consistent and clearly stated

A Medee MAS Situational Method is a sequence of Medee MAS Method Fragments, possibly nested within phases and iterations, which describes how a MAS project shall be executed according to a specific project situation

The *coherence* and *standardization* of Medee MAS method fragments aims to improve the elaboration of method fragments as well as the composition of situational methods.

The following aspects assure the coherence of Medee MAS method fragments and situational methods: (i) method fragments and situational methods are described in terms of SPEM 2.0 elements (e.g. task, work product, role, process pattern, delivery process) and their associations; (ii) method fragments must pertain to one of the four granularity layers—activity, phase, iteration, and process; and (iii) method fragments are defined through internal and external views.

The standardization of Medee MAS method fragments, and consequently the standardization of situational methods composed out of them, is based on the following notions: (i) encapsulation of work products and milestones generated by a method fragment using a common framework, the Medee MAS Work Product Framework; (ii) utilization of a common set of development roles (e.g. system analyst, developer, tester) for defining who performs the work described in a method fragment, provided by the Medee Common Roles; and (iii) classification of method fragments based on a semiotic criterion, made available by the Medee MAS Semiotic Taxonomy. The Medee MAS Work Product Framework and Medee Common Roles are described in the course of this section, while the Medee MAS Semiotic Taxonomy is presented in Sect. 4.3, since it plays an important role during the composition of Medee situational methods.

3.2.2 Main concepts

The Medee Method Framework main concepts and corresponding relations are illustrated in Fig. 3 as a UML class diagram. Such a diagram shows the four layers in which a MAS method fragment can be defined—activity, phase, iteration, process—represented as its subclasses: MAS Activity Method Fragment, MAS Iteration Method Fragment, MAS Phase Method Fragment, and MAS Process Method Fragment (Fig. 3, lower).

[4] In order to improve readability, hereon in this paper uses the Arial Narrow font to show Medee concepts, and continues to show SPEM concepts using Comic Sans font.

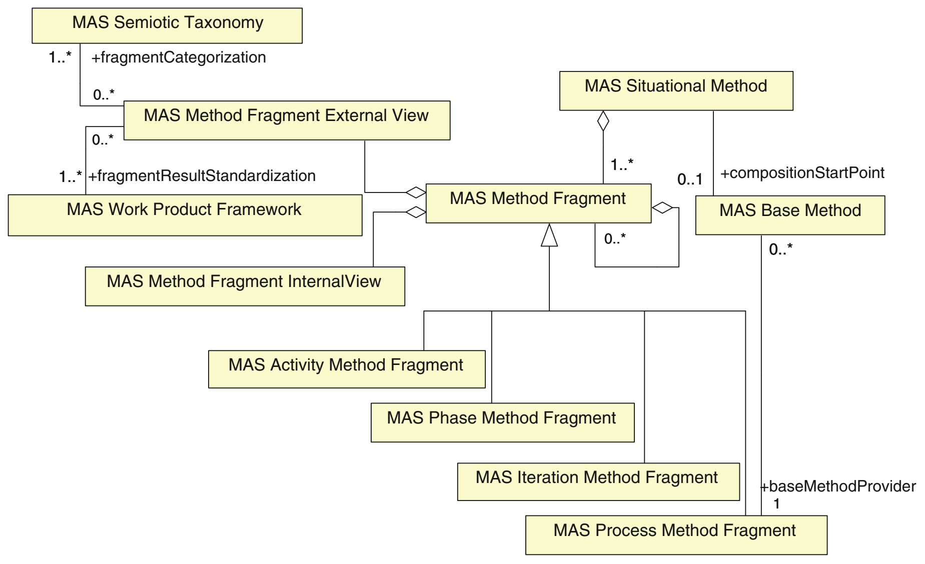

MAS Semiotic Taxonomy
1..*
+fragmentCategorization
0..*
MAS Method Fragment External View
0..*
1..*
+fragmentResultStandardization
MAS Work Product Framework
MAS Situational Method
1..*
0..1
+compositionStartPoint
MAS Method Fragment
MAS Base Method
0..*
0..*
MAS Method Fragment InternalView
MAS Activity Method Fragment
MAS Phase Method Fragment
MAS Iteration Method Fragment
+baseMethodProvider
1
MAS Process Method Fragment

Fig. 3 Main concepts of the Medee Method Framework

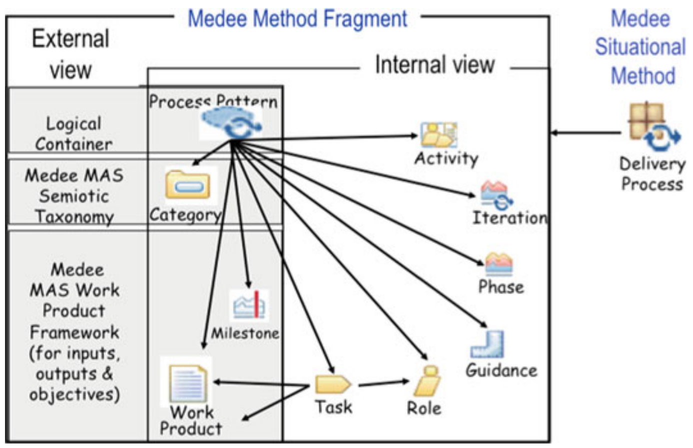


Fig. 4 Main elements of Medee MAS Situational Method and Medee MAS Method Fragments

Several method fragments may be combined to form a larger one, as represented by the MAS method fragment auto-association in Fig. 3 (center). For instance, a MAS Phase Method Fragment may contain several MAS Activity Method Fragments.

Furthermore, Fig. 3 (upper right) shows how a MAS Situational Method may be composed: it might involve several MAS Method Fragments as well as a MAS Base Method, which in turn is provided by a MAS Process Method Fragment. Following the *base method* notion proposed by Karlsson [40], a MAS Base Method offers a starting point for the MAS situational method composition in a top–down fashion.

Finally, Fig. 3 (upper left) shows that a MAS Method Fragment encompasses external and internal views. The internal view involves only SPEM concepts (Fig. 4), while the external view involves two Medee concepts, the MAS Work Product Framework and the MAS Semiotic Taxonomy.

As previously mentioned, the use of SPEM 2.0 concepts (and their relationships) to describe MAS Method Fragments and MAS Situational Methods consists of the first step towards a coherent definition. Thus, as depicted in Fig. 4 from a diagrammatical perspective, a Medee MAS Situational Method encompasses a delivery process, which describes a complete end-to-end project lifecycle, while a Medee MAS Method Fragment encompasses a process pattern, which offers a logical container for the SPEM elements pertaining to the fragment. By means of such a process pattern and depending on the layer of granularity, a method fragment contains several SPEM elements, such as activities, tasks, steps, work products, roles, categories, phases, iterations, milestones, and guidance.

The remainder of this section presents the MAS method fragment granularity layers in detail, followed by the Medee MAS Work Product Framework and the Medee Common Roles. We also define the main differences between the external and internal views of MAS method fragments next.

3.2.3 Fragment granularity layers

The four granularity layers for Medee Method Fragments— activity, iteration, phase, and process—contribute to reach a coherent definition. In order to define these four layers, we adopted the notions of Jacobson et al. [39] and SPEM homonym concepts.

A Medee MAS method fragment in the activity layer, called Medee MAS Activity method fragment, consists of the smallest fragment that composes a MAS situational method. Such a method fragment is built on the SPEM homonym element—activity—and allows you to represent a small portion of work that is performed by roles in order to generate one or more work products.

Furthermore, a Medee MAS method fragment in the phase layer (Medee MAS Phase method fragment) encompasses work that is performed during a significant project period. Also, a Medee MAS method fragment in the iteration layer (Medee MAS Iteration method fragment) offers a way to organize work that is executed more than once during the MAS development lifecycle in repetitive cycles. Both encompass Medee MAS Activity method fragments.

Finally, in the process layer, a Medee MAS Process method fragment represents a whole MAS development lifecycle, by encompassing MAS Phase method fragments and/or MAS Iteration method fragments. Fragments in this layer can provide Medee MAS Base methods, which may be used as a starting point for the composition of Medee MAS situational methods.

3.2.4 Medee MAS work product framework

The Medee MAS Work Product Framework allows to define method fragments inputs, outputs, and objectives in a standardized way.

In order to achieve such a goal, this framework involves the following elements: **A**gent, **E**nvironment, **I**nteraction, **O**rganization, and **U**ser Requirements. The first four elements are used to encapsulate the work products and milestones related to the homonym MAS components proposed by the Vowel paradigm [20,21]. The last one deals with the notion of system-to-be requirements, and allows the method engineer to encapsulate work products and milestones, mainly generated during the MAS requirement phase, which describe the set of capabilities the MAS application is expected to provide.

Hence, using such a framework, a method engineer can state the work products that are involved in a Medee MAS Method Fragment in a standard manner, independently of the notation adopted by the MAS development approach from which the fragment was sourced. For instance, this framework allows clearly stating that the *Role model* sourced from Gaia and the *Role identification model* sourced from PASSI are not related to the same MAS component, as their names could suggest: the former is related to the organization component, while the latter is related to the agent component.

3.2.5 Medee common roles

The Medee Common Roles allow you to define, in a standardized way, the staff that is responsible to perform the pieces of work encompassed in a fragment. Therefore, it consists of a set of roles that are used for specifying the primary and additional performers assigned to a task in a MAS method fragment. They are: System Analyst, MAS Designer, MAS Developer, and MAS Tester.

These four roles subsume those commonly referred by AOSE methods. For instance, concerning requirements activities, PASSI proposes the *System Analyst role* and Tropos the *Requirements Engineer role*. Nonetheless, such a set of Medee roles may incorporate new ones whenever needed. For example, a new Medee common role may be defined to represent the role played by the MAS project team member responsible for planning the project and keeping the team focused on the project objectives, e. g. the MAS Project Manager role.

3.2.6 Medee method fragment views

The External and Internal views aim to improve standardization and coherence of the proposed MAS Method Fragment definition by depicting, respectively, the fragment interface and the set of SPEM elements that compose each fragment. They have been inspired by the concept of method component views proposed by Karlsson [40].

Thus, the Internal View offers a detailed and deep representation of SPEM elements involved in a method fragment elaboration, as well as the relationships between such elements.

In contrast, the External View allows method fragments to be analyzed as standard black boxes: it uses process patterns as a kind of *logical containers* for MAS Method Fragments; it applies categories for classifying method fragments according to semiotic criteria; and it encapsulates milestones and work products in terms of the Medee MAS Work Product Framework.

4 Medee composition model

4.1 Overview

In order to build MAS situational methods out of method fragments, it is important to describe how such fragments can be select according to a project situation. Therefore, the Medee Composition Model constitutes a key component of the Medee method framework: it allows the situational method to be tailored focused on leveraging project situation strengths and mitigating project situation weaknesses.

This model involves two taxonomies—the Medee Project Factors Taxonomy and the Medee MAS Semiotic Taxonomy—that are connected through the Medee Composition Guidelines. On one hand, the Medee Project Factors Taxonomy provides a structured way to characterize the MAS project situation through a set of project factors that covers four dimensions—people, problem, product, and resource—as proposed by Basili [2]. On the other hand, the Medee MAS Semiotic Taxonomy provides a broad categorization of MAS method fragments taking into account their semiotic aspects, among them pragmatic, semantic, and syntactical ones.

Finally, joining these two taxonomies, the Medee Composition Guidelines indicate the method fragment category that better deals with each project factor.

The path between a given project situation and a suitable Medee Situational Method encompasses several steps. Firstly, the project situation is assessed using the Medee Project Factors Taxonomy, by identifying a set of project factors that characterize the project. For instance, it involves a small team (people factor) and a short deadline (resource factor). Secondly, the guidelines associated with the identified project factors should be analyzed. Such guidelines suggest which aspects of a given situational method are suitable to handle these specific factors, and provide a list containing the Medee MAS Semiotic Taxonomy categories that cover such aspects. For instance, a guideline may indicate that a short deadline usually requires high productivity levels from the project team members, and thus shows the semiotic categories dealing with this aspect. Thirdly, the method fragments classified using the MAS semiotic categories can be selected as suitable candidates for being considered in the Situational Method. Finally, the MAS situational method can be composed out of the selected fragments, which aim to leverage project situation strengths and mitigate its weaknesses.

As previously mentioned, the Medee Composition Model is represented in terms of SPEM elements—namely, categories and guidelines—and built upon the EPF Composer. Therefore, the Medee Project Factors Taxonomy consists of a set of categories that are organized into a hierarchal structure of project factors, while the Medee Composition Guidelines consists of a collection of guidelines classified into those categories. The MAS Semiotic Taxonomy consists of a collection of nested categories that form a hierarchal semiotic structure, which are used to categorize MAS Method Fragments. Such categories are then pointed by the guidelines (through hyperlinks), closing the path between the project factors and the method fragments.

The Medee Project Factors Taxonomy, Medee MAS Semiotic Taxonomy and the Medee Composition Guidelines are presented next, while their use during the situational composition is described through the Medee Delivery Process, presented in Sect. 5.2.

4.2 Medee project factors taxonomy

The Medee Project Factors Taxonomy allows a MAS project situation to be characterized according to several aspects. It is based on the project factors proposed by Basili [2] as part of the QIP paradigm, which suggests that a project situation should be characterized according to several project factors organized into six dimensions: people, problem, process/method, product, resource, and tool factors.

Thus, on one hand, the Medee Project Factors Taxonomy used five out of these six factor dimensions, giving rise to the following four project factor categories: people, problem, product, and resource categories. The process/method dimension was not taken into account since it corresponds to the MAS situational method itself. Moreover, the resource and tool dimensions were collapsed into one category, since the MAS field does not encompass a high variety of resources and tools that justify treating them separately.

On the other hand, the project factors proposed in [2] were adapted to deal with MAS development aspects. For instance, some product related factors were tailored to reflect agent architectures, instead of dealing with general-purpose software architectures.

Figure 5 depicts the factors that form the Medee Project Factors Taxonomy as a UML class diagram, where *People factors* concern the MAS project team characteristics such as team size, application domain experience, etc.; *Problem factors* allow the characterization of the problem to be solved by the MAS application using the class of problem, state of problem definition, problem susceptibility to change, and problem related constraints; *Product factors* are those related to the software product itself and include aspects such as the number

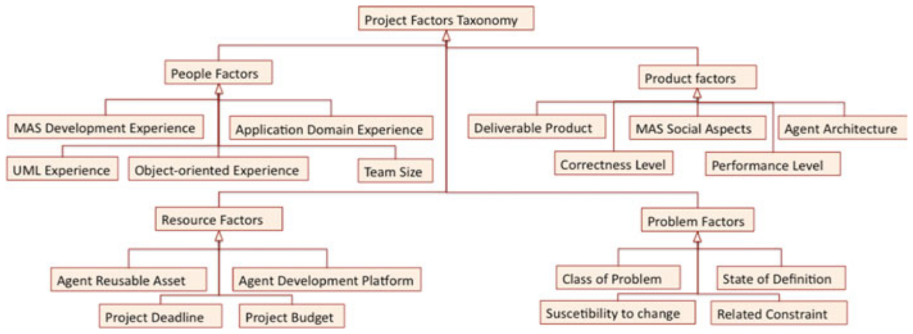


Fig. 5 Medee project factors taxonomy

of expected deliverable products, among others; finally, *Resource factors* concern nonhuman elements that are involved or consumed during software development, such as project deadlines.

4.3 Medee MAS semiotic taxonomy

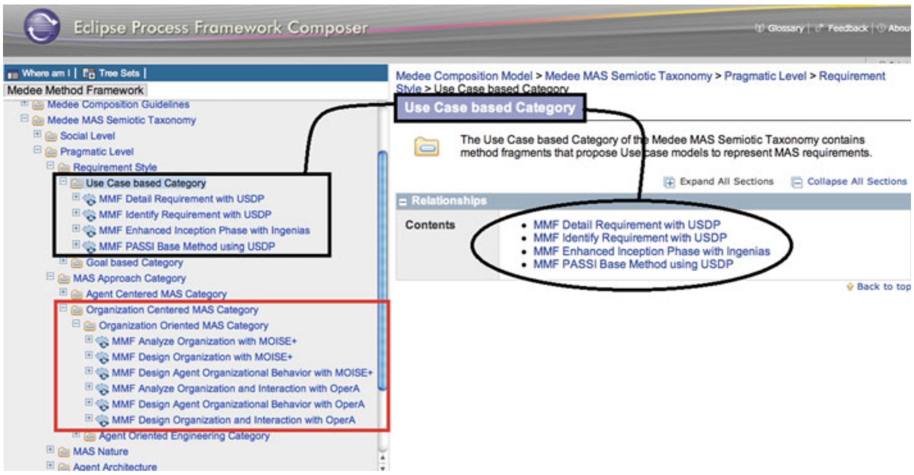
The categorization of MAS method fragments sourced from several AOSE methods and AO models is made by using the Medee MAS Semiotic Taxonomy [9,10], which is briefly described here. A detailed description is found in [10].

Semiotics deals with the syntactic (structure), semantics (meaning), and pragmatics (usage) aspects of signs. Based on the Semiotic Ladder proposed by Stamper in [54], we proposed the Medee MAS Semiotic Taxonomy to classify method fragments. This semiotic ladder extends the traditional division of Semiotics—syntactic, semantic and pragmatic—by including three new sign aspects called *social*, *empirics,* and *physical*. Thus, the proposed taxonomy involves the following levels: *Social, Pragmatic, Semantic, Syntactic,* and *Empirical levels*. The physical level is not taken into account because usually MAS development approaches do not deal with physical issues, such as specific hardware platforms for MAS.

Using such a semiotic perspective, this taxonomy brings concepts together from three main sources: (i) MAS specific development aspects originated from AOSE methods and AO models, (ii) Situational Method Engineering related concepts, mainly those proposed by Harmsen [30], and (iii) Software Engineering notions [39].

Given that MAS projects involve a group of developers embedded in a social context—as a software company or an academic research group —the *Social Level* allows you to distinguish MAS method fragments according to rules, preferences, and procedures related to the development context. The *Pragmatic Level* allows MAS method fragments to be distinguished based on their usage and intention. The *Semantic Level* allows MAS method fragments to be classified based on their meaning in the composition of a situational method. Therefore, this level is mainly concerned with the situational method engineering typical aspects. The *Syntactic Level* allows MAS method fragments to be classified according to their structure and format. This level takes into account categories related to the notation and to the language used to structure and to express them. Finally, the *Empirical Level* allows MAS method fragments to be classified according to their development standards and patterns.

The Medee MAS Semiotic Taxonomy use is twofold. Firstly, it is involved in a path between the MAS project situation and the Medee MAS Situational Method, as described in Sect. 4.1. Secondly, this taxonomy contributes to achieve standardization of Medee MAS method fragments, as explained in Sect. 3.



Medee Method Framework
- Medee Composition Guidelines
- Medee MAS Semiotic Taxonomy
  - Social Level
  - Pragmatic Level
    - Requirement Style
      - Use Case based Category
        - MMF Detail Requirement with USDP
        - MMF Identify Requirement with USDP
        - MMF Enhanced Inception Phase with Ingenias
        - MMF PASSI Base Method using USDP
      - Goal based Category
    - MAS Approach Category
      - Agent Centered MAS Category
      - Organization Centered MAS Category
        - Organization Oriented MAS Category
          - MMF Analyze Organization with MOISE+
          - MMF Design Organization with MOISE+
          - MMF Design Agent Organizational Behavior with MOISE+
          - MMF Analyze Organization and Interaction with OperA
          - MMF Design Agent Organizational Behavior with OperA
          - MMF Design Organization and Interaction with OperA
        - Agent Oriented Engineering Category
  - MAS Nature
  - Agent Architecture

Medee Composition Model > Medee MAS Semiotic Taxonomy > Pragmatic Level > Requirement Style > Use Case based Category

Use Case based Category

The Use Case based Category of the Medee MAS Semiotic Taxonomy contains method fragments that propose Use case models to represent MAS requirements.

Relationships

Contents
- MMF Detail Requirement with USDP
- MMF Identify Requirement with USDP
- MMF Enhanced Inception Phase with Ingenias
- MMF PASSI Base Method using USDP

Fig. 6 Medee MAS Semiotic Taxonomy used to classify method fragments for developing MAS

For instance, in a pragmatic perspective, the taxonomy can be used to support the search for fragments to model system requirements in terms of use cases and to deal with organization-centered MAS (Fig. 6), since the project team is skilled in UML and the software product to be built involves an agent organization.

4.4 Medee composition guidelines

The goal of the Medee Composition Guidelines is to show the association between project factors and method fragment categories, providing a kind of glue that joins the Medee Project Factors Taxonomy and Medee MAS Semiotic Taxonomy.

In order to do that, these guidelines offer a rationale that points out a suitable method fragment category to deal with a specific project factor. They use informal inference rules and best practices for developing software, mainly based on Software Engineering and AOSE principles.

For example, the Unstable Requirements guideline (Table 1) states the following (informal) inference rule: changes in the problem to be solved during the software development usually involve modifying the system requirements previously captured. Besides, it suggests the following best practices—MAS method fragments that provide high user participation and high iteration can help handling continuous requirement changes during the MAS project—which is part of the common knowledge of Software Engineering best practices: iteration and user participation can mitigate the risks involved in unstable system requirements [39,41,53].

Moreover, a composition guideline can be associated with one or more project factors. For instance, the Unstable Requirements guideline is associated with two problem factors, State of Problem Definition and Problem Susceptibility to Change, as illustrated in Table 1 and Fig. 7. Thus, this guideline joins these two project factors to the following method fragment categories—High User Participation and High Iterative Degrees from the Social Level—and the Requirement Discipline Category from the Semantic Level of the Medee MAS Semiotic Taxonomy.

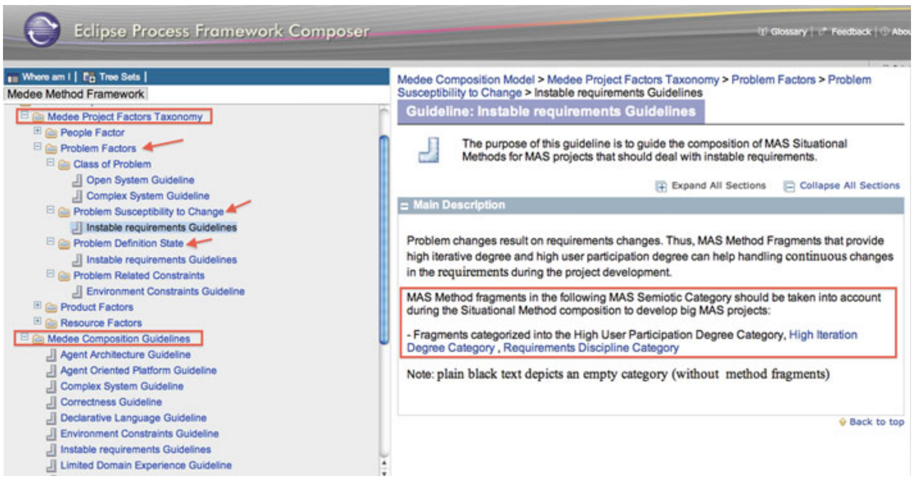
Currently, the Medee Composition Guidelines encompass 17 (seventeen) elements, as shown in Table 1. Furthermore, this table shows the association between such guidelines and the

Table 1 Example of Medee Composition Guidelines

Project factors taxonomy		Medee composition guideline	MAS semiotic taxonomy	
Project factor	Dimension		Category	Level
Agent architecture	Product	Agent Architecture Guideline	Agent Architecture Category	Pragmatic
Agent development platform	Resource	Agent-oriented Platform Guideline	Development Platform Category	Empiric
Class of problem	Problem	Complex System Guideline	MAS Approach Category	Pragmatic
Correctness	Product	Correctness Guideline	Validation Degree Category	Social
			User Participation Degree Cat.	
			Fragment Discipline Category	Semantic
Problem related constraints	Problem	Environment Constraints Guideline	MAS Approach Category	Pragmatic
			MAS Nature Category	
Problem susceptibility to change	Problem	Unstable requirements Guideline	User Participation Degree Cat.	Social
			Iteration Degree Category	
Problem definition state			Fragment Discipline Category	Semantic
Application domain experience	People	Limited Domain Experience Guideline	User Participation Degree Cat.	Social
			Iteration Degree Category	
			Development Type Category	
Deliverable product	Product	MAS Component Guideline	MAS Component Category	Pragmatic
MAS development experience	People	MAS Development Experience Guideline	Fragment Source Category	Semantic
MAS social aspect	Product	MAS Social Aspect Guideline	MAS Approach Category	Pragmatic
Project team size	People	Method Ceremony Guideline	Discipline Category	Semantic
MAS development experience				
Deliverable products	Product			
Project deadline	Resource			
Object oriented method experience	People	Object Oriented Method Experience Guideline	Fragment Root Category	Semantic
Class of problem	Problem	Open System Guideline	MAS Nature Category	Pragmatic
Available reusable assets	Resource	Organization Reusable Asset Guideline	Reusable Assets Category	Empiric

Table 1 continued

Project factors taxonomy		Medee composition guideline	MAS semiotic taxonomy	
Project factor	Dimension		Category	Level
Performance level	Product	Performance Guideline	Code Generation Category	Empiric
			Development Platform Category	
			Fragment Language Category	Syntactic
Project deadline	Resource	Productivity Guideline	Reuse Degree Category	Social
			Success Degree Category	
Project budget			Code Generation Category	Empiric
			Reusable Assets Category	
UML experience	People	UML Experience Guideline	Fragment Language Category	Syntactic


Eclipse Process Framework Composer

Glossary | Feedback | About

Where am I | Tree Sets |

Medee Method Framework

- Medee Project Factors Taxonomy
  - People Factor
  - Problem Factors
    - Class of Problem
      - Open System Guideline
      - Complex System Guideline
    - Problem Susceptibility to Change
      - Instable requirements Guidelines
    - Problem Definition State
      - Instable requirements Guidelines
    - Problem Related Constraints
      - Environment Constraints Guideline
  - Product Factors
  - Resource Factors
- Medee Composition Guidelines
  - Agent Architecture Guideline
  - Agent Oriented Platform Guideline
  - Complex System Guideline
  - Correctness Guideline
  - Declarative Language Guideline
  - Environment Constraints Guideline
  - Instable requirements Guidelines
  - Limited Domain Experience Guideline

Medee Composition Model > Medee Project Factors Taxonomy > Problem Factors > Problem Susceptibility to Change > Instable requirements Guidelines

Guideline: Instable requirements Guidelines

The purpose of this guideline is to guide the composition of MAS Situational Methods for MAS projects that should deal with instable requirements.

Expand All Sections Collapse All Sections

Main Description

Problem changes result on requirements changes. Thus, MAS Method Fragments that provide high iterative degree and high user participation degree can help handling continuous changes in the requirements during the project development.

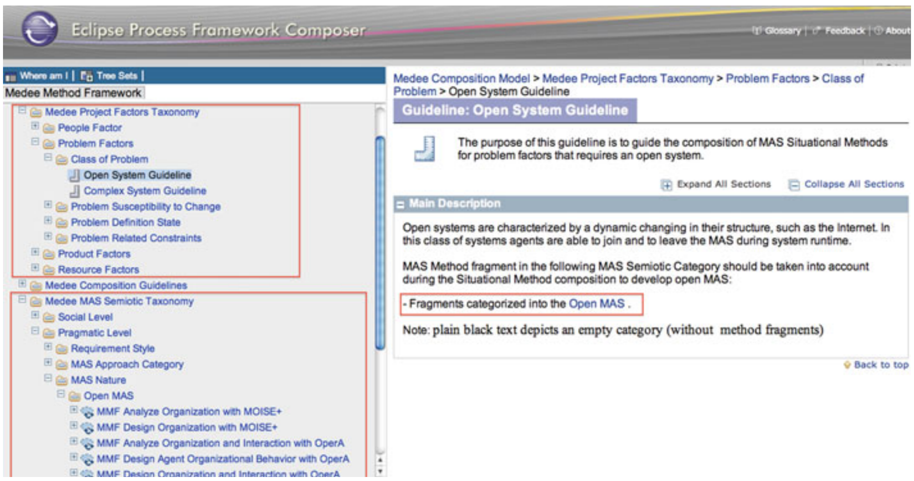
MAS Method fragments in the following MAS Semiotic Category should be taken into account during the Situational Method composition to develop big MAS projects:

- Fragments categorized into the High User Participation Degree Category, High Iteration Degree Category , Requirements Discipline Category

Note: plain black text depicts an empty category (without method fragments)

Back to top

Fig. 7 Unstable Requirements guideline and its inter-relationship with Medee Project Factors Taxonomy and Medee MAS Semiotic Taxonomy


Eclipse Process Framework Composer

Glossary | Feedback | About

Where am I | Tree Sets |

Medee Method Framework

- Medee Project Factors Taxonomy
  - People Factor
  - Problem Factors
    - Class of Problem
      - Open System Guideline
      - Complex System Guideline
    - Problem Susceptibility to Change
    - Problem Definition State
    - Problem Related Constraints
  - Product Factors
  - Resource Factors
- Medee Composition Guidelines
- Medee MAS Semiotic Taxonomy
  - Social Level
  - Pragmatic Level
    - Requirement Style
    - MAS Approach Category
    - MAS Nature
      - Open MAS
        - MMF Analyze Organization with MOISE+
        - MMF Design Organization with MOISE+
        - MMF Analyze Organization and Interaction with OperA
        - MMF Design Agent Organizational Behavior with OperA
        - MMF Design Organization and Interaction with OperA

Medee Composition Model > Medee Project Factors Taxonomy > Problem Factors > Class of Problem > Open System Guideline

Guideline: Open System Guideline

The purpose of this guideline is to guide the composition of MAS Situational Methods for problem factors that requires an open system.

Expand All Sections Collapse All Sections

Main Description

Open systems are characterized by a dynamic changing in their structure, such as the Internet. In this class of systems agents are able to join and to leave the MAS during system runtime.

MAS Method fragment in the following MAS Semiotic Category should be taken into account during the Situational Method composition to develop open MAS:

- Fragments categorized into the Open MAS .

Note: plain black text depicts an empty category (without method fragments)

Back to top

Fig. 8 Medee Composition Model elements and their relationships

project factors of the Medee Project Factors Taxonomy (see columns titled *project factor* and *dimension*), as well as the semiotic categories indicated by them (see columns titled *category* and *level*).

An important point to note is that the Medee Composition Guidelines are more of a placeholder schema, instead of being a broad, complete, and exhaustive set of guidelines for MAS composition. Rather, it aims to provide an initial sub-set of elements that should be completed and refined through the experience of using the Medee method framework.

Summing up with the Medee composition model, Fig. 8 shows its three elements—the Project Factors Taxonomy, Semiotic Taxonomy, and Composition Guidelines—built upon the EPF Composer. On one hand, this figure (upper left frame) illustrates the categories from the Medee Project Factors Taxonomy, showing that one of the product factors, the Class of Problem, is related to the Open System Guideline and the Complex System Guideline. On the other hand, the Open System Guideline (Fig. 8, right) indicates that one category of the MAS Semiotic Taxonomy may encompass appropriate method fragments to deal with such a product factor, the Open MAS category, which is part of the MAS Nature category at the Pragmatic Level (Fig. 8, left down).

5 Medee method repository and delivery process

Having a well-founded definition of MAS method fragments, a well established way of categorizing them, and a model for guiding the situational selection and composition of those fragments, a repository and a process for managing fragments and situational methods is needed. Therefore, a three-layered architecture was defined to describe the way in which the Medee Method Repository is organized.

Moreover, the Medee Delivery Process was developed to specify how to capture method elements, to elaborate method fragments, and to compose situational methods using this repository (Fig. 9). Both were built upon the EPF Composer and are described in the next subsections.

5.1 Medee Method repository architecture

As illustrated in Fig. 10, the Medee Method Repository is organized in a layered architecture, in which the first pillar provides method elements to define MAS method fragments stored in the second pillar, which in turn provides method fragments to compose the Medee Methods stored in the third pillar.

The first pillar—Medee Elements Pillar—consists of the repository foundation: it stores method elements captured from the MAS development approaches, such as AOSE methods and AO models. The captured data is modeled, documented, and managed as a collection of SPEM elements, such as activities, tasks, roles, work products, guidance, and categories. Furthermore, this pillar may store whole AOSE methods represented in terms of these SPEM elements, called AOSE method As Is, which offer a common basis for comparing such methods since they are built without any reference to the method fragment related notions. Indeed, this pillar does not store any MAS method fragment, since such fragments are stored in the second pillar, the Medee Fragments Pillar.

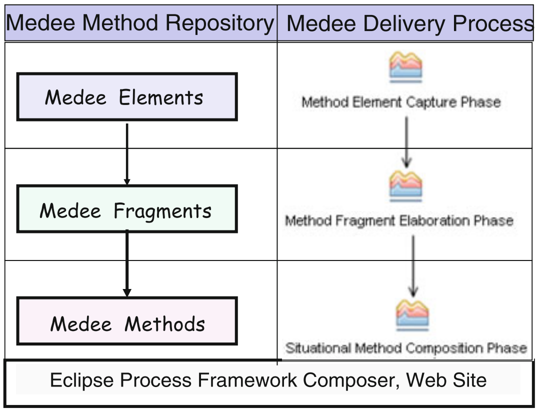

Medee Method Repository
Medee Delivery Process
Medee Elements
Method Element Capture Phase
Medee Fragments
Method Fragment Elaboration Phase
Medee Methods
Situational Method Composition Phase
Eclipse Process Framework Composer, Web Site

Fig. 9 Medee Method Repository layers and Delivery Process phases

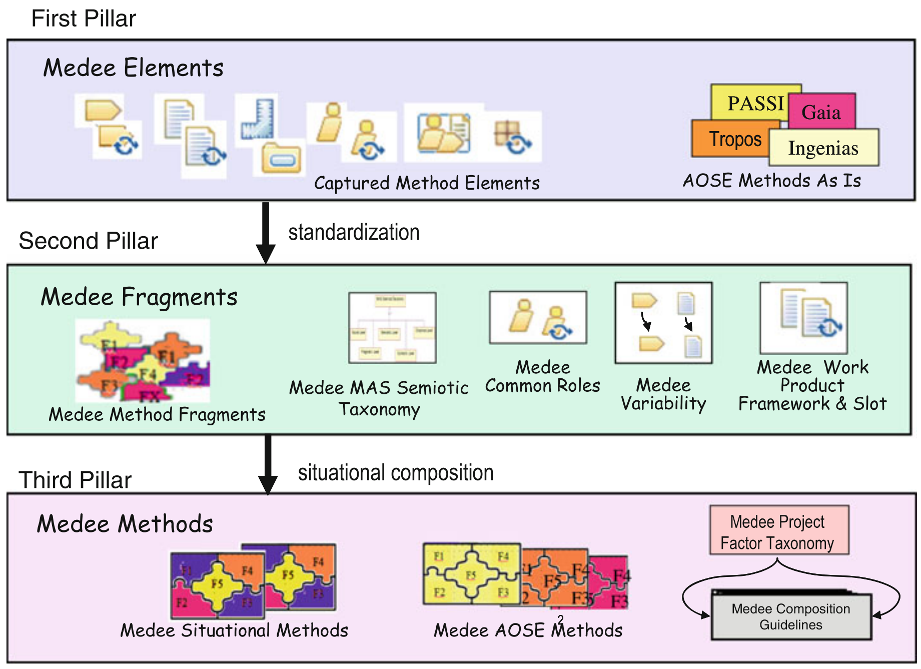


Fig. 10 Medee Method Repository layers content

Thus, MAS method fragments stored in the second pillar are built upon the SPEM elements stored in the previous one, as depicted in Fig. 10. However, before being used as part of a fragment, these SPEM elements are extended to achieve standardization and coherence required by the MAS method fragment definition, as explained in Sect. 3.1. Therefore, the Medee Fragments Pillar stores the four granularity layers of method fragments (Sect. 3.2) sourced from AOSE methods and AO models, each of them categorized by the MAS Semiotic Taxonomy (Sect. 4.3), encapsulated by the MAS Work Product Framework (Sect. 3.2), and involving MAS common development roles (Sect. 3.2). Moreover, this pillar stores the categories that form the MAS Semiotic Taxonomy itself. Hence, the second pillar constitutes the kernel of the Medee Repository.

Finally, the Medee Methods Pillar stores two kinds of methods: Medee Situational Methods and Medee AOSE Methods. The former are composed according to a given project situation, encompassing fragments usually sourced from several MAS development approaches. The latter consist of AOSE methods that have been reengineered in terms of MAS method fragments. Furthermore, this pillar stores the categories and guidelines that form the Medee Project Factors Taxonomy and Medee Composition Guideline, respectively.

In brief, the third pillar constitutes the consumption-side of the Medee Method Repository, while the first and second ones constitute its supply-side, storing the building blocks, i.e. method elements and method fragments, for composing Medee methods, i.e. Medee situational methods and Medee AOSE methods.

5.1.1 Method fragment standardization and situational composition

The following strategy—based on the extension of method elements—is adopted to standardize fragments sourced from several approaches in order to facilitate their composition in situational methods. Firstly, method elements captured from MAS development approaches are modeled and documented as SPEM elements following their original definitions, and then stored in the Medee Elements Pillar.

Secondly, these method elements are extended to allow the elaboration of method fragments standardized and suitable for situational use. Such an extension is due to several reasons, among them (i) MAS development approaches may not adopt a standard nomenclature for designating work products generated during a MAS project, (ii) AOSE methods may not adopt a common set of development roles, (iii) MAS development approaches may not explicitly state tasks inputs and outputs, and finally (iv) MAS development approaches may not clearly state how to share work products among them.

Thus, along with the use of the Medee MAS Work Product Framework for encapsulating the original work products, and the Medee Common Roles for replacing the original roles, the following concepts are involved in the proposed strategy: Medee MAS task variability, Medee MAS work product variability, and Medee MAS work product slot. The first two allow you to extend tasks and work products stored in the Medee Elements pillar without modifying them. Roughly speaking, it is done through a variability mechanism supplied by SPEM that allows method elements (e.g. task, work product) to be extended. For instance, an original task captured from Tropos may be extended to a new task that replaces the original roles by some of the Medee Common Roles, and the original input work product by a new one defined according to the Medee MAS Work Product Framework. At the end of this procedure we have two tasks stored in the Medee repository: the original task captured from Tropos (in the first pillar) and the new standardized task (in the second pillar).

Medee MAS work product slot allows work products to be shared among MAS method fragments sourced from distinct MAS development approaches, offering a kind of glue for concatenating fragments during situational method composition. In order to achieve such a fragment concatenation, the Medee MAS work product slot concept is underpinned by a quasi-homonym feature from the EPF Composer that handles work products: the work product slot. In brief, it consists of an abstract work product that represents a placeholder for concrete ones. The fulfillment of one work product slot by a concrete work product is dynamically performed by the EPF Composer.

Furthermore, MAS work product slots may be fulfilled by some MAS work product variability as well as be associated with MAS task variability as input and/or output. Such associations provide great flexibility for handling MAS method fragments during method composition: fragments inputs and outputs can be defined in terms of placeholders that are fulfilled by the MAS work product variability available in the method configuration relating to the composition of situational methods.

Indeed, such flexibility consists of one of the cornerstone of the situational method composition using the Medee Method Framework, since it provides a seamless flow of work products between method fragments sourced from distinct MAS development approaches, as illustrated by the example presented in Sect. 6.4. A set of Medee work product slots was defined and stored in the Medee Fragments pillar, among them the MPS[5] Agent-Analysis, MPS Environment-Design, MPS Organization-Analysis, and MPS User Requirements. In this way, such Medee strategy allows method elements to be captured once and then to be reused whenever needed. For example, method elements such as tasks and work products captured from Tropos may be used for (i) building Tropos As Is method in terms of SPEM elements, (ii) elaborating MAS

5 MPS stands for MAS work Product Slot.

method fragments in several layers (e.g. activity, phase, process), and (iii) composing Medee methods out of these fragments.

All of this strategy is documented and detailed in the Medee Delivery Process, as shown next.

5.2 Medee Delivery Process

This section details the Medee Delivery Process, the component of the Medee Method Framework in charge of specifying how to populate the three pillars of the Medee Method Repository according to the characteristics and strategy shown in Sect. 5.1.

As previously mentioned, the Medee Delivery Process itself is defined in terms of SPEM elements, elaborated in the EPF Composer, and published on HTML pages. Therefore, it is built upon phases, activities, tasks, steps, work products, roles, and delivery process. Indeed, the Medee Delivery Process encompasses 3 phases, 11 activities, 17 tasks, and 93 steps. Such tasks involve 25 work products (as input and/or output), 3 roles (method engineer, MAS development approach expert, project manager), and a couple of guidance. Thus, it describes in detail how to manage the Medee Repository and how to use it for building and publishing Medee situational methods on websites.

The Medee Delivery Process phases are: Medee Method Element Capture, Medee Method Fragment Elaboration, and Medee Method Composition. As their names indicate, each phase deals with a specific pillar of the Medee Method Repository. Moreover, the Medee Delivery Process has two workflows, as illustrated in Fig. 11 through an Activity diagram[6]: one for capturing information from MAS development approaches, creating method elements and elaborating MAS method fragments, and the other for composing Medee methods.

5.2.1 Medee method element capture phase

The purpose of the Method Element Capture phase is to populate the first pillar of the Medee repository with the method elements captured from MAS development approaches. It involves three activities: Capture method content, Build AOSE method as is, and Publish AOSE method as is. The last two activities are performed whenever the MAS development approach is an AOSE method, such as Gaia and Tropos. Otherwise, these activities are skipped. This phase is conducted by a method engineer (primary role) with the optional support from an expert on MAS development (additional role).

The purpose of the Capture method content activity is to analyze, model, and store the knowledge captured from a given MAS development approach as a collection of SPEM elements, such as task, work product, role, categories, and guidance. This activity encompasses two tasks: the Outline Method Content task involves a thorough analysis of the MAS development approach literature to identify the main SPEM elements that can be used to represent them, while the Detail Method Content task consists of modeling such elements in detail. For instance, the documentation of AOSE methods following the IEEE FIPA template [37] is one of the input work products for this activity.

The purpose of the Build AOSE Method As Is activity is to build up the AOSE method as a whole, based on SPEM elements captured during the previous activity. Finally, the Publish

6 An Activity diagram illustrates how the process elements (e.g. activities, phases, iterations) flow together. Such diagrams are defined by SPEM and provided by the EPF Composer.

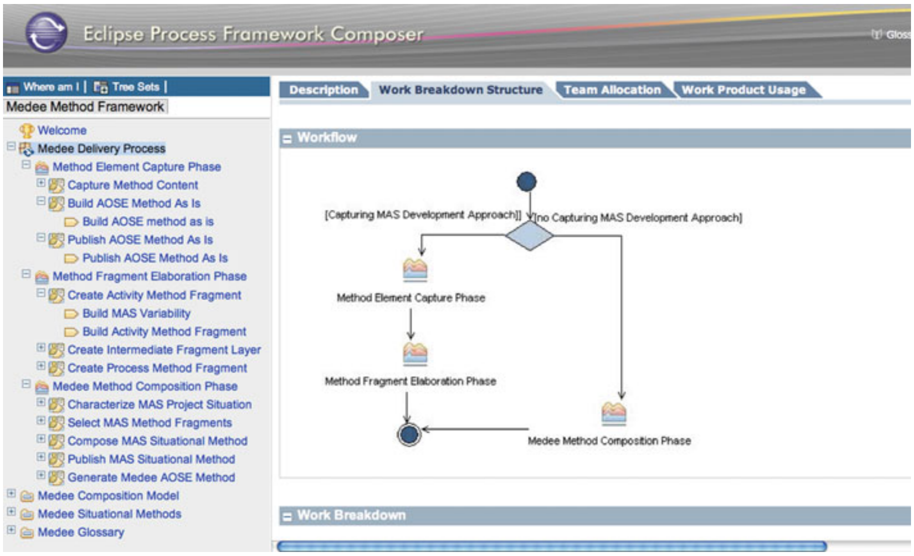

Eclipse Process Framework Composer
Where am I | Tree Sets |
Medee Method Framework
Welcome
Medee Delivery Process
Method Element Capture Phase
Capture Method Content
Build AOSE Method As Is
Build AOSE method as is
Publish AOSE Method As Is
Publish AOSE Method As Is
Method Fragment Elaboration Phase
Create Activity Method Fragment
Build MAS Variability
Build Activity Method Fragment
Create Intermediate Fragment Layer
Create Process Method Fragment
Medee Method Composition Phase
Characterize MAS Project Situation
Select MAS Method Fragments
Compose MAS Situational Method
Publish MAS Situational Method
Generate Medee AOSE Method
Medee Composition Model
Medee Situational Methods
Medee Glossary
Description
Work Breakdown Structure
Team Allocation
Work Product Usage
Workflow
[Capturing MAS Development Approach]
[no Capturing MAS Development Approach]
Method Element Capture Phase
Method Fragment Elaboration Phase
Medee Method Composition Phase
Work Breakdown

Fig. 11 Medee Delivery Process: workflows and phases

AOSE Method As Is activity aims to publish the AOSE method As Is built up during the previous one as a fully hyperlinked collection of HTML pages.

5.2.2 Medee Method Fragment Elaboration phase

The purpose of the Medee Method Fragment Elaboration phase is to populate the second pillar of the Medee Repository, the Medee Fragments Pillar, with method fragments built on method elements stored in the repository first pillar (Sect. 5.1). This phase involves three activities: Create activity method fragment, Create intermediate fragment layer, and Create process method fragment. The latter is performed whenever the MAS development approach is an AOSE method, such as Gaia and Tropos. Otherwise, it is skipped.

The Create activity method fragment activity aims to elaborate Medee Method Fragments of the activity layer (Sect. 3.2). It encompasses two tasks: Build MAS variability and Build activity method fragment. The first consists of extending the captured elements stored in the first pillar (like work products and tasks) so that they can be used to create new method fragments. This is made through MAS work product variability and MAS task variability, as aforementioned. The second task consists of using such MAS variability elements to elaborate Medee activity method fragments. At the end of this activity, the Medee repository will contain the MAS activity method fragments ready for use to elaborate fragments in the upper layers of granularity (Sect. 3.2).

The purpose of the Create intermediate fragment layer activity is to elaborate MAS method fragments in the phase and iteration layers of granularity, using existing MAS activity method fragments.

Finally, the Create process method fragment activity aims to generate the fragments that provide MAS Base Methods during a top–down situational method composition. At the end of this activity, a new MAS process method fragment will be stored in the Medee fragments pillar to be used for composing Medee situational methods using a top–down approach.

5.2.3 Medee Situational Method Composition phase

The Medee Situational Method Composition phase aims to populate the third pillar of the Medee repository with two kinds of methods: Medee situational methods and Medee AOSE methods, where the former are composed according to a given project situation and the latter represent AOSE methods described as a set of Medee Method fragments. There are four activities in charge of composing and publishing MAS situational methods: Characterize MAS Project Situation, Select MAS Method Fragments, Compose MAS Situational Method, and Publish MAS Situational Method. Moreover, there is only one activity in charge of building and publishing a Medee AOSE method: the Generate Medee AOSE Method (Fig. 12).

This phase is strongly based on the Medee Composition Model, along with some aspects inspired by Harmsen [30], Brinkkemper [7], and Karlsson [40].

The purpose of the Characterize MAS Project Situation activity is to analyze the characteristics of a given project situation by assessing its relevant factors through the four dimensions of the Medee Project Factors Taxonomy: people, problem, product, and resource. In summary, this activity consists of determining the project factors that best characterize the MAS project. For instance, the current project involves a small team (people factor), a dynamic MAS environment (problem factor), a broad set of deliverables (product factor), and a short deadline (resource factor).

Based on the Medee Composition Model, the Select MAS Method Fragments activity aims to identify and to select those method fragments that are more suitable for the MAS project situation. For this, it encompasses two tasks: Select Candidate MAS Method Fragment and Analyze MAS Base Method. The first consists of identifying appropriate fragments for the current MAS project situation, based on the Medee composition guidelines. Thus, it involves analyzing these guidelines, inspecting the semiotic categories indicated by them, and thus selecting the candidate method fragments among those included in inspected categories. At the end, this task generates a Method Fragment Preliminary List, which contains a preliminary selection of MAS method fragments that should be considered during the situational composition. The

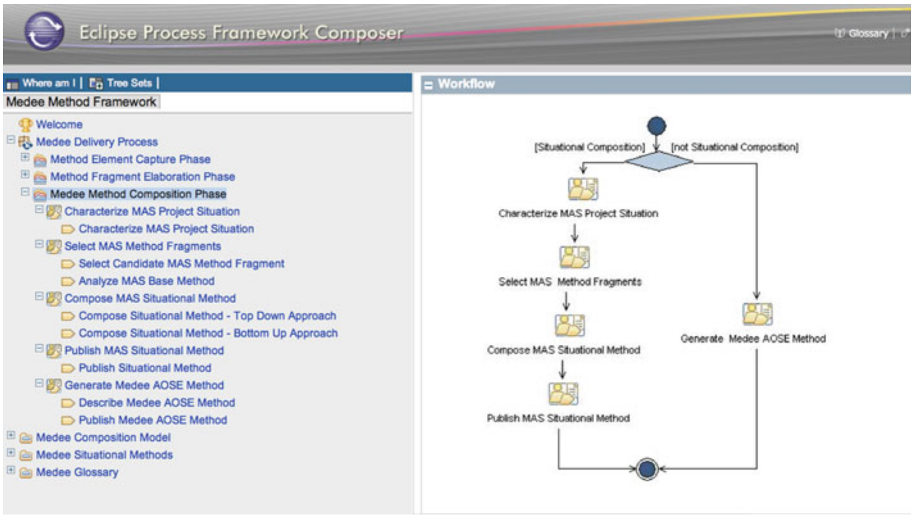

Eclipse Process Framework Composer
Glossary
Where am I | Tree Sets |
Workflow
Medee Method Framework
Welcome
Medee Delivery Process
Method Element Capture Phase
Method Fragment Elaboration Phase
Medee Method Composition Phase
Characterize MAS Project Situation
Characterize MAS Project Situation
Select MAS Method Fragments
Select Candidate MAS Method Fragment
Analyze MAS Base Method
Compose MAS Situational Method
Compose Situational Method - Top Down Approach
Compose Situational Method - Bottom Up Approach
Publish MAS Situational Method
Publish Situational Method
Generate Medee AOSE Method
Describe Medee AOSE Method
Publish Medee AOSE Method
Medee Composition Model
Medee Situational Methods
Medee Glossary
[Situational Composition]
[not Situational Composition]
Characterize MAS Project Situation
Select MAS Method Fragments
Compose MAS Situational Method
Publish MAS Situational Method
Generate Medee AOSE Method

Fig. 12 Medee Method Composition Phase activities

Analyze MAS Base Method task aims to verify whether it is possible to choose a MAS base method among the MAS process method fragments included in the Method Fragment Preliminary List. If this is the case, such a list is refined—by eliminating fragments that already pertain to the MAS base method and those that can be easily replaced by another one belonging to it—giving rise to a Method Fragment Final list.

The Compose MAS situational method activity aims to build a MAS situational method according to the definition presented in Sect. 3.2, by adopting a suitable composition mechanism for the current project situation: top–down [40] or bottom–up [7,30]. The former may be adopted whenever the Method Fragment Final List includes a MAS base method. Otherwise, the latter may be used. A top–down composition consists of tailoring the selected MAS base method, by adding MAS method fragments captured from other MAS development approaches, and eventually eliminating those MAS method fragments that are not required for the MAS project situation. Conversely, a bottom–up composition consists of specifying a work breakdown structure for the new MAS situational method from scratch, by outlining the sequence of iterations, phases, and milestones of the situational method, and then assembling the selected method fragments into these work breakdown elements.

Finally, the Publish MAS situation method activity aims to publish the composed situational method as a fully hyperlinked collection of HTML pages.

5.3 Medee MAS method repository contents

Currently, the Medee Repository is populated with 260 (two hundred and sixty) Medee elements and 64 (sixty four) Medee fragments sourced from AOSE methods such as Gaia, Tropos, PASSI, and Ingenias, from AO models such as MOISE+ and Opera, and from general-purpose development methods like Unified Software Development Process (USDP) [39], as depicted in Table 2. Moreover, such a repository contains AOSE methods represented in terms of SPEM elements (see column AOSE method As Is) and in terms of Medee fragments (see column Medee AOSE methods).

It should be observed that our approach aims to provide method elements and method fragments that can be easily modified and enhanced, instead of offering the best description of them, since AOSE methods and AO models are among the software artifacts that should be continually improved.

The remainder of this section presents examples involving the two first phases of the Medee Delivery Process—Method Element Capture and Method Fragment Elaboration phases—while an example of the Medee Method Composition is given in Sect. 6.

5.4 Examples of Medee elements and fragments

In this section, we present examples of Medee elements and Medee fragments captured from MOISE+ and Tropos. They belong to the first and second pillars of the repository, respectively, and will be further used to compose a Medee situational method (Sect. 6).

5.4.1 Medee elements captured from MOISE+

MOISE+ [33,35,36] is an agent organization model that proposes three organizational dimensions to explain how a MAS organization can be described: a *Structural Specification*, a *Functional Specification,* and a *Deontic Specification*. Moreover, MOISE+ literature proposes a

Table 2 Medee method repository contents

MAS development approach	Captured method elements					AOSE Method as i s	Sourced Medee MAS method fragment					Medee AOSE method
	Task	Work product	Role	Guidance	Total		Activity	Phase	Iteration	Process	Total	
Gaia	10	9	0	14	33	1	7	3	0	2	12	2
Tropos	10	9	1	30	50	1	8	4	0	1	13	1
PASSI	14	15	5	22	56	1	10	5	1	1	17	1
Ingenias	14	6	8	12	40	1	10	2	0	0	12	0
USDP	2	8	0	2	12	0	2	0	0	0	2	0
MOISE+	5	5	0	31	41	0	5	0	0	0	5	0
OperA	5	7	0	16	28	0	3	0	0	0	3	0
TOTAL	60	59	14	127	260	4	45	14	1	4	64	4

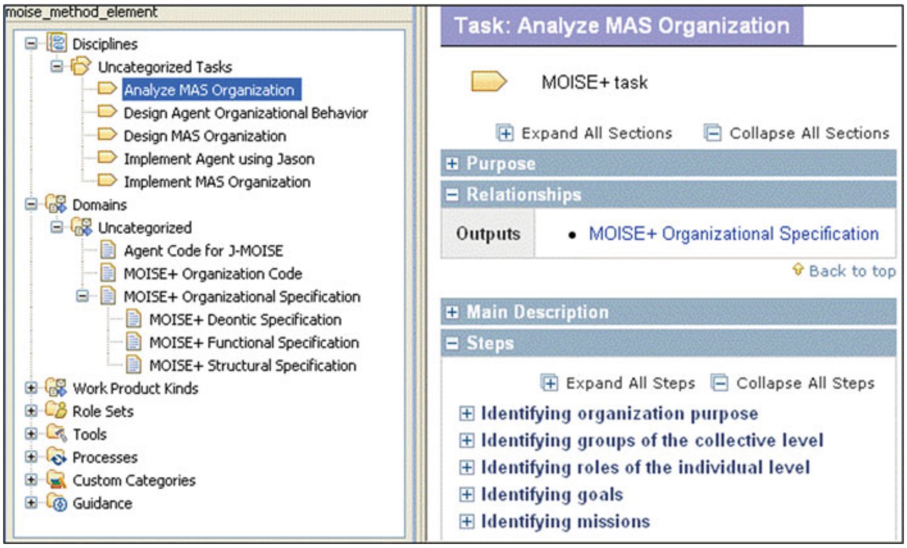


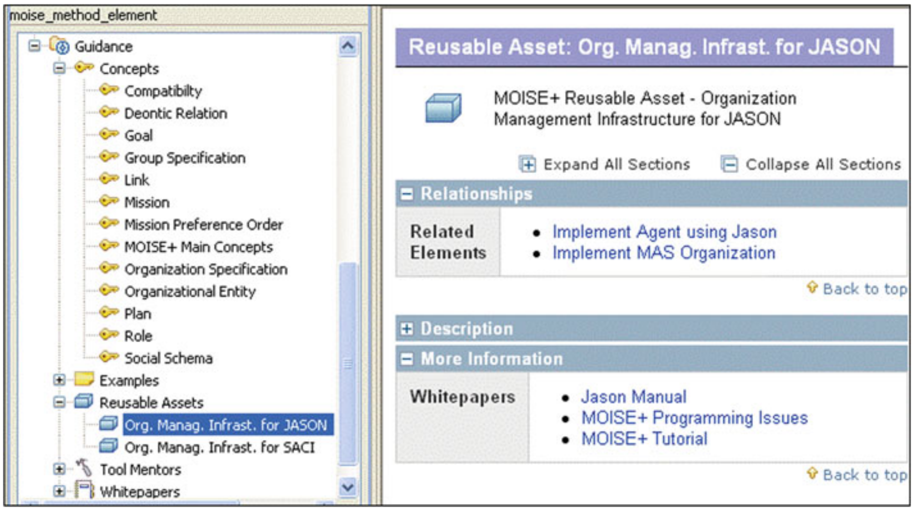
Fig. 13 Tasks and work products captured from MOISE+, detailing the Analyze MAS Organization task

tool for simulating these specifications, called *Organizational Entity Dynamic Simulator* [34], and reusable assets for implementing organizational management infrastructure: the *J-MOISE+* and the S-MOISE+ [35,36]. The former is developed on Jason [4,5], which is an agent-oriented development platform.

The Method Element Capture phase was performed to populate the Medee Element pillar with several tasks, work products, and guidance captured from MOISE+, as illustrated in Figs. 13, 14.

As depicted in Fig. 13 (left down side), the work products captured from MOISE+ are the following: *Agent Code for J-MOISE*; *MOISE+ Organization Code*; and *MOISE+ Organizational Specification,* encompassing structural, functional, and deontic specifications. The first two are implicitly outlined in [36] while the last one is explicitly defined in MOISE+ main references [33,35]. Moreover, Fig. 13 (left upper side) illustrates the five tasks concerning the generation of these work products: Analyze MAS Organization, Design Agent Organizational Behavior, Design MAS Organization, Implement Agent using Jason and Implement MAS Organization. It is important to note that such tasks are not explicitly proposed in the MOISE+ literature. Hence, they are the result of our analysis and interpretation of this literature.

Along with these tasks and work products, several guidance captured from MOISE+ were also described as Method Elements, as illustrated in Fig. 14 (left side). Such guidance encompasses concepts for describing MOISE+ main notions, examples of MOISE+ specifications, whitepapers that provide links to the MOISE+ literature, tool mentors describing the tools available for dealing with the MOISE+ specifications, such as the Organizational Entity Dynamics Simulator, and reusable assets that can speed up the development of a MOISE+ organization. One of these reusable assets, the Organizational management infrastructure for Jason (J-MOISE+), is depicted in detail in Fig. 14 (right side): it is related to the Implement MAS Organization and Implement Agent using Jason tasks. Additionally, three whitepapers offer more information about this reusable asset: *Jason manual* [4], *MOISE+ tutorial* [36], and *MOISE+ Programming Issues* [35].



Reusable Asset: Org. Manag. Infrast. for JASON

MOISE+ Reusable Asset - Organization Management Infrastructure for JASON

Relationships

Related Elements	• Implement Agent using Jason • Implement MAS Organization

Description

More Information

Whitepapers	• Jason Manual • MOISE+ Programming Issues • MOISE+ Tutorial

Fig. 14 Guidance captured from MOISE+, detailing the Organization Management Infrastructure for Jason

5.4.2 Medee fragments sourced from MOISE+

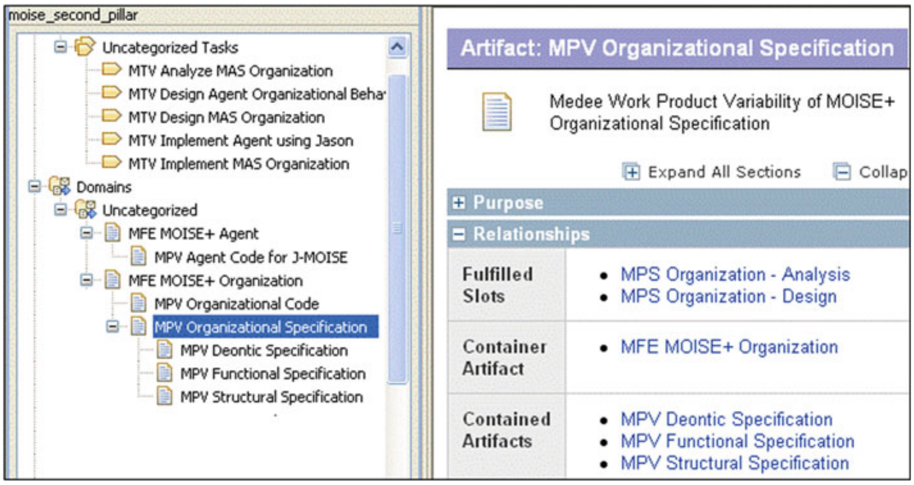
Since MOISE+ does not offer phases, iterations, or a whole development process, the population of the Medee Fragment pillar with MAS method fragments sourced from this AO model only involved the first activity proposed by the Method Fragment Elaboration phase, the Create Activity Method Fragment activity.

Firstly, the work products and tasks described in the previous section have been extended using MAS work product variability and MAS task variability, before giving rise to the five MAS activity method fragments sourced from MOISE+. We adopted prefixes to identify MAS Method Fragments (MMF), MAS Task Variability (MTV), MAS work Product Variability (MPV), MAS work Product Slot (MPS), and MAS work product Framework Element (MFE).

Figure 15 depicts one of these MAS work product variability, the MPV Organization Specification, which extends MOISE+ Organizational Specification by stating that it fulfills two slots, the MPS Organization Analysis and MPS Organization Design, since this work product covers both the analysis and the design of a MAS organization. Moreover, it is encapsulated by the MFE MOISE+ Organization. It also contains the MPV Deontic Specification, MPV Functional Specification, and MPV Structural Specification.

In this way, MAS work product variability allows to clearly state two important situational aspects of a work product: its standardization in terms of the MAS Work Product Framework, and the particular slot that it can fulfill among those proposed by the set of Medee MAS work product slots.

Moreover, through MAS task variability we have extended captured tasks. For instance, as shown in Fig. 16 (right side), the MTV Analyze MAS Organization—that extends the quasi-homonym task—has a primary performer's role assigned out of the Medee common role set (MAS designer and System analyst). Furthermore, it specifies that the MPS User Requirements is its mandatory input, the MPS Environment-Analysis is its optional input, and the MPV MOISE+ Organizational Specification is its output (replacing the original task output by the related work product variability).



moise_second_pillar

- Uncategorized Tasks
  - MTV Analyze MAS Organization
  - MTV Design Agent Organizational Beha
  - MTV Design MAS Organization
  - MTV Implement Agent using Jason
  - MTV Implement MAS Organization
- Domains
  - Uncategorized
    - MFE MOISE+ Agent
      - MPV Agent Code for J-MOISE
    - MFE MOISE+ Organization
      - MPV Organizational Code
      - MPV Organizational Specification
        - MPV Deontic Specification
        - MPV Functional Specification
        - MPV Structural Specification

Artifact: MPV Organizational Specification

Medee Work Product Variability of MOISE+ Organizational Specification

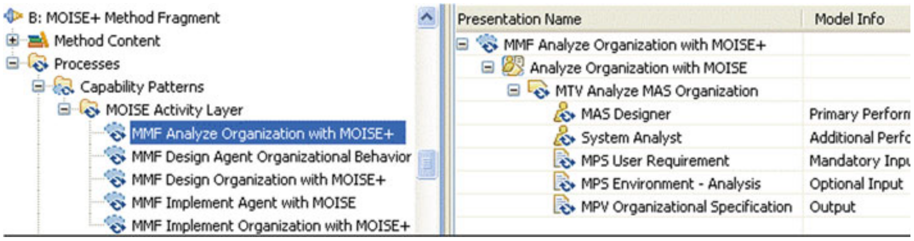
Expand All Sections Collap

Purpose

Relationships

Fulfilled Slots	• MPS Organization - Analysis • MPS Organization - Design
Container Artifact	• MFE MOISE+ Organization
Contained Artifacts	• MPV Deontic Specification • MPV Functional Specification • MPV Structural Specification

Fig. 15 MAS work product variability for MOISE+, detailing the MPV Organizational Specification



- B: MOISE+ Method Fragment
  - Method Content
  - Processes
    - Capability Patterns
      - MOISE Activity Layer
        - MMF Analyze Organization with MOISE+
        - MMF Design Agent Organizational Behavior
        - MMF Design Organization with MOISE+
        - MMF Implement Agent with MOISE
        - MMF Implement Organization with MOISE+

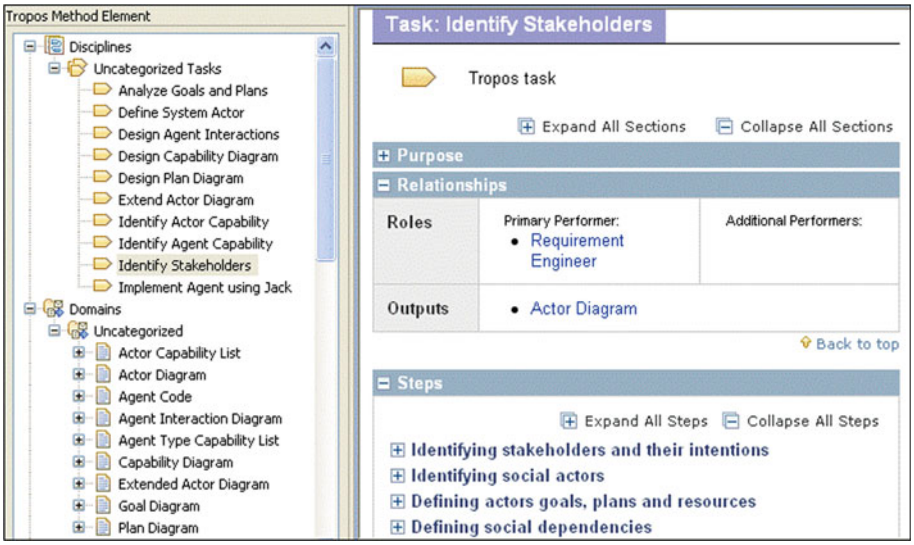
Presentation Name	Model Info
MMF Analyze Organization with MOISE+	
Analyze Organization with MOISE	
MTV Analyze MAS Organization	
MAS Designer	Primary Perforn
System Analyst	Additional Perfc
MPS User Requirement	Mandatory Inpu
MPS Environment - Analysis	Optional Input
MPV Organizational Specification	Output

Fig. 16 MAS Activity Method fragments sourced from MOISE+, detailing the MMF Analyze Organization with MOISE+

Secondly, Medee Fragment pillar was populated with five MAS Activity Method Fragments built upon these extended method elements: MMF Analyze Organization with MOISE+, MMF Design Agent Organizational Behavior with MOISE+, MMF Design Organization with MOISE+, MMF Implement Agent with MOISE+, and MMF Implement Organization with MOISE+ (Fig. 16, left side).

Figure 16 (right side) presents the MMF Analyze Organization with MOISE+ in detail. It consists of a process patter nesting a quasi-homonym activity that holds the MTV Analyze MAS Organization with MOISE+. Moreover, it is performed by the MAS Designer and System Analyst (primary and additional roles) and generates the MPV Organizational Specification as output. Finally, it is important to highlight that this method fragment may dynamically receive as mandatory input a work product that fulfills the MAS User Requirement slots, such as the MPV Tropos Actor Diagram and MPV Tropos Goal Diagram. Moreover, it could dynamically receive a work product as optional input fulfilling the MAS Environment Component, as the one sourced from GAIA.

Finally, these five MAS activity method fragments have been categorized by the Medee MAS Semiotic Taxonomy. For instance, MMF Analyze Organization with MOISE+ has been categorized by the Organization Oriented MAS and Organization Component categories from the Pragmatic Level, and by the Activity Method Fragment and Analysis Discipline categories from the Semantic Level.



Tropos Method Element

- Disciplines
  - Uncategorized Tasks
    - Analyze Goals and Plans
    - Define System Actor
    - Design Agent Interactions
    - Design Capability Diagram
    - Design Plan Diagram
    - Extend Actor Diagram
    - Identify Actor Capability
    - Identify Agent Capability
    - Identify Stakeholders
    - Implement Agent using Jack
- Domains
  - Uncategorized
    - Actor Capability List
    - Actor Diagram
    - Agent Code
    - Agent Interaction Diagram
    - Agent Type Capability List
    - Capability Diagram
    - Extended Actor Diagram
    - Goal Diagram
    - Plan Diagram

Task: Identify Stakeholders

Tropos task

Expand All Sections Collapse All Sections

Purpose

Relationships

Roles	Primary Performer: • Requirement Engineer	Additional Performers:
Outputs	• Actor Diagram	

Back to top

Steps

Expand All Steps Collapse All Steps

- Identifying stakeholders and their intentions
- Identifying social actors
- Defining actors goals, plans and resources
- Defining social dependencies

Fig. 17 Tasks and work products captured from Tropos, detailing the Identify Stakeholders task

5.4.3 Medee elements captured from Tropos

Tropos [6,25–27] proposes a development method for MAS involving the following phases: *Early Requirements, Late Requirements, Architectural Design*, *Detailed Design* and *Implementation*. The goal of the first two phases is to provide a set of functional requirements, as well as non-functional requirements, for the system to be built, while *Architectural Design* and *Detailed Design* phases focus on the system specification. Finally, the *Implementation* phase transforms the results of the preceding phases using an agent development platform to code the MAS.

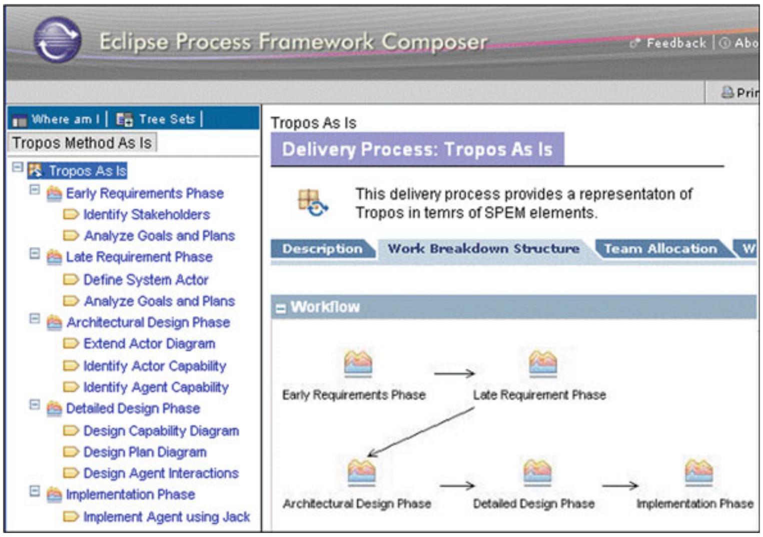
The Method Element Capture phase was performed to populate the Medee Element pillar with elements captured from Tropos (Fig. 17): ten tasks, nine work products, and several guidance, including guidelines, whitepapers, and concepts.

Moreover, Fig. 17 (right) depicts one of these tasks, the *Identify Stakeholder*, that represents the first piece of work proposed by Tropos. This task is performed by the *Requirements Engineer* (primary role) and aims to produce the *Actor Diagram* following four steps (lower right). Also, the Tropos As Is method was built upon these elements and published on HTML pages using the EPF Composer. Figure 18 illustrates a delivery process containing the five phases proposed by Tropos.

5.4.4 Medee fragments sourced from Tropos

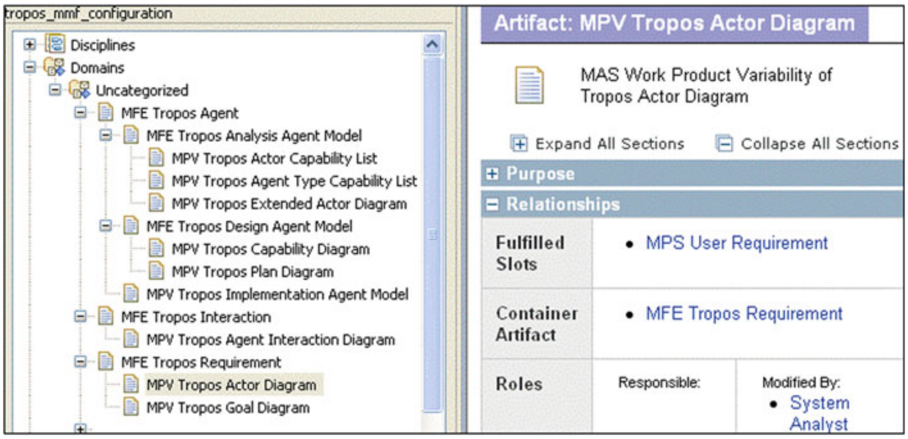
The Medee Fragment pillar was populated with MAS method fragments built upon method elements captured from Tropos according to activities proposed by the Method Fragment Elaboration phase—Create Activity Method Fragment, Create Intermediate Method Fragment, and Create Process Method Fragment.

Before creating MAS activity method fragments, the nine work products and ten tasks captured from Tropos were extended, as illustrated in Fig. 19 (right). For instance, the MPV



This delivery process provides a representaton of Tropos in temrs of SPEM elements.

Fig. 18 Tropos As Is published as HTML pages


tropos_mmf_configuration
Disciplines
Domains
Uncategorized
MFE Tropos Agent
MFE Tropos Analysis Agent Model
MPV Tropos Actor Capability List
MPV Tropos Agent Type Capability List
MPV Tropos Extended Actor Diagram
MFE Tropos Design Agent Model
MPV Tropos Capability Diagram
MPV Tropos Plan Diagram
MPV Tropos Implementation Agent Model
MFE Tropos Interaction
MPV Tropos Agent Interaction Diagram
MFE Tropos Requirement
MPV Tropos Actor Diagram
MPV Tropos Goal Diagram
Artifact: MPV Tropos Actor Diagram
Expand All Sections
Collapse All Sections
Purpose
Relationships

MAS Work Product Variability of Tropos Actor Diagram

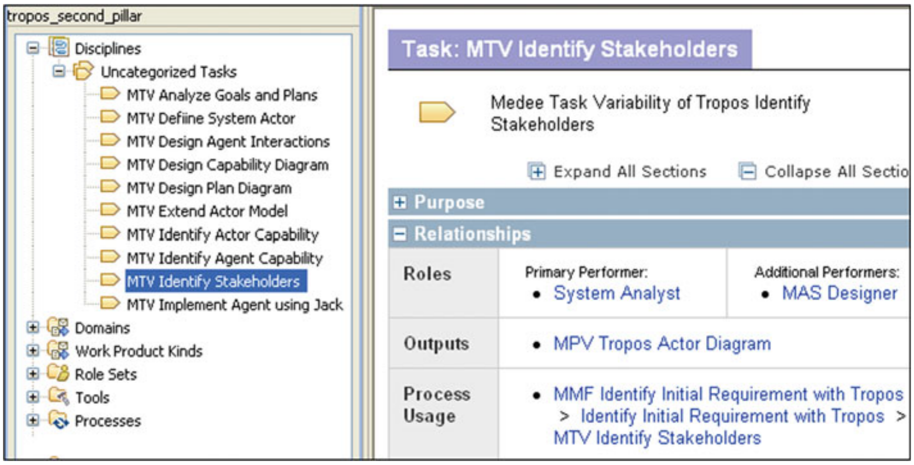
Fulfilled Slots	• MPS User Requirement	
Container Artifact	• MFE Tropos Requirement	
Roles	Responsible:	Modified By: • System Analyst

Fig. 19 MAS Work Product Variability for Tropos, detailing the MPV Tropos Actor Diagram

Tropos Actor Diagram extends the *Actor Diagram*, by specifying the fulfillment of the MPS User Requirements slot, and encapsulation by the MFE Tropos Requirements.

Figure 20 (right) details the MAS Task Variability called MTV Identify Stakeholders, by showing its primary and additional performers (System Analyst replacing the *Requirements Engineer* role), as well as its output work product (MPV Tropos Actor Diagram).

Hence, eight Medee fragments pertaining to the activity layer were built, as depicted in Fig. 21 (left). Furthermore, this figure (right) offers a detailed view of one of these fragments, the MMF Identify Initial Requirements with Tropos: it consists of a process pattern nesting a quasi-homonym activity for holding the MTV Identify Stakeholder; it is performed by the System Analyst and MAS Designer (primary and additional roles); and the MPV Tropos Actor Diagram as output. Nonetheless, this fragment does not require any input work product.


tropos_second_pillar

- Disciplines
  - Uncategorized Tasks
    - MTV Analyze Goals and Plans
    - MTV Define System Actor
    - MTV Design Agent Interactions
    - MTV Design Capability Diagram
    - MTV Design Plan Diagram
    - MTV Extend Actor Model
    - MTV Identify Actor Capability
    - MTV Identify Agent Capability
    - MTV Identify Stakeholders
    - MTV Implement Agent using Jack
- Domains
- Work Product Kinds
- Role Sets
- Tools
- Processes

Task: MTV Identify Stakeholders

Medee Task Variability of Tropos Identify Stakeholders

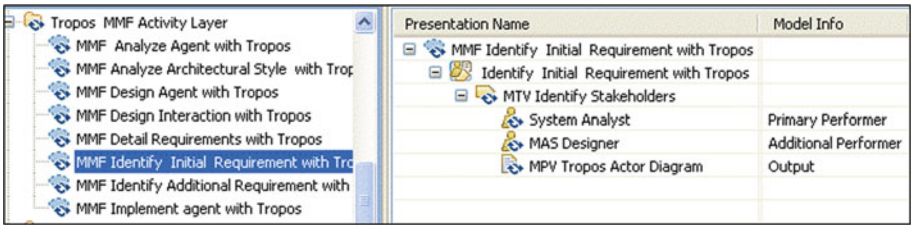
Expand All Sections Collapse All Sectio

Purpose

Relationships

Roles	Primary Performer: • System Analyst	Additional Performers: • MAS Designer
Outputs	• MPV Tropos Actor Diagram	
Process Usage	• MMF Identify Initial Requirement with Tropos > Identify Initial Requirement with Tropos > MTV Identify Stakeholders	

Fig. 20 MAS Task Variability for Tropos, detailing MTV Identify Stakeholders task


- Tropos MMF Activity Layer
  - MMF Analyze Agent with Tropos
  - MMF Analyze Architectural Style with Trop
  - MMF Design Agent with Tropos
  - MMF Design Interaction with Tropos
  - MMF Detail Requirements with Tropos
  - MMF Identify Initial Requirement with Tro
  - MMF Identify Additional Requirement with
  - MMF Implement agent with Tropos

Presentation Name	Model Info
MMF Identify Initial Requirement with Tropos	
Identify Initial Requirement with Tropos	
MTV Identify Stakeholders	
System Analyst	Primary Performer
MAS Designer	Additional Performer
MPV Tropos Actor Diagram	Output

Fig. 21 MAS Activity Method fragments sourced from Tropos, detailing the MMF Identify Initial Requirements with Tropos

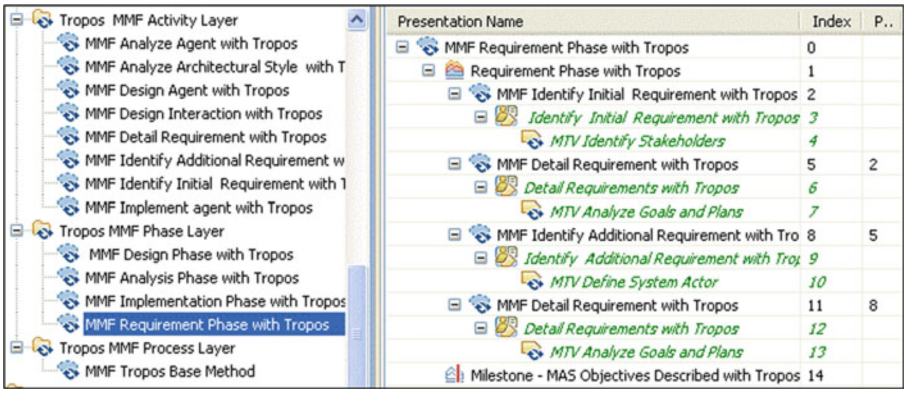
Moreover, these eight MAS activity method fragments were categorized using the Medee MAS Semiotic Taxonomy. For instance, the MMF Identify Initial Requirements with Tropos was categorized as follows: (i) Goal based style category from Pragmatic Level; (ii) Activity Method Fragment, Require-ments Discipline, and Tropos Method categories from Semantic Level; (iii) Graphical Notation category from Syntactic Level.

Since Tropos does not propose any iteration, the elaboration of intermediate method fragments concerned the creation of four MAS phase method fragments, as illustrated in Fig. 22 (left): MMF Requirements Phase with Tropos, MMF Analysis Phase with Tropos, MMF Design Phase with Tropos, and MMF Implementation Phase with Tropos and one milestone.

As detailed in Fig. 22 (right), the MMF Requirements Phase with Tropos embodies three fragments: MMF Identify Requirements with Tropos, MMF Detail Requirements with Tropos (occurring twice), and MMF Identify Additional Requirements with Tropos.

The MAS process method fragment sourced from Tropos, called the MMF Tropos Base Method, encompasses the four MAS phase method fragments aforementioned, as well as the Tropos Base Method milestone. Indeed, Fig. 23 depicts the way in which MAS method fragments sourced from Tropos are put together to form the MAS Tropos Base Method.

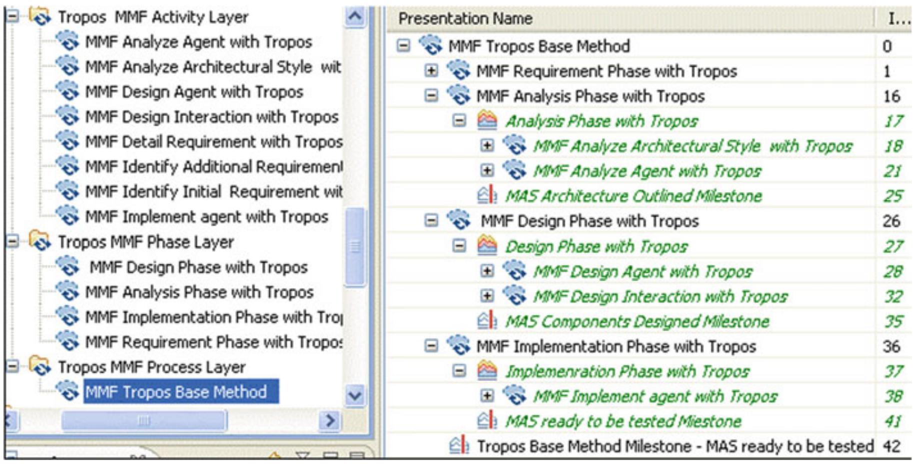
Thus, the MMF Tropos Base Method is ready for use during a situational composition, as described in Sect. 6.4. However, before taking part in a situational composition, the MMF Tropos Base Method was classified into several categories of the Medee MAS Semiotic taxonomy. For instance, considering the Social level of this taxonomy, it was categorized as follows:



```
Tropos MMF Activity Layer
    MMF Analyze Agent with Tropos
    MMF Analyze Architectural Style with T
    MMF Design Agent with Tropos
    MMF Design Interaction with Tropos
    MMF Detail Requirement with Tropos
    MMF Identify Additional Requirement w
    MMF Identify Initial Requirement with T
    MMF Implement agent with Tropos
Tropos MMF Phase Layer
    MMF Design Phase with Tropos
    MMF Analysis Phase with Tropos
    MMF Implementation Phase with Tropos
    MMF Requirement Phase with Tropos
Tropos MMF Process Layer
    MMF Tropos Base Method
```

Presentation Name	Index	P..
MMF Requirement Phase with Tropos	0	
Requirement Phase with Tropos	1	
MMF Identify Initial Requirement with Tropos	2	
Identify Initial Requirement with Tropos	*3*	
MTV Identify Stakeholders	*4*	
MMF Detail Requirement with Tropos	5	2
Detail Requirements with Tropos	*6*	
MTV Analyze Goals and Plans	*7*	
MMF Identify Additional Requirement with Tro	8	5
Identify Additional Requirement with Tro	*9*	
MTV Define System Actor	*10*	
MMF Detail Requirement with Tropos	11	8
Detail Requirements with Tropos	*12*	
MTV Analyze Goals and Plans	*13*	
Milestone - MAS Objectives Described with Tropos	14	

Fig. 22 MAS Phase Method fragments sourced from Tropos, detailing the MMF Requirements Phase with Tropos



```
Tropos MMF Activity Layer
    MMF Analyze Agent with Tropos
    MMF Analyze Architectural Style wit
    MMF Design Agent with Tropos
    MMF Design Interaction with Tropos
    MMF Detail Requirement with Tropos
    MMF Identify Additional Requirement
    MMF Identify Initial Requirement wit
    MMF Implement agent with Tropos
Tropos MMF Phase Layer
    MMF Design Phase with Tropos
    MMF Analysis Phase with Tropos
    MMF Implementation Phase with Tro
    MMF Requirement Phase with Tropos
Tropos MMF Process Layer
    MMF Tropos Base Method
```

Presentation Name	I...
MMF Tropos Base Method	0
MMF Requirement Phase with Tropos	1
MMF Analysis Phase with Tropos	16
Analysis Phase with Tropos	*17*
MMF Analyze Architectural Style with Tropos	*18*
MMF Analyze Agent with Tropos	*21*
MAS Architecture Outlined Milestone	*25*
MMF Design Phase with Tropos	26
Design Phase with Tropos	*27*
MMF Design Agent with Tropos	*28*
MMF Design Interaction with Tropos	*32*
MAS Components Designed Milestone	*35*
MMF Implementation Phase with Tropos	36
Implemenration Phase with Tropos	*37*
MMF Implement agent with Tropos	*38*
MAS ready to be tested Milestone	*41*
Tropos Base Method Milestone - MAS ready to be tested	42

Fig. 23 MAS Method Fragments sourced from Tropos, organized in the MMF Tropos Base Method

(i) High Utilization Degree Category, since it is well known and used; (ii) Low Reutilization Degree Category, as it does not involve MAS component reuse; (iii) Low Validation Degree Category, since it does not provide validation and verification activities; and (iv) Low Iteration Degree Category, as Tropos work breakdown structure does not deal with iterations.

Having given examples of Medee elements and fragments stored in the Medee Repository, we are ready to describe the composition of a Medee MAS situational method, which is made in the sequence.

6 Composing a MAS Situational Method using Medee

In order to validate the Medee method framework, we built MAS situational methods for a project that involved students in a graduate course in charge of developing a MAS for the Multiagent Contest.[7] The remainder of this section describes this case study in detail.

[7] http://multiagentcontest.org/2010.

6.1 Case study overview

The goal of the Multiagent Contest is to develop a MAS to solve a cooperative task in a highly dynamic environment. The environment of the MAS is a grid-like world in which animals (e.g. *cows*) are moving around collectively in one or more groups showing swarm like behavior. There are two corrals, each one belonging to an agent team (e.g. *herder team*), which competed to lead cows to their own corral. The winning herder team is the one that has a higher number of cows in its corrals.

This kind of application clearly fits the requirements we had in mind as a motivation for creating the Medee method framework: it possibly needs combining AOSE methods and AO models to produce a good solution for the problem. Particularly, students followed a situational method to build a MAS application, i.e., their own team. The experiment involved another MAS application, the *control herder team,* which competed with those applications built by the students. This former was a simplified version of a team that we have built to compete in previous editions of the Agent Contest [28].

Two situational methods were composed using Medee fragments: the Tropos–MOISE+ situational method and the Gaia-MOISE+ situational method. Due to space limitations, only the Tropos–MOISE+ situational method will have its composition described here. Both methods are published at the Medee website.

The situational method composition was performed following the activities proposed in the Medee Method Composition phase presented in Sect. 5.2—Characterize MAS project situation, Select MAS method fragments, Compose MAS situational method, and Publish MAS situation method—as shown next.

6.2 Characterizing the MAS project situation

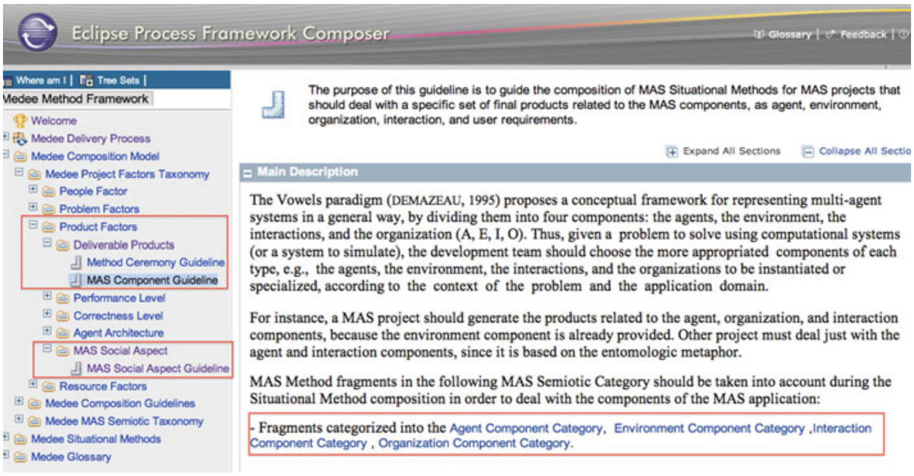
The current project situation was characterized through the factors provided by the Medee Project Factors Taxonomy: people, problem, product, and resource. The project situation characterization can be summarized as follows.

The people involved in the project (people factors) consist of small development teams encompassing two or three students with little prior experience in the application domain, in MAS development approaches, and in UML (including use cases).

The problem to be tackled (problem factors) involves agent cooperation and coordination. Furthermore, the problem is quite well defined, is not susceptible to change during project development, and can be solved using closed MAS in a dynamic environment.

Concerning the product related factors, the project aims to deliver (deliverable product factor) two final products: a scientific paper describing and discussing the several development phases of the MAS application, and the execution code of the MAS application using a BDI agent platform (Agent Architecture factor). Additionally, in order to deal with the cooperation and coordination among agents, the MAS application should adopt an organization-centered MAS approach (MAS social aspect factor).

Finally, the project teams had to manage the following resource factors to deliver such final products: a short project deadline (ten weeks) and a low budget that roughly corresponds to 160 working hours. Nonetheless, the project could mitigate this restriction of time and effort using the available reusable assets for leveraging the MAS application development, mainly the J-MOISE+ tool [35].



The purpose of this guideline is to guide the composition of MAS Situational Methods for MAS projects that should deal with a specific set of final products related to the MAS components, as agent, environment, organization, interaction, and user requirements.

The Vowels paradigm (DEMAZEAU, 1995) proposes a conceptual framework for representing multi-agent systems in a general way, by dividing them into four components: the agents, the environment, the interactions, and the organization (A, E, I, O). Thus, given a problem to solve using computational systems (or a system to simulate), the development team should choose the more appropriated components of each type, e.g., the agents, the environment, the interactions, and the organizations to be instantiated or specialized, according to the context of the problem and the application domain.

For instance, a MAS project should generate the products related to the agent, organization, and interaction components, because the environment component is already provided. Other project must deal just with the agent and interaction components, since it is based on the entomologic metaphor.

MAS Method fragments in the following MAS Semiotic Category should be taken into account during the Situational Method composition in order to deal with the components of the MAS application:

- Fragments categorized into the Agent Component Category, Environment Component Category ,Interaction Component Category , Organization Component Category.

Fig. 24 Assisting the Method Engineer to generate the method fragments preliminary list through the Medee Composition Model built upon EPF Composer

6.3 Selecting method fragments for the current situation

Based on the Medee Composition Model, this activity aimed to select those method fragments—among the 64 fragments stored in the Medee Method Repository—that better suit for the project situation already assessed in terms of project factors. It encompassed two tasks—Select Candidate MAS Method Fragment and Analyze MAS Base Method.

The first task involved the selection of fragment candidates among all fragment layers in order to produce a Method Fragment Preliminary List. Thus, this task consisted of analyzing those Medee composition guidelines indicated by the project factors, as well as identifying and then inspecting the semiotic categories indicated by them, and finally selecting the candidate method fragments, as explained in Sect. 5.2.

For instance, the Deliverable product factor (scientific paper and running code) led to two guidelines, MAS Component and Method Ceremony, while the MAS social aspect factor led to the quasi-homonym guideline, as depicted in Fig. 24 (left). Moreover, Fig. 24 (lower right frame) shows that the MAS Component guideline points out some semiotic categories that deal with this product factor, among them Agent Component and Organization Component categories. Then, the method engineer only had to click over these suggested categories of the Medee MAS Semiotic Taxonomy to inspect them, i.e., to see the list of available fragments.

The Method Fragment Preliminary List contained the following method fragments, among others: (i) MAS method fragments sourced from MOISE+, for analyzing, designing, and implementing organizations, as well as for implementing agents; (ii) MAS method fragments sourced from Tropos, mainly the MMF Tropos Base Method; (iii) MAS method fragments sourced from Gaia, involving the analysis and design of MAS components such as agents and interactions.

The last task involved the analysis of such a list to verify whether it was possible to choose a MAS base method among the MAS process method fragments included in it. Therefore, the MMF Tropos Base Method had been adopted since it is suitable for dealing with several project factors, among them: it encompasses the required development phases, including requirements; it offers two of the three required MAS components, i.e., agents and interactions. Furthermore, a top–down situational method composition could add to the MMF Tropos Base

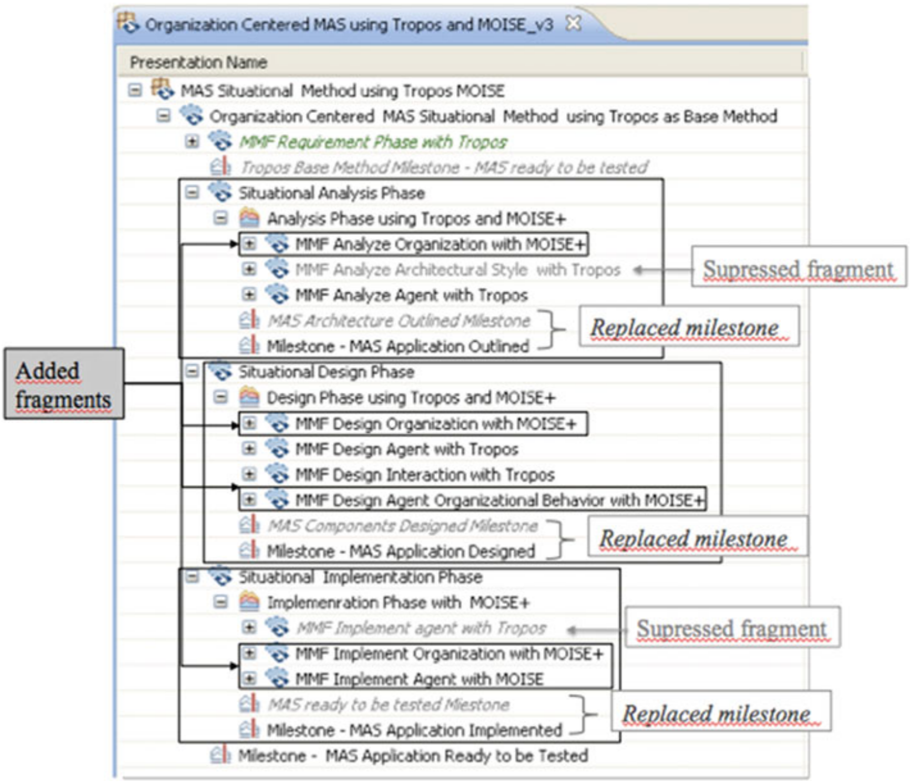

```
Organization Centered MAS using Tropos and MOISE_v3
Presentation Name
⊟ MAS Situational Method using Tropos MOISE
  ⊟ Organization Centered MAS Situational Method using Tropos as Base Method
    ⊞ MMF Requirement Phase with Tropos
      Tropos Base Method Milestone - MAS ready to be tested
    ⊟ Situational Analysis Phase
      ⊟ Analysis Phase using Tropos and MOISE+
        ⊞ MMF Analyze Organization with MOISE+
        ⊞ MMF Analyze Architectural Style with Tropos
        ⊞ MMF Analyze Agent with Tropos
          MAS Architecture Outlined Milestone
          Milestone - MAS Application Outlined
    ⊟ Situational Design Phase
      ⊟ Design Phase using Tropos and MOISE+
        ⊞ MMF Design Organization with MOISE+
        ⊞ MMF Design Agent with Tropos
        ⊞ MMF Design Interaction with Tropos
        ⊞ MMF Design Agent Organizational Behavior with MOISE+
          MAS Components Designed Milestone
          Milestone - MAS Application Designed
    ⊟ Situational Implementation Phase
      ⊟ Implemenration Phase with MOISE+
        ⊞ MMF Implement agent with Tropos
        ⊞ MMF Implement Organization with MOISE+
        ⊞ MMF Implement Agent with MOISE
          MAS ready to be tested Milestone
          Milestone - MAS Application Implemented
    Milestone - MAS Application Ready to be Tested
```

Fig. 25 Top–down situational composition, detailing Situational phases using Tropos and MOISE+

Method those fragments sourced from MOISE+ in order to cover the Organization component, as shown next.

6.4 Composing the Tropos–MOISE+ situational method

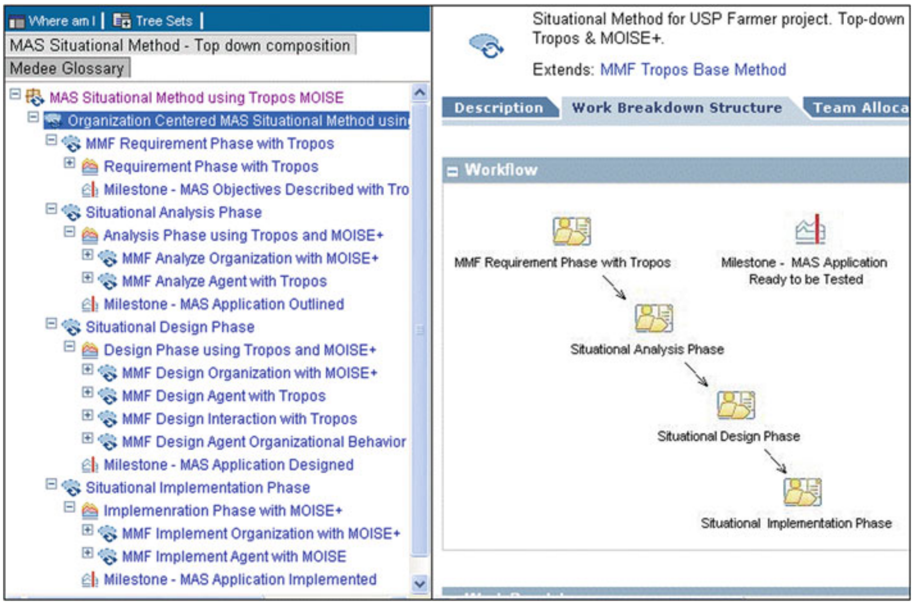
Composing the Tropos–MOISE+ situational method using a top–down approach consisted of tailoring the MMF Tropos Base Method by adding MAS method fragments sourced from MOISE+ and suppressing those fragments that were no longer needed in the situational context.

Thus, as depicted in Fig. 25 and explained in the next paragraphs, the five MAS activity method fragments sourced from MOISE+ were added to the MMF Tropos Base Method, while some fragments had been suppressed from it.

Moreover, Fig. 25 illustrates how the EPF Composer counts on functionalities for tailoring methods, i.e. tailoring delivery processes, such as suppressing process patterns (represented in gray), keeping some of them unchanged (represented in green) and adding new ones.

As explained in Sect. 5.4, the MMF Tropos Base Method embodies four phases: MMF Requirements Phase with Tropos, MMF Analysis Phase with Tropos, MMF Design Phase with Tropos, and MMF Implementation Phase with Tropos.

The MMF Requirements Phase with Tropos was kept in the situational method without any modification, since it proposes a goal-based approach for dealing with requirements that is


Where am I | Tree Sets |
MAS Situational Method - Top down composition
Medee Glossary

- MAS Situational Method using Tropos MOISE
  - Organization Centered MAS Situational Method usin
    - MMF Requirement Phase with Tropos
      - Requirement Phase with Tropos
      - Milestone - MAS Objectives Described with Tro
    - Situational Analysis Phase
      - Analysis Phase using Tropos and MOISE+
        - MMF Analyze Organization with MOISE+
        - MMF Analyze Agent with Tropos
      - Milestone - MAS Application Outlined
    - Situational Design Phase
      - Design Phase using Tropos and MOISE+
        - MMF Design Organization with MOISE+
        - MMF Design Agent with Tropos
        - MMF Design Interaction with Tropos
        - MMF Design Agent Organizational Behavior
      - Milestone - MAS Application Designed
    - Situational Implementation Phase
      - Implemenration Phase with MOISE+
        - MMF Implement Organization with MOISE+
        - MMF Implement Agent with MOISE
      - Milestone - MAS Application Implemented

Situational Method for USP Farmer project. Top-down Tropos & MOISE+.

Extends: MMF Tropos Base Method

Description | Work Breakdown Structure | Team Alloca

Workflow

MMF Requirement Phase with Tropos
Milestone - MAS Application Ready to be Tested
Situational Analysis Phase
Situational Design Phase
Situational Implementation Phase

Fig. 26 Tropos–MOISE+ situational method published as HTML pages

appropriate to teams that are not familiar with the UML. The remaining phases were tailored to allow the development of the organization-centered MAS application using MOISE+.

Thus, Fig. 25 depicts that the MMF Analyze Organization with MOISE+ was added to the Situational Analysis phase, while the MMF Analyze Architectural Style with Tropos was suppressed, since the MOISE+ Organizational Specification already encompasses a MAS architectural style.

Moreover, given that the new Situational Analysis phase generates work products sourced from Tropos and MOISE+, its milestone was replaced by a new one, which embodies MOISE+ specification and Tropos diagrams as required results.

Furthermore, the Situational Design phase had two fragments sourced from MOISE+ added to the original ones from MMF Design Phase with Tropos (for designing agents and interactions with Tropos): MMF Design Organization with MOISE+ and MMF Design Agent Organizational Behavior with MOISE+. Consequently, the previous milestone was conveniently replaced by a suitable one.

Finally, the Situational Implementation phase only encompassed the two fragments sourced from MOISE+ for implementing organizations and agents, MMF Implement Organization with MOISE+ and MMF Implement Agent with MOISE+, since the fragment sourced from Tropos related to agent implementation was suppressed. Consequently, a new milestone that could represent the situational required work products replaced the previous one.

6.5 Publishing the Tropos–MOISE+ situational method

The Tropos–MOISE+ situational method was published as a fully hyperlinked collection of HTML pages, as illustrated in Fig. 26.

Moreover, Fig. 26 (upper left) depicts that such HTML pages also contain the Medee Glossary. The glossary was published in conjunction with the MAS situational method and it consists of a collection of term definitions that aims to facilitate the understanding of the concepts used to define MAS method fragments and situational methods.

As mentioned before, the Tropos–MOISE+ Situational Method, as well as the Gaia-MOISE+ Situational Method, was used by our students to develop a MAS for the Multiagent Contest. Due to space limitations, it is out of the scope of this paper to describe the MAS that were generated, interested readers may refer to [8].

Having explained how to use the Medee method framework for composing situational methods, the next section discusses the main aspects of our approach.

7 Discussion

As described in the course of this paper, the components of the Medee method framework—the Conceptual Model, the Composition Model, the Method Repository, and the Delivery Process—cover most of a typical situational method procedure: from managing the method repository to building the situational method.

Furthermore, our approach is based on several aspects that constitute advancements in the way of dealing with situational methods for AOSE. Firstly, it is fully compliant with SPEM. Secondly, it offers fragments sourced from both AOSE methods and AO models. Thirdly, it is independent of a given MAS meta-model. Finally, it offers a structured way for dealing with AOSE special aspects. The remainder of this section presents a brief discussion about these aspects, taking into account related work described in Sect. 2.4.

7.1 SPEM compliance

As previously explained, not only the Medee Conceptual Model and Method Repository are represented in terms of SPEM elements—since we introduce neither new elements nor new associations to define method fragment and situational methods—but also the Medee Delivery Process and the Medee Composition Model are specified using SPEM. As explained in Sect. 2.4, such a full compliance with SPEM is not offered by OPF [32], which is not based on SPEM, neither by the other approaches based on the process fragment definition [49–51], since they involve elements not provided by SPEM, like those pertaining to a MAS meta-model.

In our opinion, the full compliance with SPEM offers important benefits and represents advancement in the current state of the art in AOSE. Firstly, SPEM is the de facto standard for method meta-modeling in Software Engineering as a whole and in AOSE field in particular. Thus, on one hand, we can leverage SPEM skills available in the software engineering community in order to build situational methods for AOSE and, on the other hand, we may accelerate the dissemination of AOSE concepts in the software industry, by using this standard for representing development methods.

Secondly, SPEM covers more than modeling and documenting methods aspects, since it also offers elements for managing and enacting methods and fragments, providing sophisticated mechanisms for reusability, modularization, extension, packaging and deployment of methods, as explained in Sect. 2.2. Therefore, being SPEM compliant allows the use of SPEM-based tools, such as the EPF Composer. This has the advantage of supporting several procedural steps for building situational methods, from the MAS method repository management to the MAS project execution, since the project team can use the MAS situational method deployed to website during the project, as illustrated in this paper. Finally, Medee method framework users skilled in EPF Composer can easily manage, configure, publish, and distribute AOSE situational methods through websites.

7.2 Putting together AOSE methods and AO models

The Medee method framework provides fragments from both AO models and AOSE methods. As explained in Sect. 2.4, some approaches are concerned with the extension of a particular AOSE method to incorporate organizational concept, like O-MaSE and ASPECS. However, they do not offer a whole situational environment for dealing with fragments sourced from several AOSE methods and AO models, nor for composing situational methods out of those fragments.

Therefore, putting together method fragments sourced from several AOSE methods and AO models—by storing them in the same method repository and composing situational methods out of them—consists of an innovative approach in the AOSE field.

7.3 Independency of any MAS meta-model

The Medee method framework does not depend on the availability of a common meta-model for representing concepts used by both AOSE methods and AO models.

As explained in Sect. 2.4, other approaches [16,49–51] are strongly dependent on a given MAS meta-model. Thus, typical steps for building situational methods, such fragment elaboration, fragment selection, and situational method building, are based on such a meta-model. Unfortunately, the AOSE community has not yet achieved a consensus about the main MAS concepts [18]. In fact, most of the MAS development approaches provide their own meta-model. Thus, the Medee Method Framework offers an important benefit: it allows the AOSE community to take advantage of Situational Method Engineering principles for building MAS situational methods out of fragments sourced from several AOSE methods and AO models even though a consensus concerning a common MAS meta-model was not yet (if ever) achieved.

7.4 Adopting an integrated way to deal with AOSE particularities during the situational composition

AOSE development approaches involve particular aspects, such as specific system architecture, development platform, design and programming languages. The Medee method framework takes these aspects into account in an integrated way, from the method repository management to the situational composition. Firstly, the Medee user can standardize method fragments by using common MAS development roles—like MAS Developer and MAS Tester—as well as Medee semiotic categories, as such MAS component (e.g. agent, environment, organization), MAS nature (e.g. open/closed), design language (e.g. UML, AUML), and programming language (e.g. Java, AgentSpeak).

Secondly, the Medee user can clearly state the project characteristics—in terms of people, problem, product, and resource factors—taking into account AOSE aspects. Possible examples are: the project team has no previous experience with developing MAS although having some skills related to object-oriented methods and UML; the product to be delivered involves an organization-centered approach, the agent architecture is based on BDI.

Thirdly, both MAS project factors and method fragment categories are integrated into the Medee Composition Model, while the way how to proceed for characterizing the project, selecting fragments and putting them together in a situational method is described in the Medee Delivery Process.

Summing up, to the best of our knowledge, in the AOSE field there is no currently such a broad and integrated approach for situational composition, totally SPEM compliant and built upon an open source tool, the EPF Composer.

8 Conclusions and future work

We presented a situational approach for the development of MAS projects, the Medee method framework. Such an approach encompasses new definitions for MAS Method Fragment and MAS Situational Method, a model for leveraging project situation strengths during the situational composition, as well as a method repository and a process that describes in detail how to elaborate method fragments and build methods on demand according to MAS project characteristics.

The Medee method framework is full SPEM compliant and built upon an open source tool, the EPF Composer. Currently, it contains more than two hundred elements, including 64 (sixty four) Medee method fragments sourced from Gaia, Tropos, Ingenias, PASSI, MOISE+, and OperA, as well as the representation of these four AOSE methods in terms of SPEM elements. It is worth noting that the Medee Method Repository may be extended with method fragments sourced from other AOSE methods and AO models. The step-by-step work necessary to perform such an extension is described in the Medee Delivery Process, presented in Sect. 5.2.

Finally, we illustrated how such an approach could be used to compose a situational method for developing a MAS project using method fragments sourced from Tropos and MOISE+.

Our future work is concerned with three main aspects. Firstly, we want to extend the Medee Method Repository in order to offer method fragments for testing MAS applications. However, the most popular AOSE methods do not provide activities related to MAS testing. Thus, this method fragments could be sourced from research dealing especially with the testing discipline. Secondly, we are currently developing a tool to facilitate and speed up the selection of MAS method fragments. Finally, we intent to enhance the Medee Composition Model to deal with MAS project success performance indicators, since the achievement of some success performance indicators can be limited by project factors [30]. Thus, it is important to identify those project factors that can negatively impact the desired success indicators and try to mitigate their effect over project success, using the appropriate method fragments.

Acknowledgments This research has been mostly supported by FAPESP[8], Brazil. Jaime Sichman is partially supported by CNPq, Brazil. Sara Casare was partially supported by CAPES, Brazil. Zahia Guessoum is partially supported by CNRS, France. We would like to thank the invaluable work from the reviewers that helped us to improve the paper quality and readability.

References

1. Agerfalk, P. J., Brinkkemper, S., Gonzalez-Perez, C., Henderson-Sellers, B., Karlsson, F., Kelly, S., et al. (2007). Modularization constructs in method engineering: Towards common ground? In J. Ralyté, S. Brinkkemper, & B. Henderson-Sellers (Eds.), *IFIP situational method engineering: Fundamentals and experiences* (pp. 359–368). Boston: Springer.
2. Basili, V. R. (1993). The experience factory and its relationship to other improvement paradigms. In I Sommerville & P. Manfred (Eds.), *Fourth European Conference on Software Engineering ESEC 93* (pp. 68–83). doi:10.1007/3-540-57209-0_6.

[8] Regular research project 2009/10121-4 and international cooperation FAPESP-CNRS project 2009/54773-5.

3. Bernon, C., Camps, V., Gleizes, M. P., & Picard, G. (2005). Engineering adaptive multiagent systems: The ADELFE methodology. In B. Henderson-Sellers & P. Giorgini (Eds.), *Agent oriented methodologies* (pp. 172–202). London: Idea Group Publishing.
4. Bordini, R., & Hubner, J. (2007). *Jason : A Java-based interpreter for an extended version of AgentSpeak*. Retrieved February 25, 2013 from http://jason.sourceforge.net/Jason.pdf.
5. Bordini, R. H., Hubner, J. F., & Wooldridge, M. (2007). *Programming multiagent systems in agentspeak using Jason*. New York: Wiley.
6. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., & Perini, A. (2004). Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multiagent Systems*, *8*(3), 203–236.
7. Brinkkemper, S. (1996). Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, *38*(4), 275–280.
8. Casare, S. J. (2012). *Medee: A method framework for multiagent systems*. Ph.D. Thesis, Universidade de São Paulo, São Paulo, Brazil. Retrieved February 22, 2013 from http://www.teses.usp.br/teses/disponiveis/3/3141/tde-05032012-162517/en.php.
9. Casare, S. J., Brandão, A. A. F., & Sichman, J. S. (2010). A semiotic perspective for multiagent systems development. In *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)* (pp. 1373–1374). Toronto, Canada.
10. Casare, S. J., Brandão, A. A. F., & Sichman, J. S. (2010). A semiotic taxonomy to support multiagent systems situational development. In *1st Workshop on Autonomous Software Systems at CBSoft 2010, Salvador. AUTOSOFT 2010—I Workshop on Autonomous Software Systems—Anais CBSoft 2010*. SBC, 2010, Vol. 10 (pp. 31–40). Retrieved February 26, 2013 from http://www.autosoft.pcs.usp.br/images/cbsoft2010_autosoft_anais.pdf.
11. Casare, S. J., Guessoum, Z., Brandão, A. A. F., & Sichman, J. S. (2010). Towards a new approach for MAS situational method engineering: A fragment definition. In O. Boissier, A. E. F. Seghrouchni, S. Hassas, & N. Maudet (Eds.), *Proceedings of The Multiagent Logics, Languages, and Organizations Federated Workshops 2010*, Vol. 627 (pp 3–16). Lyon-France, Aachen: CEUR-WS. http://ceur-ws.org/Vol-627/fipa_1.pdf.
12. Casare, S. J., Guessoum, Z., Brandão, A. A. F. & Sichman, J. S. (2010). Devising situational method fragments for organization centered MAS development. In *Proceedings of the 8th European Workshop on Multi-Agent Systems*—EUMAS 2010. Paris, Vol. 1 (pp. 1–16).
13. Castelfranchi, C. (1995). Commitments: From individual intentions to groups and organizations. In T. Ishida (Ed.), *Proceedings of the First International Conference on Multiagent Systems (ICMAS 95)* (pp. 41–48).
14. Cohen, P., & Levesque, H. (1990). Intention is choice with commitment. *Artificial Intelligence*, *42*, 213–261.
15. Cossentino, M. (2005). From requirements to code with the PASSI methodology. In B. Henderson-Sellers & P. Giorgini (Eds.), *Agent oriented methodologies* (pp. 79–106). London: Idea Group Publishing.
16. Cossentino, M., Gaglio, S., Garro, A., & Seidita, V. (2007). Method fragments for agent design methodologies: From standardization to research. *International Journal on Agent Oriented Software Engineering (IJAOSE)*, *1*(1), 91–121.
17. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., & Koukam, A. (2010). ASPECS: An agent-oriented software process for engineering complex systems. *Autonomous Agents and Multiagent Systems*, *20*(2), 260–304.
18. DeLoach, S. (2009). Moving multiagent systems from research to practice. *International Journal of Agent-Oriented Software Engineering*, *3*(4), 378–382.
19. DeLoach, S. A., & García-Ojeda, J. C. (2010). O-MaSE: A customizable approach to designing and building complex, adaptive multiagent systems. *Intl Journal on Agent Oriented Software Engineering (IJAOSE)*, *4*(3), 244–280.
20. Demazeau, Y. (1995). From interactions to collective behavior in agent-based systems. In *Proceedings of the 1st European Conference on Cognitive Science* (pp. 117–132) Saint-Malo.
21. Demazeau, Y. (2010). Purposive multiagent systems. In *Proeedings of the International Conference on Agents and Artificial Intelligence, ICAART* (pp. 22–24), Vol. 1—Artificial Intelligence, Valencia, Spain, January.
22. Dignum, V. (2004). *A model for organizational interaction: Based on agents, founded in logic*. Ph.D. Thesis, Universiteit Utrecht, Utrecht, The Netherlands
23. Firesmith, D., & Henderson-Sellers, B. (2002). *The OPEN process framework—An introduction*. Harlow: Addison Wesley.
24. Garcia-Ojeda, J. C., Deloach, S. (2010). The O-MaSE process: A standard view. In *Proceedings of Workshop FIPA Design Process Documentation and Fragmentation Working Group (DPDF WG) at*

Multiagent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010) (pp. 55–66) Lyon, France.

25. Giorgini, P., et al. (2004). The Tropos methodology. In V. Bergenti, M. P. Gleizes, & F. Zambonelli (Eds.), *Methodologies and software engineering for agent systems* (pp. 89–106). Dordrecht, The Netherlands: Kluwer Academic Publishers.
26. Giorgini, P., et al. (2005). Tropos: A requirement-driven methodology for agent-oriented software. In B. Henderson-Sellers & P. Giorgini (Eds.), *Agent-oriented methodologies* (pp. 20–45). London: Idea Group Publishing.
27. Giorgini, P., et al. (2005). The Tropos meta-model and its use. *Informatica*, *29*, 401–408.
28. Gouveia, G. P., Pereira, R. H., & Sichman, J. (2011). The USP farmers herding team. *Annals of Mathematics and Artificial Intelligence*, *61*(4), 369–383.
29. Guessoum, Z., Cossentino, M., & Pavón, J. (2004). Roadmap of agent-oriented software engineering—The agentlink perspective. In V. Bergenti, M. P. Gleizes, & F. Zambonelli (Eds.), *Methodologies and software engineering for agent systems* (pp. 431–450). New York: Kluwer Academic Publishers.
30. Harmsen, A. F. (1997). *Situational method engineering*. Utrecht: Moret Ernst & young.
31. Haumer, P. (2007). *Eclipse process framework composer—Part 1—Key concepts*. Retrieved February 28, 2013 from http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf.
32. Henderson-Sellers, B. (2005). Creating a comprehensive agent-oriented methodology: Using method engineering and OPEN meta-model. In B. Henderson-Sellers & P. Giorgini (Eds.), *Agent-oriented methodologies* (pp. 368–397). Dordrecht, The Netherlands: Idea Group Publishing.
33. Hubner, J., Sichman, J., & Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in multiagent systems. In G. Bittencourt & G. L. Ramalho (Eds.), *16th Brazilian Symposium on AI, SBIA'02, LNAI 2507* (pp. 118–128). Berlin: Springer.
34. Hubner, J., Sichman, J., Boissier, O., et al. (2006). S-MOISE+: A middleware for developing organized multiagent systems. In O. Boissier (Ed.), *Coordination, organizations, institutions, and norms in multiagent systems, LNCS 3913* (pp. 64–78). Berlin: Springer.
35. Hubner, J. F., Sichman, J. S., & Boissier, O. (2007). Developing organised multiagent systems using the MOISE+ model: Programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, *1*(3), 370–395.
36. Hubner, J., Sichman, J. S., & Boissier, O. (2008). *MOISE+ tutorial*. Retrieved February 22, 2013 from http://moise.sourceforge.net/doc/tutorial.pdf.
37. IEEE-FIPA. (2012). *Design Process Documentation Template*, IEEE FIPA DPDF Working Group, #SC00097B, 2012–01-09. Retrieved February 25, 2013 from http://fipa.org/specs/fipa00097/SC00097B.pdf.
38. ISO 2007. ISO/IEC 24744:2007. (2007). Software engineering—Meta-model for development methodologies.
39. Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The unified software development process*. Reading, MA: Addison-Wesley.
40. Karlsson, F. (2005). Method configuration—Method and computerized tool support. Doctoral dissertation, Department of Computer and Information Science, Linkoping University.
41. Kruchten, P. (2003). *The rational unified process: An introduction* (3rd ed.). Reading, MA: Addison-Wesley.
42. Lemaıtre, C., & Excelente, C. B. (1998). Multiagent organization approach. In *2nd Iberoamerican Workshop on Distributed AI and MAS*, Toledo, Spain.
43. OMG. (2008). Object management group. Software & systems process engineering meta-model specification, version 2.0. OMG document number: formal/2008-04-01. Retrieved January 26, 2013 from http://www.omg.org/spec/SPEM/2.0/.
44. Pavón, J., Gómez-Sanz, J. J., & Fuentes, R. (2005). The INGENIAS methodology and tools. In B. Henderson-Sellers & P. Giorgini (Eds.), *Agent oriented methodologies* (pp. 236–276). London: Idea Group Publishing.
45. Picard, G., Hubner, J. F., Boissier, O., & Gleizes, M. -P. (2009). Réorganisation et auto-organisation dans les Système Multiagents. In Z. Guessoum & S. Hamas (Eds.), *Journées Francophones sur les systèmes multiagents— Génie Logiciel Multiagent (JFSMA09)* (pp. 89–97). France: Cepadués Editions.
46. Ralyté, J., Deneckére, R., & Rolland, C. (2003). Towards a generic model for situational method engineering. In J. Eder et al. (Eds.), *Proceedings of 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)* (pp. 95–110). Klagenfurt: Springer.
47. Rougemaille, S., Migeon, F., Millan, T., & Gleizes, M.-P. (2009). Methodology fragments definition in SPEM for designing adaptive methodology: A first step. In M. Luck, J. J. Gomez-Sanz (Eds.), *AOSE 2008, LNCS*, Vol. 5386 (pp. 74–85), Reading, MA: Springer.

48. Seidita, V., Cossentino, M., & Gaglio S. (2006). A repository of fragments for agent systems design. In *Proceedings of the Workshop on Objects and Agents (WOA 2006)* (pp. 130–137), CEUR.
49. Seidita, V., Cossentino, M., Hilaire, V., Gaud, N., Galland, S., Koukam, A., et al. (2010). The meta-model: A starting point for design processes construction. *International Journal of Software Engineering and Knowledge Engineering*, *20*(4), 575–608.
50. Seidita, V., Cossentino, M., & Chella, A. (2012). A proposal of process fragment definition and documentation. *Lecture Notes in Computer Science*, *7541*, 221–237.
51. Seidita, V., Cossentino, M., & Chella, A. (2012). *How to extract fragments from agent oriented design process, Workshop AOSE@AAMAS 2012*. Retrieved January 25, 2013 from http://www.pa.icar.cnr.it/cossentino/paper/aose0212-fragment.pdf.
52. Sichman, J. S., & Demazeau, Y. (2001). On social reasoning in multiagent systems. *Revista Iberoamericana de Inteligencia Artificial*, *13*, 68–84.
53. Sommerville, I. (2006). *Software engineering* (8th ed.). Reading, MA: Addison-Wesley.
54. Stamper, R. (1996). Signs, norms, and information system. In B. Holmqvist, P. B. Andersen, H. Klein, & R. Posner (Eds.), *Signs at work: Semiosis & information processing in organizations* (pp. 349–397). Berlin: Walter de Gruyter.
55. Wood, M., & DeLoach, S. A. (2001). An overview of the multiagent systems engineering methodology. In *Proceedings of the First International Workshop on Agent-Oriented Software Engineering* (pp. 207–221). LNCS, Vol. 1957, Berlin: Springer.
56. Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing multiagent systems: The Gaia methodology. *ACM Transaction on Software Engineering and Methodology*, *12*(3), 417–470.