Documenting SODA: An Evaluation of the Process Documentation Template

Ambra Molesini
ALMA MATER STUDIORUM – Università di Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
Email: ambra.molesini@unibo.it

Andrea Omicini
ALMA MATER STUDIORUM – Università di Bologna a Cesena
Via Venezia 52, 47521 Cesena, Italy
Email: andrea.omicini@unibo.it

***Abstract*—This paper presents an experimental evaluation of the methodology documentation template proposed by the IEEE FIPA Design Process Documentation and Fragmentation working group [1]. Generally aimed at agent-oriented methodologies, the template is here put to the test by documenting the SODA methodology, so as to obtain a partial but significant evaluation of the template's strengths and weaknesses.**

I. INTRODUCTION

While artificial systems grow in complexity, the role of models, methodologies and development processes in their engineering becomes increasingly relevant. In the field of computational systems, AOSE (Agent-Oriented Software Engineering) methodologies and processes are research subjects of foremost interest since agent-based models and technologies are nowadays recognised as providing one of the most suitable approaches to the engineering of complex software systems such as self-organising and pervasive systems.

There is common agreement both in the traditional software engineering and in the AOSE fields that there is not a unique methodology or process that fits all the application domains [2]. This means that the methodology or process must be adapted to the particular characteristics of the domain for which the software is developed.

A variety of (special-purpose) AOSE methodologies have been defined in the past years [3], [4], [5], [6], [7] to discipline and support the development of multi-agent systems (MAS). However, the increasing number of AOSE methodologies and the key role in the construction of new ad-hoc methodologies or processes played by the Situational Method Engineering [8], [9], [10] have hindered the applicability of AOSE methodologies. In fact, each methodology has its own specific meta-model, notation, and process. All of these features are fundamental for a correct understanding of a methodology, and should then be described in a manual or book that the project manager and his/her team of developers closely follow [11]. However, such manuals typically do not represent in a proper way the methodologies since they do not suitably formalise those three crucial aspects. Instead, methodologies are typically explained through an ambiguous textual description focussing more on the notation language and on the work products rather than on other essential aspects such as the process, the concepts meta-model and the application domain.

Although it is possible to describe a methodology / process without an explicit documentation, formalising its underpinning ideas is valuable for checking consistency, or when planning extensions or modifications: there, a formal documentation can be exploited to check both the software development process and the completeness and expressiveness of methodologies. More generally, the relevance of documentation becomes clear when studying the completeness and the expressiveness of a methodology / process, and when comparing / integrating different methodologies / processes together.

In this context, the IEEE FIPA Design Process Documentation and Fragmentation (DPDF) working group [1] has recently proposed a template for documenting AOSE methodologies. So, the objective of this paper is to present an experimental evaluation of the FIPA DPDF template by means of the application of such a template to the SODA methodology [12], [13].

Accordingly, the paper is structured as follows. Section II briefly presents the IEEE FIPA DPDF template, while Section III presents some excerpts from the SODA documentation. Then, Section IV discusses the template's strengths and weaknesses identified during the creation of the SODA documentation. Conclusions are reported in Section V.

II. PROCESS DOCUMENTATION TEMPLATE IN A NUTSHELL

The IEEE FIPA DPDF working group has recently proposed a template for documenting AO methodologies and processes. This template takes into account the three aforementioned methodology features—meta-model, notation, and process. In the first place, the template has been conceived with no reference to any particular methodology—which should guarantee that all methodologies can be documented using the proposed template. Moreover, the template is also neutral regarding the meta-model and/or the modelling notation adopted in the description of the the methodology.

Secondly, the template has a simple structure resembling a tree—see above for further details. This implies that the documentation can be built up in a natural and progressive way, addressing in first place the general description and meta-model definition – which constitute the root elements of the methodology –, subsequently detailing the branches

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 I
THE IEEE FIPA DPDF PROCESS TEMPLATE

representing the phases. Next, thinner branches like activities or sub-activities have to be described. As a result, the template can in principle support complex methodologies and different situations.

In the third place, the use of the template is easy for a software engineer as it relies on few assumptions. Moreover, the notation suggested is the OMG's standard Software Process Engineering Metamodel (SPEM) [14] along with few extensions [15].

So, in the remainder of this section, we only report a brief presentation of the template—interested readers can refer to [1] for the details. The template is based on the definition of process and process model as proposed by [16]. 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 suggested in [15]—given that the MAS metamodel might constrain the way in which fragments can be defined and reused.

The template schema reported in Table I introduces the fundamental components of the process model definition. The template has a structure that provides a natural decomposition of the process elements in a tree-like structure, whose root is the Introduction section, which includes a description of the process lifecycle and the MAS metamodel. Introduction tries to provide a general overview of the process detailing the original objectives of the methodology, its intended domain of application, its scope, limits and constraints (if any), etc. The *Metamodel* part provides a complete description of the MAS metamodel adopted in the process, along with the definitions of its composing elements. Thus, the different conceptual elements considered when modelling the system should be identified and described. The attention to the MAS metamodel is not new in the agent-oriented community, and it is also coherent with the emerging model-driven approaches, which are always based on the system metamodel. The methodology's 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. Such a structure is compliant to the SPEM specification that is explicitly adopted as a part of this template, although with some (minor) extensions—see [15]. The next part of the template is represented by a number of Phase sections, one for each phase composing the whole process. The main aim of each Phase section is to define the phase mainly from a process-oriented point of view; that is, workflow, work products and process roles are the centre 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 focusses on the modeling notation adopted by the methodology 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.

III. SODA DOCUMENTATION

This section presents only the main excerpts from the SODA documentation—interested readers can find the full documentation at [17]. In particular next subsections describe SODA according to the template structure (TABLE I): Subsection III-A presents the introduction of the documentation, Subsection III-B details the SODA phases, finally Subsection III-C shows the SODA work product dependencies.

A. Documentation Introduction

SODA (Societies in Open and Distributed Agent spaces) [17] is an agent-oriented methodology for the analysis and design of agent-based systems, which adopts the Agents & Artifacts (A&A) building blocks [18], and introduces a *layering principle* as an effective tool for scaling with the system complexity, applied throughout the analysis and design process. Since its first version [12], SODA is not concerned with *intra-agent* issues: designing a multi-agent system with SODA leads to defining agents in terms of their required observable behaviour and their roles in the multi-agent system. Then, whichever methodology one may choose to define the agent structure and inner functionality, it could be easily used in conjunction with SODA. SODA focusses on *inter-agent* issues, like the engineering of societies and environment for MAS.

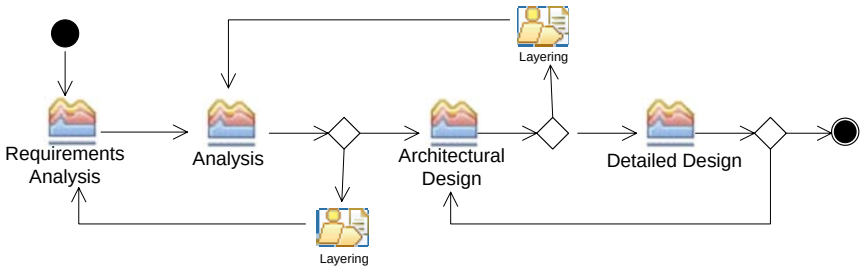

Fig. 1. SODA process lifecycle

The abstractions belonging to the SODA meta-model – called MAS Meta-model Elements (MMMElements) – [17] are logically divided into three categories: *i)* abstractions for modelling / designing the system active part (task, role, agent, etc.); *ii)* abstractions for the reactive part (function, resource, artifact, etc.); and *iii)* abstractions for interaction and organisational rules (relation, dependency, interaction, rule, etc.). In its turn, the SODA *process* (Fig. 1) 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. Due to the transformational nature of SODA, each sub-phase models (designs) the system exploiting a specific and different subset of the SODA MMMElements: each subset always includes at least one abstraction for each of the above categories—that is, at least one abstraction for the system active part, one for the reactive part, and another for interaction and organisational rules.

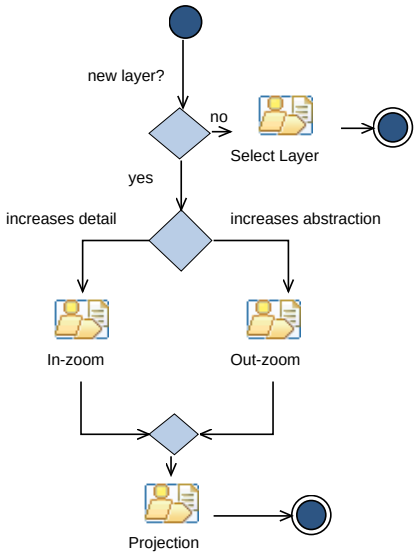

Fig. 2. The layering capability pattern

In addition, since the SODA process (Fig. 1) is iterative, each step can be repeated several times, by suitably exploiting the SODA layering mechanism (Fig. 2). The layering in Fig. 1 is represented as a simple Activity of the process: actually, it is a *capability pattern* [19], i.e., a reusable portion of the process—see Fig. 2. In particular, layering exhibits two different functionalities: *(i)* the selection of a specific layer for refining / completing the abstractions models in the methodology process, and *(ii)* the creation of a new layer in the system by in-zooming (i.e., increasing the system detail) or out-zooming (i.e., increasing the system abstraction) activities. In the latter case, the layering process terminates with the projection activity needed to project the abstractions from one layer to another "as they are", so as to maintain the consistency in each layer. The layering pattern is also used within sub-phases—except in the Detailed Design, where the layering principle is, by definition, not applicable.

B. Phases of the SODA Process

1) Requirements Analysis: The goal of Requirements Analysis is to characterise of both the customers' requirements and the legacy systems with which the system should interact, as well as to highlight the relationships among requirements and legacy systems. The diagram in Fig. 3 presents the Requirements Analysis activities, each related to its corresponding task(s) use [19]. The diagram also reports the roles involved in this step and the work products that should be produced—i.e., the SODA tables describing the abstract entities of the Requirements Analysis. The *Requirements Modelling* activity is composed by the Actors Description and Requirements Description tasks both performed by the Requirement Analyst role supported by the Domain Expert (Fig. 3). In this activity, *Requirement* and *Actor* are used for modelling the customers' requirements and the requirement sources, respectively, while the *ExternalEnvironment* notion is used as a container of the *LegacySystems* that represent the legacy resources of the environment in the *Environment Modelling* activity. The relationships between requirements and legacy systems are modelled in the *Relation Modelling* activity in terms of a suitable *Relation*.

As one could see in Fig. 3, other three activities appear in the diagram – *Requirement Layering*, *Environment Layering*, and *Relation Layering* – and are strictly related – through the *predecessor* relationship – to the previously described activities. These activities represent the Layering capability pattern since during each activity of this step it is possible to create a new (or, to select a specific) layer for detailing (or abstracting) the specific model under investigation. It is important to note

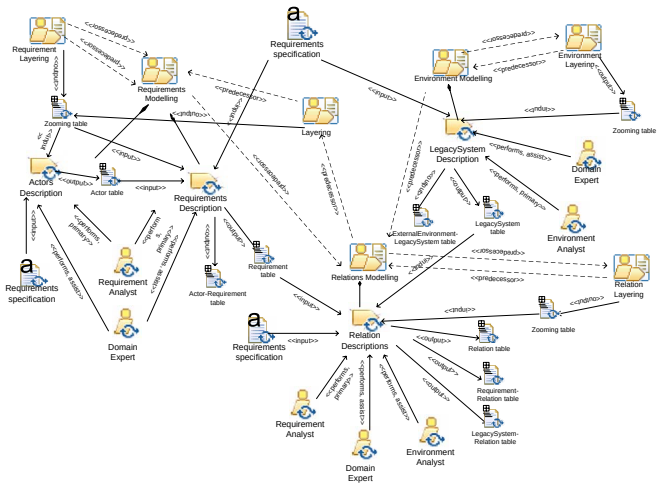

Fig. 3. Requirements Analysis flow of activities, work products and roles

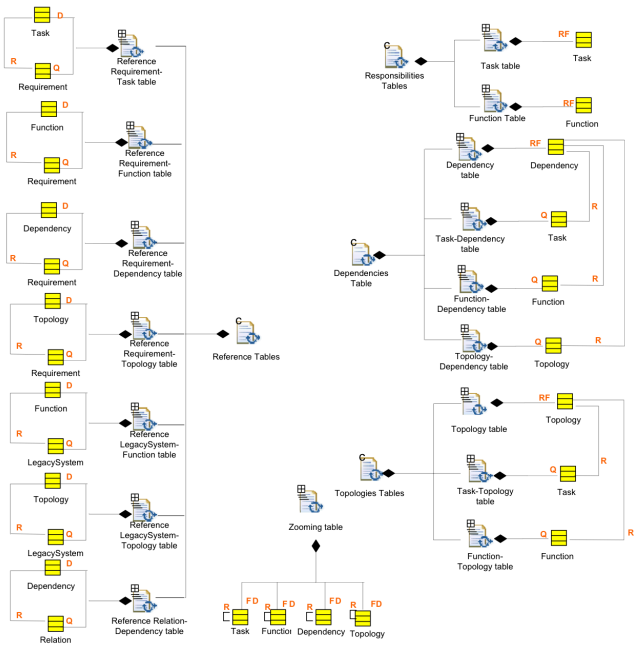

Fig. 4. Analysis work products

that layerings do not represent an iteration of the step: instead, they refine the model over which they are applied—and are a peculiarity of SODA. The Requirements Analysis iteration is done through the *Layering* activity located between the *Relation Modelling* and the *Requirements Modelling* activities (Fig. 3). The same consideration also applies to the Analysis and Architectural Design steps.

2) Analysis: The first activity in the Analysis step is *Moving from Requirements*, where the MMMElements identified in the previous step are mapped onto the MMMElements adopted in this stage to generate the initial version of the Analysis models. The work product of this activity is represented by a set of relational table called Reference Tables depicted in Fig. 4, where the relation between Analysis work-products and Analysis MMMElements are showed (D means define/instantiate, R means relate, Q means quote/cite, F means refine). In particular, Reference Tables defines (label D in Fig. 4) the Analysis MMMElements and puts them in relation (label R in Fig. 4) with the Requirements Analysis MMMElements.

The Analysis step expresses the requirement representation in terms of more concrete MMMElements such as *tasks* and *functions*. Tasks are activities requiring one or more competences and are analysed in the *Task analysis* activity, while functions are reactive activities aimed at supporting tasks analysed in the *Function analysis* activity. The structure of the environment, analysed in the *Topology analysis* activity, is also modelled in terms of *topologies*—i.e., topological constraints over the environment. The relations highlighted in the previous step are here the starting point for the definition of *dependencies* among such abstract entities in the *Dependency analysis* activity.

3) Architectural Design: This stage (Fig. 5) is one of the more complex sub-phases in SODA. The first activity is *Transition* (Fig. 5), where the MMMElements identified in the previous step are mapped onto the MMMElements adopted in this stage so as to generate the initial version of the Architectural Design models. The main goal is to assign responsibilities for achieving tasks to *roles – Role Design* activity composed by the Action Design task – and for providing functions to *resources—Resource Design* activity composed by Operation Design task. In order to attain one or more tasks, a role should be able to perform *actions – Role Design* activity –; analogously, the resource should be able to execute *operations* providing one or more functions—*Resource Design* activity. The topology constraints lead to the definition of *spaces*, i.e., conceptual places structuring the environment in the *Space Design* activity. Finally, the dependencies identified in the previous phase become here *interactions* and *rules*. Interactions represent the acts of the interaction among roles, among resources and between roles and resources, and are designed in the *Interaction Design* activity; rules, instead, enable and bound the entities' behaviour and are designed in the *Constraint Design* activity.

4) Detailed Design: The Detailed Design step (Fig. 6) is the only stage where the layering principle is not applicable, since its goal is to choose the most adequate representation level for each architectural entity, thus leading to depict one (detailed) design from the several potential alternatives architectures outlined in the previous step. So, as shown in Fig. 6, the first activity of this sub-process is *Carving*, which represents a sort of boundary between the Architectural Design and the Detailed Design, where the chosen system architecture is "carved out" from all the possible architectures.

The next activity is *Mapping* (Fig. 6), where the carved MMMElements are mapped onto the MMMElements adopted in this stage, thus generating the initial version of the Detailed Design models. Then, the Detailed Design presents three different activities. The *Agent Design* activity design the active

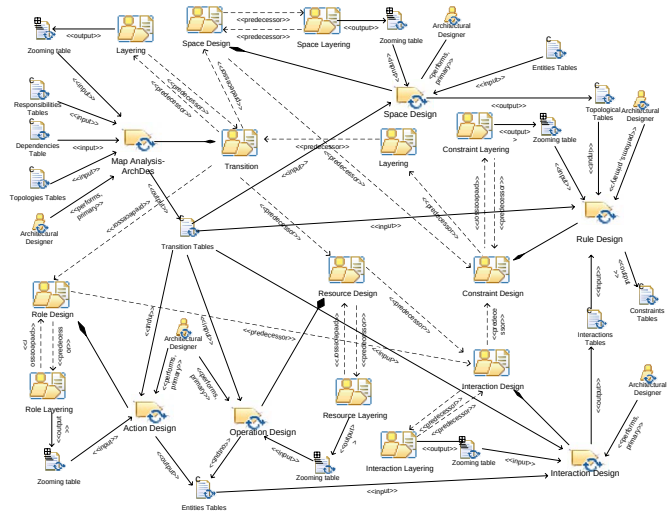

Fig. 5. Architectural Design flow of activities, work products and roles

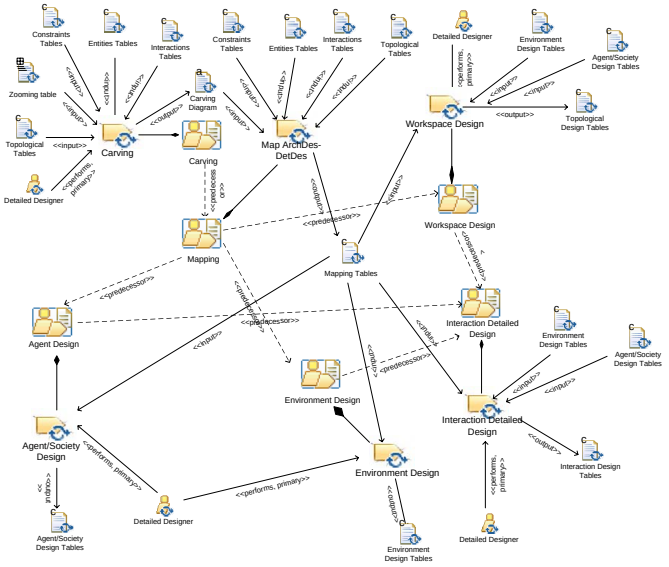

Fig. 6. Detailed Design flow of activities, work products and roles

part of the system in terms of *Agent* and agent *Society*. More precisely, agents are intended here as autonomous entities able to play several roles, while a society can be seen as a group of interacting agents and artifacts whose overall behaviour is essentially autonomous and proactive: they are designed during the *Agent Design* activity.

In the *Environment Design* activity, the resources identified in the previous step are here mapped onto suitable *Artifact*, while *Aggregate* is defined as a group of interacting agents and artifacts whose overall behaviour is essentially functional and reactive.

The *Workspace Design* activity defines the topology of the environment in terms of *Workspace*s that take the form of an open set of artifacts and agents: artifacts can be dynamically added to or removed from workspaces, and agents can dynamically enter (join) or exit workspaces. Finally, during the *Interactions Design* activity, the interaction are designed in terms of *Use* – interaction between agents and artifacts –, *Manifest* – interaction between artifacts and agents –, *SpeakTo* – interaction among agents – and *LinkedTo* – interaction among artifacts – MMMElements.

C. Work Product Dependencies

The diagram in Fig. 7 describes the high-level dependencies among the different sets of SODA work products—called composite work products. In fact, due to the huge number of SODA work products – each table represent a work product – it is not possible to create a comprehensive and readable diagram depicting all the work products and their dependencies.

IV. Discussion

This paper evaluates a template for process documentation that seems to provide a good framework in the documentation of processes for agent-oriented development. However, the current version of the template presents several limitations—some of which related more to the notational language than to the template structure.

The main weakness of the current template structure is related to the absence of a clear indication of where to describe the tools and guidance [19] applied both in the overall process and in the specific part of the process. In particular, when trying to document SODA we had some problems in the documentation of the layering. This is quite a peculiar aspect of SODA, which adopts the layering principle as a tool for managing the system complexity spread all over the process—excluding the Detailed Design step.

The first issue was about the place where to place the layering description. From our experience in developing and explaining SODA, we understood that the layering should be presented before detailing the different SODA's steps. This because the layering is not a simple mechanism for creating the iteration in the SODA process, but it is also a way for refining single SODA model, so it is largely adopted in the first three steps. Looking at Fig. 3 and Fig. 5, it is possible to see both the layering activity (Layering) that *inducting* – if necessary – a new iteration of the step and the different layering activities (such as Requirement Layering, Relation Layering, etc.) devoted to the models refinement. The above considerations led us to the definition of a new sub-section in the documentation introduction (TABLE I) where the layering is documented. After our requests for explanations, the right place where the process guidance and mechanisms should be positioned is now under discussion inside the IEEE FIPA DPDF working group.

The second issue coming from the layering documentation is related to the structure of the newly created sub-section. We decided to structure the new sub-section in a very similar way to the structure of the single phases, since the layering has its portion of process, its specific activities and tasks, and obviously its work product. We also created a specific diagram

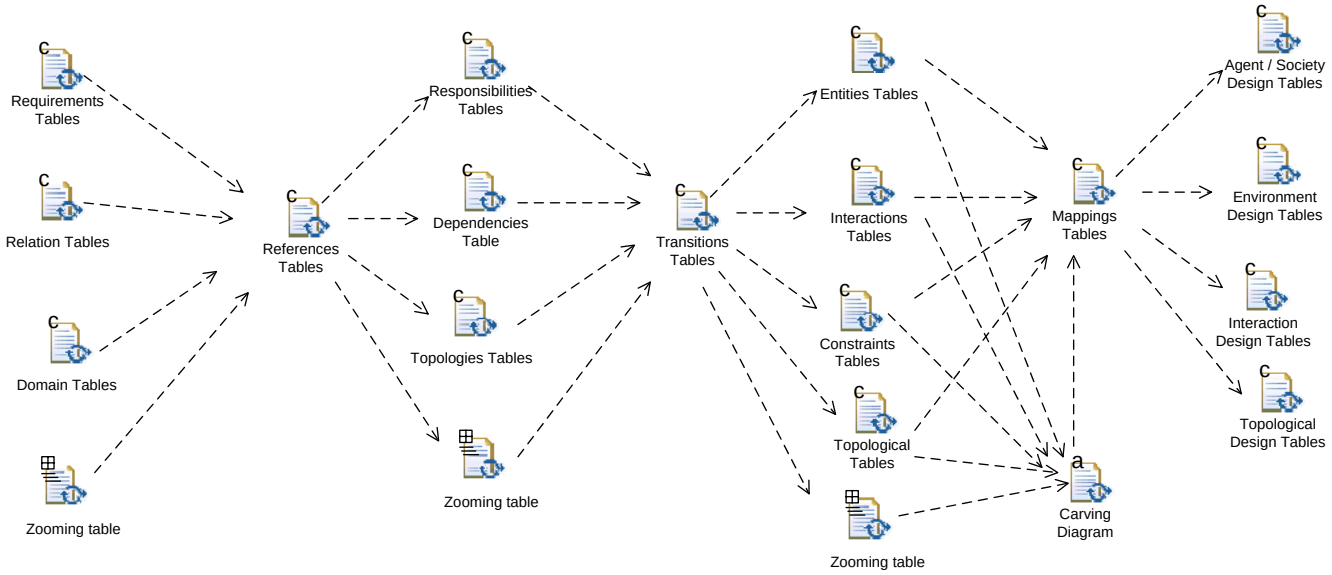

Fig. 7. SODA work products dependencies

– like the diagram in Fig. 4 – that puts the Zooming table – i.e., the layering work product – in relation with the SODA's MMMElements, in order to explain how the layering is closely related to the SODA meta-model [17].

The last layering issue is related to the SPEM notation. In fact, the diagrams in Fig. 3 and Fig. 5 are very difficult to understand due to the huge number of strictly-related activities. In particular, in Fig. 5 there are six different layering activities – one for the step iteration and five for the models refinement – and the only *predecessor* relation is not powerful enough for explaining the right flow of activities in the Architectural Design step—so this diagram alone is not sufficiently expressive. For instance, the Role Design activity in Fig. 5 is the predecessor of two different and alternative activities, on the one hand the Role Layering and on the other hand the Interaction Design. In this diagram, at the best of our knowledge, there is no way for expressing this alternative. Even though such an alternative is documented in another activity diagram (see the SODA documentation at [17]), in our view the alternative paths should be highlighted also in the diagram in Fig. 5.

This consideration about the SPEM limitation leads to another SPEM issue tied to the work product documentation. In different diagrams such as those presented in Fig. 5 and Fig. 7 we are able to document only the SODA composite work products in order to have readable diagrams. The diagrams proposed by SPEM seems not so suitable for documenting those methodologies which a great number of work products like SODA.

Apart from the weaknesses highlighted above, the template seems very valuable, since it forced us to re-think the way of presenting our methodology. Usually, when discussing a methodology, authors are more worried about identifying the models to construct, the concepts to define, etc. than in detailing the phases and the activities to do, or in defining the order of these activities. Thanks to this template now the authors should pay more attention to process-related aspects.

In addition, a major task of the documentation template should be to offer a good support for process evaluation and comparison. When comparing SODA to the other methodologies documented in [1], we can notice for instance that PASSI [20] and SODA meta-models are different in the content (different elements, concepts and models); however, using the same approach in describing them easily enables the study of similarities and differences between them. Furthermore, from the comparison of the SODA documentation and the documentations (see [1]) of PASSI and INGENIAS [21], we have easily deduced that INGENIAS has a limited set of models, which are however quite complex since each of them include many concepts. On the other hand, PASSI and SODA have respectively more diagrams and tables, but each of them introduces few concepts. In addition, the documentation highlights the different attention paid to the environment modelling. This is a primary activity in SODA, a task in INGENIAS, while PASSI demands the study of the environment in the next phase. Another difference concerns the identification of role and agent concepts that in PASSI and INGENIAS is done in the first phase of the process, whereas SODA defines such abstractions only in the design phase. So, the use of the template easily supports the identification of such differences.

V. Conclusions and Future Work

In this paper, we used IEEE FIPA DPDF template for documenting the SODA design process, and represents an experimental evaluation of the template. SODA proved to be a good test for the template since the creation of the SODA documentation highlighted a weakness in the template

structure. The SODA documentation showed the power of the template in process documentation, and sketched some of its benefits briefly comparing SODA to PASSI and INGENIAS.

In addition, the creation of the SODA documentation helped us formalising the SODA process and organising in a proper way the huge number of SODA work products. In particular, diagrams such as those depicted in Fig. 4 proved to be very valuable in relating the SODA work products to the SODA MMMElements. This improved the general SODA understanding—including ours. Finally, the SODA documentation could be considered as the starting point for the SODA fragmentation, since in the future the models produced for the documentation will be used for identifying and documenting fragments. Such fragments will be reused and integrated so as to provide new ways of developing agent-oriented systems.

REFERENCES

[1] 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.

[2] I. Sommerville, *Software Engineering 8th Edition*. Addison-Wesley, 2007.

[3] P. Cuesta, A. Gómez, J. González, and F. J. Rodríguez, "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, pp. 87–98.

[4] S. A. O'Malley and S. A. DeLoach, "Determining when to use an agent-oriented software engineering pradigm," in *Agent-Oriented Software Engineering. Second Int. Workshop, AOSE 2001*, ser. Lecture Notes in Computer Science, M. Wooldridge, G. Weiss, and P. Ciancarini, Eds. Springer-Verlag, 2002, vol. 2222.

[5] C. Bernon, M. Cossentino, and J. Pavón, "Agent-oriented software engineering," *Knowl. Eng. Rev.*, vol. 20, no. 2, pp. 99–116, 2005.

[6] J. Pavón and J. Gómez-Sanz, "Agent Oriented Software Engineering with INGENIAS," *Multi-Agent Systems and Applications III*, vol. 2691, pp. 394–403, 2003.

[7] A. Mas, *Agentes Software y Sistemas Multi-Agentes*. Pearson Prentice Hall, 2004.

[8] S. Brinkkemper, K. Lyytinen, and R. Welke, *Method engineering: Principles of method construction and tool support*. Kluwer Academic Publishers, 1996.

[9] J. Ralyté and C. Rolland, "An approach for method reengineering," in *Conceptual Modeling*. London, UK: Springer-Verlag, 2001, pp. 471–484, 20th International Conference (ER 2001), Yokohama, Japan, 27-30 Nov. 2001. Proceedings. [Online]. Available: http://www.springerlink.com/content/pbtbr52cwya7qyd4/

[10] V. Seidita, M. Cossentino, V. Hilaire, N. Gaud, S. Galland, A. Koukam, and S. Gaglio, "The Metamodel: a Starting Point for Design Processes Construction," *International Journal of Software Engineering and Knowledge Engineering*, 2009, in-press.

[11] B. Henderson-Sellers and C. Gonzalez-Perez, "A comparison of four process metamodels and the creation of a new generic standard," *Information & Software Technology*, vol. 47, no. 1, pp. 49–65, 2005.

[12] A. Omicini, "SODA: Societies and infrastructures in the analysis and design of agent-based systems," in *Agent-Oriented Software Engineering*, ser. LNCS, P. Ciancarini and M. J. Wooldridge, Eds. Springer, 2001, vol. 1957, pp. 185–193, 1st International Workshop (AOSE 2000), Limerick, Ireland, 10 Jun. 2000. Revised Papers.

[13] A. Molesini, E. Nardini, E. Denti, and A. Omicini, "Situated process engineering for integrating processes from methodologies to infrastructures," in *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, S. Y. Shin, S. Ossowski, R. Menezes, and M. Viroli, Eds., vol. II. Honolulu, Hawai'i, USA: ACM, 8–12 Mar. 2009, pp. 699–706. [Online]. Available: http://portal.acm.org/citation.cfm?id=1529282.1529429

[14] O. M. G. OMG, "Software Process Engineering Metamodel Specification. Version 1.1, formal/05-01-06," http://www.omg.org/ (accessed on September 5, 2008), 2005.

[15] V. Seidita, M. Cossentino, and S. Gaglio, "Using and extending the spem specifications to represent agent oriented methodologies," in *AOSE*, ser. Lecture Notes in Computer Science, M. Luck and J. J. Gómez-Sanz, Eds., vol. 5386. Springer, 2009, pp. 46–59, 9th International Workshop, AOSE 2008, Estoril, Portugal, May 12-13, 2008, Revised Selected Papers.

[16] L. Cernuzzi, M. Cossentino, and F. Zambonelli, "Process models for agent-based development," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, pp. 205–222, 2005.

[17] SODA, "Home page," `http://soda.apice.unibo.it`. [Online]. Available: http://soda.apice.unibo.it

[18] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the A&A meta-model for multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 432–456, Dec. 2008, special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems. [Online]. Available: http://www.springerlink.com/content/l2051h377k2plk07/

[19] Object Management Group, "Software & Systems Process Engineering Meta-Model Specification 2.0," `http://www.omg.org/spec/SPEM/2.0/PDF`, Apr. 2008.

[20] M. Cossentino, "From requirements to code with the PASSI methodology," in *Agent Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds. Hershey, PA, USA: Idea Group Publishing, Jun. 2005, ch. IV, pp. 79–106. [Online]. Available: http://www.idea-group.com/books/details.asp?id=4931

[21] J. Pavòn, J. J. Gòmez-Sanz, and R. Fuentes, "The INGENIAS methodology and tools," in *Agent Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds. Hershey, PA, USA: Idea Group Publishing, Jun. 2005, ch. IX, pp. 236–276. [Online]. Available: http://www.idea-group.com/books/details.asp?id=4931

[22] B. Henderson-Sellers and P. Giorgini, Eds., *Agent Oriented Methodologies*. Hershey, PA, USA: Idea Group Publishing, Jun. 2005. [Online]. Available: http://www.idea-group.com/books/details.asp?id=4931