Process Documentation Standardization: An Initial Evaluation

Massimo Cossentino[1], Alma Gómez-Rodríguez[2], Juan C. González-Moreno[2], Ambra Molesini[3], and Andrea Omicini[3]

[1] Istituto di Calcolo e Reti ad Alte Prestazioni
National Research Council, Palermo, Italy
cossentino@pa.icar.cnr.it
[2] Departamento de Informática, Universidade de Vigo, Ourense, Spain,
{alma,jcmoreno}@uvigo.es
[3] Alma Mater Studiorum – Università di Bologna, Italy
ambra.molesini@unibo.it, andrea.omicini@unibo.it

Abstract. The creation of new ad-hoc methodologies through the Situational Method Engineering approach needs the process fragments to be defined and available. Thus, it is necessary to previously define and extract such fragments from the global development process. So, it is important to provide the means of documenting the whole process from which fragments will be obtained. This paper presents an experimental evaluation of the methodologies documentation template proposed by the IEEE FIPA Design Process Documentation and Fragmentation working group. The template will be used for documenting three different agent-oriented methodologies in order to evaluate the template's strengths and weaknesses.

1 Introduction

Nowadays, in the software engineering field, there is a common agreement about the fact that there is not a unique methodology or process that fits all the application domains; this means that the methodology or process must be adapted to the particular characteristics of the domain for which the new software is developed. There are two major ways for adapting methodologies: tailoring (particularisation or customisation of a pre-existing processes) or Situational Method Engineering (SME) [1, 2]. In the last case the process is assembled from pre-existing components, called fragments, according to user's needs. This approach enhances reusability since a method component can be used several times.

The research on SME has become crucial in the Agent-Oriented Software Engineering (AOSE) since a variety of (special-purpose) agent-oriented (AO) methodologies have been defined in the past years [3–7] to discipline and support the multi-agent system (MAS) development. Each of the AO methodologies proposed up to now exhibits a specific metamodel, notation, and process. All of these features are fundamental for a correct understanding of a methodology, and should be suitably documented for supporting the creation of new ad-hoc

AO methodologies. In fact, the SME technique is strictly related to the documentation of the existing methodologies since the successfully construction of a new process is based on the correct integration of different fragments that should be well formalized. So, methodologies' documentation should be done in a standard way in order to facilitate the user's understanding, and the adoption of automatic tools able to interpret the fragment documentation.

In this context, the IEEE FIPA Design Process Documentation and Fragmentation (DPDF) working group [8] has recently proposed a template for documenting AO methodologies. This template takes into account the three aforementioned methodologies' features. In first place, it has been conceived without considering any particular process or methodology, and this should guarantee that all processes can be documented using the proposed template. Moreover, the template is also neutral regarding the MAS metamodel and/or the modelling notation adopted in describing the process. Secondly, the template has a simple structure resembling a tree. This implies that the documentation is built in a natural and progressive way, addressing the process general description and metamodel definition which constitute the root elements of the process itself. Then, the process phases are described as branches of the tree. Finally, thinner branches like activities or sub-activities can be documented. This means the template can support complex processes and very different situations. In third place, the use of the template is easy for any software engineer as it relies on very few previous assumptions. Moreover, the suggested notation is the OMG's standard Software Process Engineering Metamodel (SPEM) [9] with few extensions [10].

So, the goal of this paper is to present an experimental evaluation of the FIPA DPDF template by means of the application of such a template to three different AO methodologies: PASSI [11], INGENIAS [12], and SODA [13].

Accordingly, the remainder of the paper is organized as follows. Section 2 provides a brief description of the FIPA DPDF template, while Section 3 presents the application of the template to the three chosen AO methodologies. Section 4 presents some proposals for the improvement of the current version of the FIPA template, whereas Section 5 presents a discussion about the results obtained by the applicaiton of the template to the documentation of the three chosen methodologies. Finally, the conclusions of the whole work are reported in Section 6.

2 Process Documentation Template in a Nutshell

The IEEE FIPA DPDF working group has recently proposed a template for documenting AO methodologies. Here we report only a brief presentation of the template—interested readers can refer to [8] for the details of the template.

The template is based on the definition of process and process model as proposed by [14]. A process model is supposed to have three basic components: the stakeholders (i.e. roles and workers), the consumed and generated products (i.e. work products), and the activities and tasks undertaken during the process—

these being particular instances (i.e. work definitions) of the work to be done. Another important component of the template is the MAS metamodel, as previously considered in [10], because it is thought that the MAS metamodel may constrain the way in which fragments can be defined and reused.

1.Introduction
1.1.The (process name) Process lifecycle
1.2.The (process name) Metamodel
1.2.1. Definition of MAS metamodel elements
2.Phases of the (process name) Process
2.1.(First) Phase
2.1.1.Process roles
2.1.2.Activity Details
2.1.3.Work Products
2.2 (Second) Phase
2.2.1.Process roles
2.2.2.Activity Details
2.2.3.Work Products
... (further phases) ...
3.Work Product Dependencies

Table 1. The proposed process template

The template schema reported in Table 1 introduces the fundamental components of the process model definition. As it can be easily seen, the template has a structure that provides a natural decomposition of the process elements in a tree-like structure where the *Introduction* – including a description of the process lifecycle and the MAS metamodel – is at the root. Introduction is meant to give a general overview of the process detailing the original objectives of the process/methodology, its intended domain of application, scope, limits and constraints (if any), etc. The *Metamodel* part provides a complete description of the MAS metamodel adopted in the process with the definitions of its composing elements. This means the different conceptual elements considered when modeling the system must be identified and described. The focus on the MAS metamodel is not new in the agent-oriented community, and is also coherent with the current emphasis on model-driven approaches, which are always based on the system metamodel. The process is supposed to be composed – from the work-to-be-done point of view – by phases. Each phase is composed of activities that, in turn, may be composed of other activities or tasks. This structure is compliant to the SPEM specification which is explicitly adopted as a part of this template although with some (minor) extensions (see [10]). The next template part is represented by several Phase sections, one for each phase composing the whole process. The main aim of each phase is to define the phase from a pretty process-oriented point of view; that is, workflow, work products and process roles are the center of the discussion. Initially, the phase workflow is introduced by using a SPEM activity diagram which reports the activities composing the

phase, and by doing a quick description of work products and roles. A SPEM diagram follows reporting the structural decomposition of each activity in terms of the involved elements: tasks, roles and work products.

In the last section, the template discusses work products with a twofold goal: the first part aims at detailing the information content of each work product by representing which MAS model elements are reported in it (and which operations are performed on them). The second part focuses on the modeling notation adopted by the process in the specific work product. The work products are described by using a SPEM work product structured document. This diagram is a structural diagram reporting the main work product(s) delivered by each phase, and the diagrams are completed by a table that describes the scope of each work product. Finally, work product dependencies are reported in a specific diagram.

3 Case Studies

The next subsections discuss the documentation of three AO processes (PASSI, INGENIAS, and SODA) using the previously proposed template. In this way, the paper tries to evaluate the suitability of the template for modeling processes. The validation is significant because the chosen methodologies follow different kinds of process and have significant differences in other composing elements. For space reason, here we report only some excerpts of the methodologies documentation—the interested reader can found more details in [8]. In particular, the next subsections present the requirement analysis phase for each of the three AO methodologies considered in this paper.

3.1 PASSI

PASSI (Process for Agent Societies Specification and Implementation) is a step-by-step requirement-to-code methodology for designing and developing multi-agent societies.[4] The methodology integrates design models and concepts from both object oriented software engineering and artificial intelligence approaches.

The PASSI design process is composed of five models: the System Requirements Model regards system requirements; the Agent Society Model deals with the agents involved in the solution in terms of their roles, social interactions, dependencies, and ontology; the Agent Implementation Model is a model of the solution architecture in terms of classes and methods; the Code Model depicts the solution at the code level; and the Deployment Model describes the distribution of the parts of the system.

Following the schema proposed in Section 2, Figure 1 summarize the description of the System Requirements phase. In particular, this phase involves two different process roles, eight work products (four UML models and four text

[4] PASSI documentation can be found at `http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs/PASSI_SPEM_2_0_ver0.2.8.pdf`.

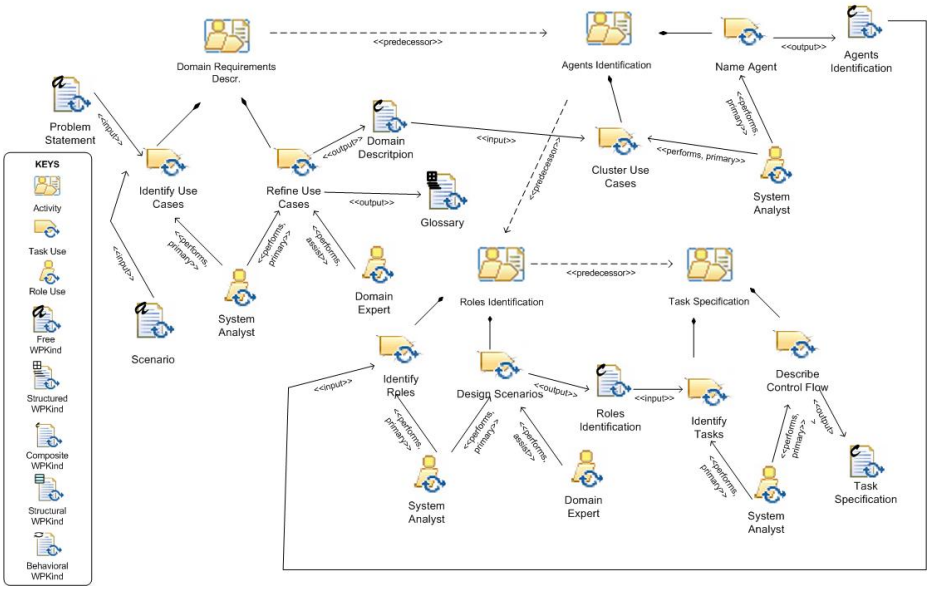

Fig. 1. The System Requirements activities, work products and stakeholders

documents) and four guidance documents (one for each UML model). The phase is composed of four activities: Domain Requirements Description, Agents Identification, Roles Identification and Task Specification, each of them composed of one or more tasks (for instance Identify Use Cases and Refine Use Cases) and delivering one work product as described by Figure 1. Tasks are under the responsibility of one or more stakeholders involved with the responsibility to perform or assist in the work to be done.

The System Requirements Model generates four composed work products (text documents including diagrams). Their relationships with the MAS metamodel elements (MMM) are described in Figure 2 where the containment relationship between each MMM element and a work product is labelled according to the action performed on the element (D means define/instantiate, R means relate, Q means quote/cite, RF means refine).

3.2 INGENIAS-UDP Process

The INGENIAS methodology covers the analysis and design of MAS and is intended for general use—that is, with no restrictions on application domains.[5] The software development process proposed by the methodology is based on Rational Unified Process [15]. The methodology distributes the tasks of analysis

[5] INGENIAS documentation can be found at `http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs/INGENIAS.pdf`.

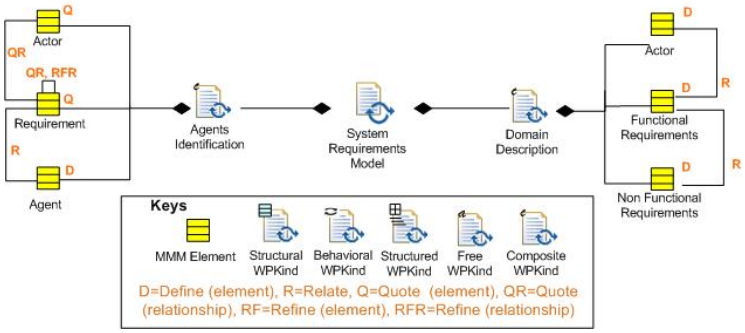

Fig. 2. The PASSI System Requirements documents structure

and design in three consecutive phases: Inception, Elaboration and Construction. Each phase may have several iterations (where iteration means a complete cycle of development). The sequence of iterations of each phase leads to the procurement of the final system. Figure 3 shows a detailed description of Inception phase of INGENIAS process.

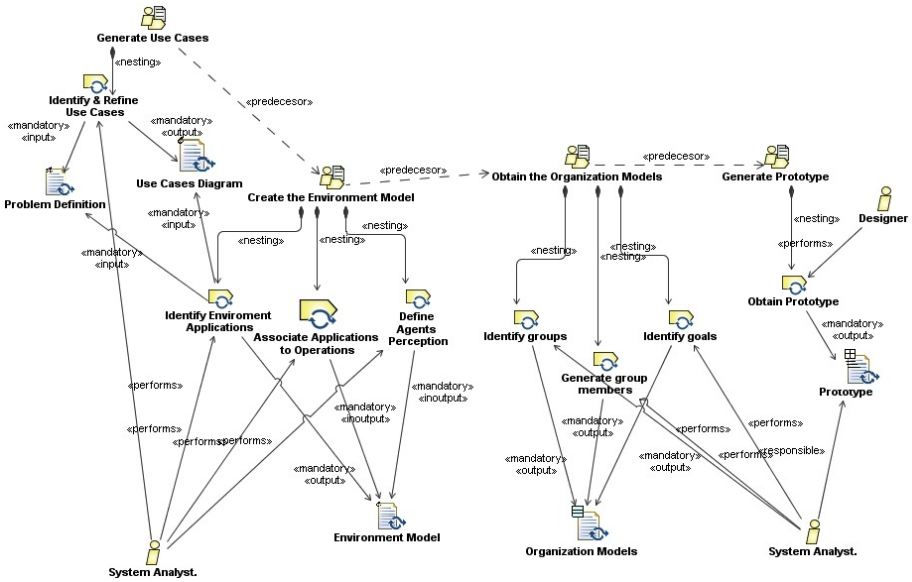

Fig. 3. INGENIAS Inception activities, workproducts and stakeholders

INGENIAS considers the development as starting from the document describing the problem, which is a mandatory input in this phase. The Inception phase is composed of several activities: generate use cases, create the Environ-

ment Model, Obtain the Organization Model and Generate Prototype. All these activities imply an important set of tasks and produce several workproducts as output, such as the Environment Model, the Organization Model or the Prototype. Besides, two roles are responsible as this phase: the System Analyst and the Designer.

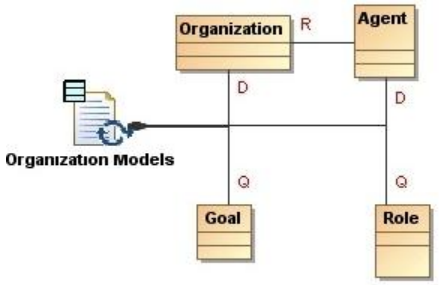

Fig. 4. INGENIAS Inception work products

As discussed above, one of the diagrams proposed in the template relates workproducts to the metamodel elements. In 4, the diagram is used for describing that the INGENIAS organisation models defines (D) the organization of the system and the agents related (R) to this organisation; while goal and role elements are only used (Q) but must have been defined previously.

3.3 SODA Process

SODA (Societies in Open and Distributed Agent spaces) [7, 16] is an agent-oriented methodology for the analysis and design of agent-based systems, which adopts the Agents & Artifacts meta-model (A&A) [17], and introduces a *layering principle* as an effective tool for scaling with the system complexity, applied throughout the analysis and design process.[6] The SODA process is organised in two phases, each structured in two sub-phases: the *Analysis phase*, which includes the Requirements Analysis and the Analysis steps, and the *Design phase*, including the Architectural Design and the Detailed Design steps. In addition, since the SODA process is iterative and incremental, each step can be repeated several times, by suitably exploiting the layering principle: so, for instance, if, during the Requirements Analysis step (Figure 5), the Requirements Analyst – one of the roles involved in the SODA process – recognizes some omissions or lacks in the requirements' definition, he/she can restart the *Requirements modelling* activity adding a new layer in the system or selecting a specific layer and then refining it through the *Requirement Layering* activity.

[6] SODA documentation can be found at `http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs/SODA.pdf`.

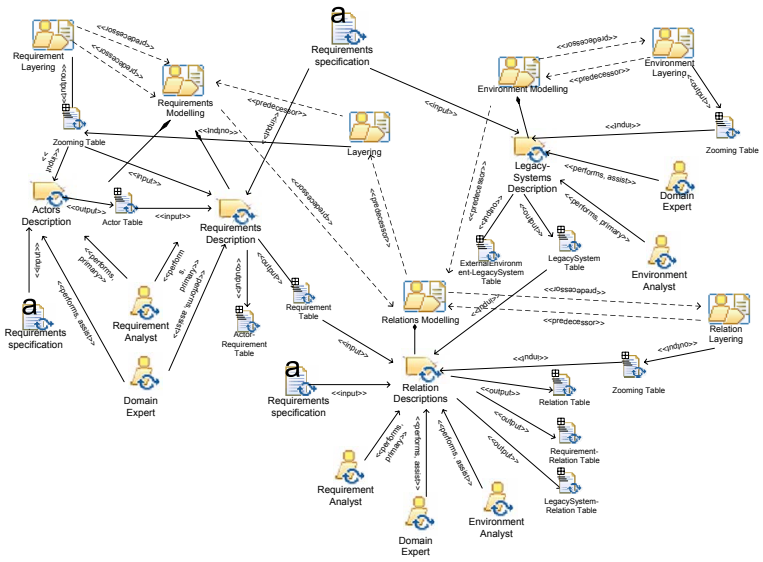

Fig. 5. The Requirements Analysis activities, work products and stakeholders

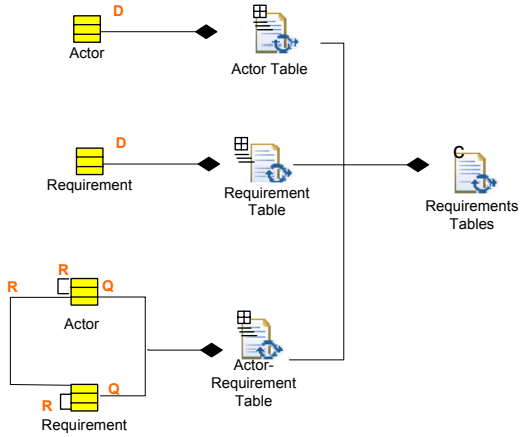

Fig. 6. The Requirements Analysis documents structure

In particular, the Requirements Analysis step involves three different process roles, nine work products (relational tables). The step is composed of three *normal* activities and four layering activities: the *normal* are Requirements Modelling, Relations Modelling, Environment Modelling, while the layering are Requirement Layering, Relation Layering, Environment Layering, and Layering. Tasks are under the responsibility of one or more roles involved with the responsibility to perform or assist in the work to be done.

Figure 6 reports only an excerpt of the Requirements Analysis documents structure. In SODA the work products are represented as relational tables organized in different sets. In particular, the diagram in Figure 6 reports the Requirements Tables set. This set describes the system requirements in terms of `Requirement` and `Actor` concepts of the SODA's metamodel: each table has a specific relationships with one or more MAS metamodel elements. For example, the Actor table and the Requirement table *define* (D) [10] respectively the `Actor` and the `Requirement` while Actor-Requirement table *quote* (Q) both and *relate* (R) `Actor` and `Requirement`.

4 Proposals for FIPA DPDF Template Improvement

During the application of the FIPA DPDF template to the three methodologies, we collected some important feedbacks on its effectiveness. Most of them will be discussed in the next section as an assessment of the work done, whereas a couple are now proposed in terms of proposals for the improvement of the FIPA DPDF template.

The first issue concerns the absence in the template of a clear indication of where to describe the techniques and guidances [9] applied both in the overall process and in some specific part of the process. In particular, when trying to document SODA we had some problems in the documentation of the layering technique. This is quite a peculiar aspect of SODA, which adopts the layering principle as a tool for managing the system complexity and it spreads all over the process—excluding the Detailed Design phase. We found two issues related to the layering documentation: *(i)* where to put the layering technique description; *(ii)* the definition of the best structure for the documentation.

In order to manage the above issues we created a new sub-section in the template introduction (Table 1); this is like the description of the single phases, since the layering technique has a portion of process with its specific activities and tasks, and obviously its work products. The proposed change perfectly suits the need for introducing the layering technique before the description of the details of the process and the structure of the section is flexible enough to fit similar needs arising in other processes.

Another limit we found in the FIPA DPDF template is the lack of a specification for detailing the content of a task. Activities are decomposed – or better decomposable according to the needs – in tasks in section 2.1.2 (see Section 2) of the template but this may not be general enough. What about the description of quite a complex task? SPEM provides the method engineer with the opportu-

nity to use the *Step* element for decomposing tasks. It is worth to remind that – according to SPEM specification [9] – a Step is *"a Section and Work Definition that is used to organize a Task Definition as Content Description into parts or subunits of work. ... A Step describes a meaningful and consistent part of the overall work described for a Task Definition. The collection of Steps defined for a Task Definition represents all the work that should be done to achieve the overall development goal of the Task Definition"*. According to this definition the usage of the Step element may prove to be very useful. It may happen – and we actually found some occurrences of that in our processes – that one specific task is too complex to be exhaustively described in the text proposed in section 2.1.2 of the template (see Section 2). It may be even the case to describe a task with an activity diagram reporting the flow of work to be done. Steps would be the main components of this diagram and, in turn, they would need a text description of the work to be done inside them.

Actually, the FIPA DPDF template specification document [8] proposes the structure for describing activities as showed in Table 2.

4.1.2.1. Activity 1 **GOAL:** Describe the work to be done within this activity **STRUCTURE:** Details of tasks and sub-activities are specified with a table that includes the following columns: - Activity: name of the activity studied in this subsection. - Tasks/Sub-Activity: sub-activity or task described in this row. - Task/Sub-activity Description. - Roles involved. Optionally, the control flow within a Task can be illustrated by a stereotyped UML Activity Diagram. These diagrams explain the execution of complicated Tasks by denoting the possible sequences of Steps, which are identified by the << `steps` >> stereotype. Details on this modeling of Tasks can be found in the current SPEM specification. When documenting a Task in this way, the diagrams are appended and each diagram is discussed in a separated paragraph that explains the illustrated steps and theirx relations.

Table 2. The activity description section in the current FIPA DPDF template

The FIPA DPDF template already prescribes the possibility to detail tasks by using steps in forms of activity diagrams, however it does not give any hint on how to document them. In order to improve the template, we propose to introduce a new optional subsection in each activity description section as depicted in Table 3.

As an example of such an extension, the decomposition of the INGENIAS Identify Environment Application task is presented in Table 4. The activity diagram depicting the workflow is omitted because of space concerns and also because the steps are performed one after the other in a simple way.

4.1.2.1.1 Decomposition of Task x of Activity 1 **GOAL:** Describe the work to be done within Task x of Activity 1. **STRUCTURE:** The workflow may be depicted by using an activity diagram reporting the steps to be done within the task. Details of steps are specified with a table reporting the following columns: - Activity: name of the activity the task studied in this subsection belongs to. - Task: name of the task detailed in this subsection. - Step: name of the Step described in this row of the table - Step Description: plain text describing the work to be done within this step. - Roles involved: roles involved in executing this step.

Table 3. The proposed extension to the FIPA DPDF template for a detailed description of tasks decomposition in steps.

Activity	**Task**	**Step**	**Step Description**	**Roles involved**
Create the Environment Model	Identify Environment Applications	Identify External Applications	Find standard packages and external software agents need to use or to communicate with	System Analyst
Create the Environment Model	Identify Environment Applications	Identify Internal Applications	Identify centralized software services agents has to shared and whose nature is not like that of an agent	System Analyst

Table 4. Steps in the INGENIAS Identify Environment Application Task

5 Discussion

This paper evaluates a template for process documentation that seems to provide a good framework in the documentation of processes for AO development. This template is based on an approach similar to the one proposed by Rumbaugh [18] in introducing UML. The approach prescribes the removal of all cluttering information – for instance, different notations – in order to highlight commonalities (and differences). As a result, the study of a new methodology becomes easier to a designer who is already fluent with the documentation style adopted in this approach. The FIPA DPDF template proposes a division of the process in phases, activities and tasks as introduced in Section 2. In this paper, we were able to identify (with a similar granularity) the phases, activities and tasks for the processes introduced in PASSI, INGENIAS and SODA (see Figures 1, 3 and 5) without specific problems—thus proving the soundness of the approach.

In particular, the figures show the flow of activities, the work products and the stakeholders of the first phase of each methodology. By analysing the figures, it is easy to understand the specific flow of activities and tasks to be followed when using the methodologies. On the other hand, the diagrams highlight the

differences among the three methodologies such as for instance the different attention paid to the environment modelling. This is a primary activity in SODA, a task in INGENIAS, while PASSI delays the study of the environment to the next phase. Another difference regards the identification of roles and agents: this is done in the first phase of the process in PASSI and INGENIAS, whereas SODA defines the same abstractions only in the design phase.

An important feature of the template is the attention paid to the MAS metamodel adopted in the process. Such a feature provides an interesting point in methodological comparison. For instance, from the comparison of the documentations produced in this study, we easily deduce that INGENIAS (Figure 4) has a reduced set of models, which however are quite complex since each of them includes many concepts. On the other hand, PASSI and SODA – Figures 2, 6 – have respectively more diagrams and tables, but each of them introduces only a few concepts. The use of the template easily supports the identification of such differences.

Furthermore, in the template, the MAS metamodel elements and their relationships are also related with work products depicting them—see respectively Figures 2, 4, and 6. Considering work products as a part of the process is fundamental for fragment definition and usage, as long as, the user must take into account the desirable results for selecting a fragment or she/he must consider what inputs are needed before it is possible to initiate a phase or an activity of the process. Moreover, the definition of different processes for several methodologies using this template – see also [8] for the documentation of others AO methodologies – suggests that it is general enough, because good results have been obtained for three different methodologies.

As previously discussed, we point out several benefits in using the proposed template. First of all, the template makes it easier to understand the process workflow, and also produces a documentation that may help in studying it. Moreover, it seems evident that it will be easier to study a new methodology when this new one is documented with an already known approach. For instance, PASSI and SODA metamodels are different in the content – different elements, concepts and models – but the similar description approach largely allows for an easy identification and study of differences between them.

Another important benefit of defining the process is that it provides a starting point for fragments extraction, and therefore for process elements reuse. The reuse would start by identifying fragments considering, for instance, as a starting point the work products produced by the fragment. The template provides diagrams that facilitate the identification. For instance the work products dependencies diagram makes it possible to introduce an order in the work products obtained. On the other hand, the diagrams detailing the activities identify the input and output work products of each task. All such information should be considered when defining a fragment.

Some limitations where noticed when documenting the processes. One issue is related to the SPEM notation: the presence of the layering activities in SODA leads to the construction of diagrams that are very difficult to understand due

to the huge number of strictly-related activities. In particular, in Figure 5 there are four different layering activities – one for the iteration and three for the models refinement –, and the only *predecessor* relation is not powerful enough for explaining the right flow of activities in the Requirements Analysis phase—so this diagram alone is not sufficiently expressive. In this diagram, at the best of our knowledge, there is no way for expressing too many information in a clear way. The problem mainly arises in SODA because of the adoption of the layering technique but generally speaking it may regard other methodologies. The essence of the problem is deeply relied to the SPEM notation and we have not solved this problem yet; we plan to find alternative solutions in the future.

Some other minor problems have arisen when documenting the processes. These problems are more related with identifying specific details of the process from available documentation rather than with the template itself. Usually, when defining a methodology authors are more worried about identifying the models to construct, the concepts to define, etc. than in detailing phases and activities to be done or in indicating the order of these activities. In some way, the adoption of the template would force the designer for a new methodology to produce a complete specification thus improving the quality of the result.

6 Conclusions and Further Work

In this paper we used the FIPA DPDF template for documenting three different AO design processes. Documentation of processes has many advantages such as: comparing and evaluating methodologies in an easy way, simplifying fragment definition and selection, and so on. This work has demonstrated the power of the template in process documentation, and sketched some of its advantages. Nevertheless, it has been made available to the scientific community, so that other processes and/or other methodologies may be defined used the template.

This work is an initial step toward the definition of a standard for fragment documentation, extraction and use. This means that in the future the models produced for these methodologies will be used for identifying and documenting fragments. The fragments will then be reused and integrated so as to provide new ways of developing AO systems. Besides, although all this work has been done within the frame of AO development, we guess that the template could be general enough to define methodologies in other fields of development. Further work should be done to prove such a statement.

7 Acknowledgements

This work has been partially supported by the project *Novos entornos colaborativos para o ensino* supported by Xunta de Galicia with grant 08SIN009305PR and by the FRASI project of the Italian Ministry of Education and Research (MIUR).

References

1. Brian Henderson-Sellers, C.G.P.: Metamodelling for software engineering. ACM Press New York, NY, USA (2003)
2. Sorbonne, U.D.P., Rolland, C., Rolland, C.: A primer for method engineering. In: Proceedings of the INFORSID Conference. (1997) 10–13
3. Cuesta, P., Gómez, A., González, J., Rodríguez, F.J.: The MESMA methodology for agent-oriented software engineering. In: Proceedings of First International Workshop on Practical Applications of Agents and Multiagent Systems (IWPAAMS'2002). (2002) 87–98
4. O'Malley, S.A., DeLoach, S.A.: Determining when to use an agent-oriented software engineering pradigm. In Wooldridge, M., Wei, G., Ciancarini, P., eds.: Agent-Oriented Software Engineering. Second Int. Workshop, AOSE 2001. Volume 2222 of Lecture Notes in Computer Science. Springer-Verlag (2002)
5. Bernon, C., Cossentino, M., Pavón, J.: Agent-oriented software engineering. Knowl. Eng. Rev. **20** (2005) 99–116
6. Pavón, J., Gómez-Sanz, J.: Agent Oriented Software Engineering with INGENIAS. Multi-Agent Systems and Applications III **2691** (2003) 394–403
7. Omicini, A.: SODA: Societies and infrastructures in the analysis and design of agent-based systems. In Ciancarini, P., Wooldridge, M.J., eds.: Agent-Oriented Software Engineering. Volume 1957 of LNCS. Springer-Verlag (2001) 185–193
8. IEEE FIPA Design Process Documentation and Fragmentation: IEEE FIPA Design Process Documentation and Fragmentation Homepage. http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/ (2009)
9. O.M.G.: Software Process Engineering Metamodel Specification. Version 2.0, formal/2008-04-01. `http://www.omg.org/` (accessed on June 25, 2009) (2008)
10. Seidita, V., Cossentino, M., Gaglio, S.: Using and extending the spem specifications to represent agent oriented methodologies. In Luck, M., Gómez-Sanz, J.J., eds.: AOSE. Volume 5386 of Lecture Notes in Computer Science., Springer (2008) 46–59
11. Cossentino, M.: From requirements to code with the PASSI methodology. [19] chapter IV 79–106
12. Pavòn, J., Gòmez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. [19] chapter IX 236–276
13. Molesini, A., Nardini, E., Denti, E., Omicini, A.: Situated process engineering for integrating processes from methodologies to infrastructures. In Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M., eds.: 24th Annual ACM Symposium on Applied Computing (SAC 2009). Volume II., Honolulu, Hawai'i, USA, ACM (2009) 699–706
14. Cernuzzi, L., Cossentino, M., Zambonelli, F.: Process models for agent-based development. Engineering Applications of Artificial Intelligence **18** (2005) 205–222
15. Kruchten, P.: The Rational Unified Process, An Introduction. Addison Wesley (1998)
16. Molesini, A., Omicini, A., Denti, E., Ricci, A.: SODA: A roadmap to artefacts. In Dikenelli, O., Gleizes, M.P., Ricci, A., eds.: Engineering Societies in the Agents World VI. Volume 3963 of LNAI. Springer (2006) 49–62
17. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems **17** (2008) 432–456
18. Rumbaugh, J.E.: Notation notes: Principles for choosing notation. JOOP **9** (1996) 11–14
19. Henderson-Sellers, B., Giorgini, P., eds.: Agent Oriented Methodologies. Idea Group Publishing, Hershey, PA, USA (2005)