Fragmenting Prometheus: new choices for developing multiagent systems

Daniela S. Yassuda, Sara J. Casare, Anarosa A. F. Brandão

Laboratório de Técnicas Inteligentes
Escola Politécnica – Universidade de São Paulo (USP).

{daniela.yassuda, anarosa.brandao}@usp.br, sjcasare@uol.com.br

Abstract. *The Medee Method Framework supports the development of multiagent systems based on a Situational Method Engineering approach. It involves a repository that stores method fragments which can be used to compose specific methods to each particular project situation. This work consists in fragmenting the agent-oriented method Prometheus following the Medee approach. In this paper, we present a brief explanation of the repository structure and the step-by-step process of fragmenting methods using Medee. Then, we illustrate how this process was applied to define Prometheus fragments: first, by describing how the method was modeled in SPEM elements; second, by discussing the decisions made in order to outline standardized activity, phase and process fragments.*

1. Introduction

The availability of tools, frameworks and methods to assist the development process is one of the factors that impact the adoption of multiagent systems by the software industry. While the multiagent paradigm is a fitting solution for various problems, the supporting assets for the design and implementation of said projects are still evolving.

The Medee Method Framework [2] aims to provide this kind of support for developers who work with multiagent systems. It allows them to build customized methods on demand according to their project characteristics. This is based on Situational Method Engineering concepts, which is a section of Method Engineering [1] that focuses on these tailored solutions to each given situation. They are called situational methods. In order to achieve that, Medee stores pieces of agent-oriented methods, such as Tropos [4] and Gaia [9], as well as organizational models, like MOISE+ [6]. Each one of these pieces is called a fragment. Fragments are standardized and represent a coherent part of a method that can be used and re-used to solve entire problems or parts of them.

The scope of this paper is the fragmentation of Prometheus [8] following the Medee Delivery Process, which is part of the Medee Method Framework. It consists of an initial step towards the use of Prometheus in situational approaches for developing multiagent systems. The next section will further detail the Medee Repository and explain the Delivery Process. Then, the fragmentation itself will be described by indicating which tools and resources were used, how Prometheus tasks and work products were depicted and discussing the aspects taken in consideration while outlining the fragments.

2. Medee Method Framework

The Medee Method Framework is composed of a method repository, a composition model and a delivery process. Here, we briefly explain the Medee Repository and the Medee Delivery Process, which is the process to guide populating the repository.

2.1 Medee Method Repository

The Medee Method Repository is constituted by three layers: Medee Elements Pillar, which stores method elements and methods "as is"; Medee Fragments Pillar, storing the fragments themselves; and Medee Methods Pillar, that keeps Medee AOSE Methods and Medee Situational Methods.

The first layer is the Elements Pillar, the foundation of the repository. It keeps method elements, which are SPEM elements – tasks, roles, work products, guidance and categories - that synthetize the content of each method or organizational model. Using said elements, it is possible to describe the entire methods with SPEM notation: these are called Methods As Is, also stored in this pillar.

The second layer stores the method fragments. The elements of the first pillar go through a standardization process, then they are assembled to build fragments. In Medee, fragments can cover five different disciplines - Requirements, Analysis, Design, Implementation and Test – and they have four different granularities – Activity, Phase, Iteration and Process.

The Medee Methods Pillar stores two kinds of methods: AOSE Methods, which are complete agent-oriented methods, and Situational Methods, which are custom-made according to project factors specified by the user. All these methods are built using Medee Fragments from the second pillar.

2.2 Medee Delivery Process

The Medee Delivery Process is the step-by-step specification on how to obtain method elements, method fragments and situational methods by populating all three pillars of the repository. In order to fragment Prometheus, we followed the first two phases of the Medee Delivery Process: Medee Method Element Capture (MMEC) Phase and Medee Method Fragment Elaboration (MMFE) Phase.

During the first phase, which aims to capture method elements to fill the first pillar of the repository, the agent-oriented method is studied, analyzed and modeled in terms of SPEM elements. All content is outlined and then detailed in form of tasks, work products, roles, guidance and categories. These method elements are used to build the activities and phases that constitute the Methods As Is, which are full methods represented in SPEM elements.

The Medee Method Fragment Elaboration Phase involves the fragmentation itself. It is necessary to standardize the method elements from the first pillar before using them to build fragments. The concern regarding making the elements and fragments uniform is due to the reusability aspect of the framework. It is expected that the fragments can be reused and recombined with fragments sourced from other methods. After that, the fragmentation starts from the smaller granularity to the bigger one. First we obtain Activity fragments, then Phase and Iteration, and, lastly, a Process fragment.

The Delivery Process has a third phase, the Medee Situational Method Composition Phase, that discusses how to build situational methods with Medee using project factors and either a top-down or a bottom-up fragment composition approach, but it is out of the scope of the paper. An interested reader can find more information in [2].

3. Fragmenting the Prometheus method

The fragmentation process is described through the two phases of the Medee Delivery Process that were performed during this work: the capture of Prometheus method content in terms of SPEM elements and the elaboration of the fragments themselves. We present a brief introduction to Prometheus, the tools used during the process and considerations about the decisions made to get the fragments. Due to space limitations, we will describe only one fragment of each discipline or component.

3.1 First layer – Capturing Method Elements

The Medee Delivery Process starts by understanding the agent-oriented method that will be fragmented. Prometheus is a method that focuses on supporting the development of goal and plan-oriented agents, though it is not limited to this approach. It is intended to be a practical method, providing detailed steps and guidance to help perform each task. Prometheus has three phases: the System Specification Phase, the Architectural Design Phase and the Detailed Design Phase.

After studying the method, the next step is to outline the method content using SPEM elements, with the tool support of the Eclipse Process Framework (EPF) Composer [4], which allows the authoring of methods with various development cycles and applications. We start the MMEC Phase capturing the method elements by modeling Prometheus tasks, roles, work products and guidance with EPF Composer, first by just sketching them out and later by detailing their descriptions and setting up all the links between them. With these method elements, it was possible to assemble the entire method together with its phases, activities and tasks, therefore mounting the Prometheus As Is method. Figure 1 shows the thirteen tasks and twenty-one work products captured.

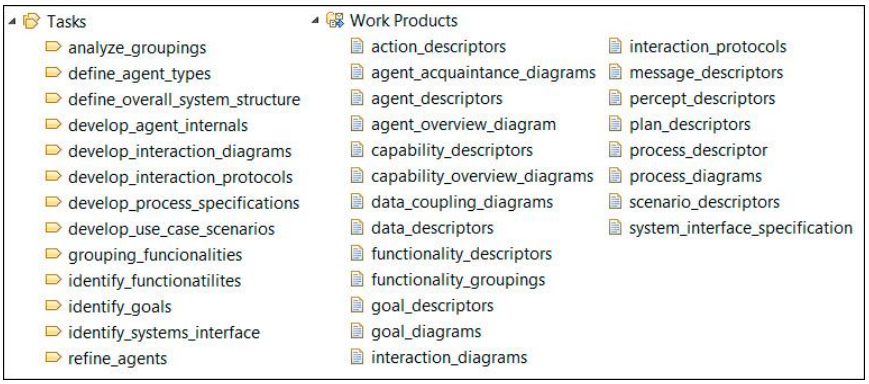

Figure 1. Prometheus tasks and work products captured during MMEC Phase.

The System Specification Phase in Prometheus As Is has four activities: *Identify Goals*, *Identify Functionalities*, *Identify System's Interface* and *Develop Use Case Scenarios*. The Architectural Design Phase contains four activities as well. There are three activities related to the definition of agent types that will exist within the system:

Grouping Functionalities, *Analyze Groupings* and *Define Agent Types*. During this phase, there is also the *Define Overall System Structure* activity and the *Develop Interaction Specification* activity. The last phase, Detailed Design, contains the following activities: *Develop Agent Internals*, *Develop Process Specifications* and *Refine Agents*. The activities involving agents include the design of its internals by using capabilities and by developing plans.

Most of the resulting work products from each task are either descriptors or diagrams. The guidance includes concepts, descriptors templates, diagram examples and a tool mentor for the Prometheus Design Tool (PDT). As for the roles, we chose to already use the Medee Roles, since they fit the specifications of Prometheus.

3.2 Second layer – Elaborating Method Fragments

The next phase of the Delivery Process, the MMFE Phase, concerns the elaboration of fragments. We use the method elements captured previously as a foundation. While crafting fragments, it is possible to extend and revise each element, as well as creating new elements if the fragmentation calls for so. The Prometheus As Is method can be used as a reference to elaborate fragments and check for consistencies.

The tasks and work products have to go through a standardization process, so all the content stored in Medee can be uniform and follow the same guidelines. In Medee Framework, all task names start with MTV (Medee Task Variability) and work product names start with MPV (Medee Product Variability). Fragment names, as seen further in this sub-section, start with MMF (Medee Method Fragment) and explicitly state its source method name. Since we already used the Medee Role Set during the first phase, there is no need to change them.

We start with the activity fragments. Then, we use these fragments to build phase fragments and finish by building a process fragment. It is necessary to have in mind that Medee Method Fragments have particular classifications that aim to ease the situational method composition process. They can cover one of the five process disciplines: Requirements, Analysis, Design, Implementation and Test. Regarding the MAS component they focus on, Medee has four categories, which are Agent, Interaction, Environment and Organization. Prometheus covers the Requirements, Analysis and Design disciplines, as well as Agent, Interaction and Environment components.

From the System Specification Phase of Prometheus, fragments of Requirements and Analysis disciplines are obtained. An example is MMF Identify Requirements with Prometheus. The activity consists of one task, MTV Identify Goals, whose aim is to describe the system's goals and sub-goals using descriptors and diagrams. These are encapsulated in one work product, MPV Goal Specification. Since descriptors and diagrams of the same subject complete each other, they were grouped together to facilitate the understanding of the task.

An important characteristic of Prometheus is that the developer is encouraged not to perform tasks sequentially. Instead, the method proposes to keep refining work products in an iterative manner, always using the knowledge and suggestions obtained while performing other tasks. This aspect is noticeable in this fragment: the task has multiple optional inputs, to prompt the developer to look for cues in other work products (Figure 2).

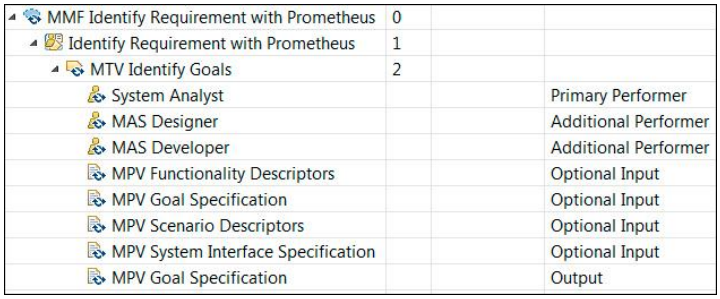

Figure 2. Consolidated view of MMF Identify Requirements with Prometheus.

The MMF Analyze Agent with Prometheus includes the System Specification task MTV Identify Functionalities, as well as two tasks sourced from the original Architectural Design Phase, MTV Group Functionalities and MTV Analyze Groupings. Since the identification of system's functionalities involves the analysis of the goal specification and also takes in consideration information from the use case scenario, this task was placed in the Analysis discipline instead of the Requirements one. The functionalities are also used to define the agent types that will be developed. This is done during the MTV Group Functionalities task, whose outcomes are diagrams that depict various possible designs. Said designs are evaluated during MTV Analyze Groupings, resulting in a list of agent types, each one encapsulating a set of functionalities.

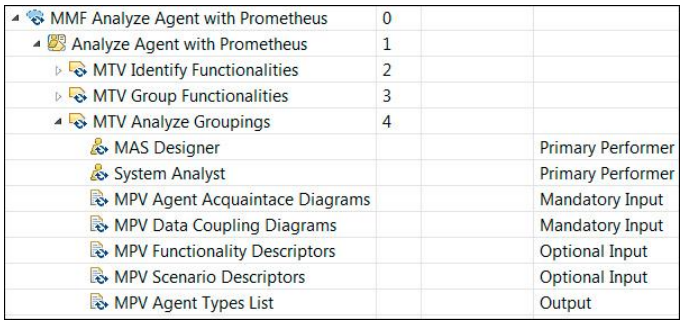

Figure 3. Consolidated view of MMF Analyze Agent with Prometheus.

During the MMF Design Interaction with Prometheus, the MAS Designer takes the use case scenarios from the Requirements phase to scheme the interaction between agents in form of MPV Interaction Protocols, MPV Interaction Diagrams and MPV Message Descriptors. These three work products contribute to the same specification, so they are grouped together in the MPV Interaction Design artifact.

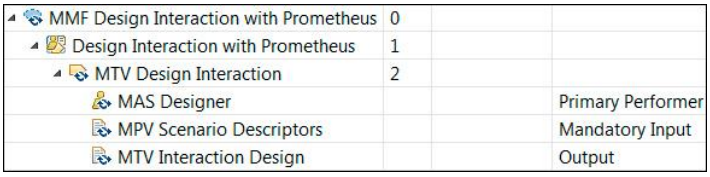

Figure 4. Work breakdown of Design activity fragments.

4. Conclusion

The described process resulted in twelve tasks, thirty-one work products, twenty-five guidance, seven activity fragments, three phase fragments and one process fragment

sourced from Prometheus. For now on, these fragments can be used to compose situational methods for multiagent systems. The fragments related to the Design discipline and Agent component are the most detailed of them, which is coherent with the characteristics of Prometheus. Due to limited space, it was not possible to display in detail all fragments with their respective tasks, inputs, outputs, assigned roles and guidance. However, they will be available in the Medee website[1]. The next step is to evaluate the fragments using the Medee Improvement Cycle [3] which allows to continuously improving fragments by submitting them to experts' assessment. In this case, we can involve Prometheus experts, MAS developers and Method engineers. Besides that, the practical use of the fragments in project situations will tell how they can be improved.

Acknowledgements

Daniela Yassuda is partially supported by grant #013/14642-4, São Paulo Research Foundation (FAPESP). Anarosa A. F. Brandão is partially supported by grant #010/2640-5, São Paulo Research Foundation (FAPESP).

References

1. Brinkkemper, S. 1996. Method Engineering: Engineering of Information Systems Development Methods and Tools. Information and Software Technology, vol. 38 (4), 275-280
2. Casare, S. J. ; Brandão, Anarosa A.F.; Guessoum, Z. ; Sichman, J.S. 2013. Medee Method Framework: a Situational Approach for Organization-Centered MAS. Autonomous Agents and Multi-Agent Systems (Dordrecht. Online), v. tbd, p. 1-44, 2013. (http://dx.doi.org/10.1007/s10458-013-9228-y).
3. Casare, S.J. 2012. *Medee: A Method Framework for Multiagent Systems*, PhD Thesis, Universidade de São Paulo, Brazil.
4. Giorgini, P. et al. 2004. The Tropos Methodology. In: Bergenti, V; Gleizes, M. P.; Zambonelli, F.(Ed.), *Methodologies and software engineering for agent systems*, Kluwer Academic Publishers, pp. 89-106.
5. Haumer, P. 2007. Eclipse Process Framework Composer – Part 1 – Key Concepts. Available at: <http://www.eclipse.org/epf>.
6. Hubner J., Sichman J., Boissier O. 2002. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: Bittencourt, G. & Ramalho, G. L. (Eds.), *16th Brazilian Symposium on AI, SBIA'02*, LNAI 2507, Berlin: Springer, p. 118-128
7. OMG. (2008). Object Management Group. Software & Systems Process Engineering Meta-Model Specification, version 2.0. OMG document number: formal/2008-04-01. http://www.omg.org/spec/SPEM/2.0/PDF.
8. Winikoff, M. & Padgham, L. (2004) The Prometheus Methodology, In Bergenti,F. Gleizes, M-P, Zambonelli, F. (Eds) Methodologies and Software Engineering for Agent Systems - The Agent-Oriented Software Engineering Handbook, Kluwer Acad. Publ.
9. Zambonelli F., Jennings N. R., Wooldridge M. 2003. Developing multiagent systems: The Gaia methodology. *ACM Transaction on Software Engineering and Methodology*, vol 12(3), 417-470

[1] http://medee.poli.usp.br/