Using Method Engineering for the Construction of Agent-Oriented Methodologies

Giancarlo Fortino, Alfredo Garro, and Wilma Russo
D.E.I.S.
Università della Calabria
Via P. Bucci, 87030 Rende (CS), Italy
Email: {fortino, garro, russow}@deis.unical.it

***Abstract*— Great emphasis has been recently given to agent-oriented methodologies for the construction of complex software systems. In this paper two approaches for the construction of agent-oriented methodologies and based on methods integration are presented: *meta-model-driven* and *development process-driven*. The former is based on the MAS meta-model adopted by designers for the development of a MAS for a specific problem in a specific application domain. The latter is based on the instantiation of a software development process in which each phase is carried out using appropriate method fragments and by the mutual adaptation of the work products coming out from each phase.**

I. Introduction

In analysing and building complex software systems, a number of fundamental techniques for helping to manage complexity have been devised [3]:

- *Decomposition*: the basic technique for tackling large problems by dividing them into smaller, more manageable chunks, each of which can then be approached in relative isolation. It helps tackling complexity because it limits the designer's scope.
- *Abstraction*: the process of defining a simplified model of the system that emphasizes some details or properties, while suppressing others. It is useful because it limits the designer's scope of interest at a given time.
- *Organization*: the process of defining and managing the interrelationships between the various system's components. The ability to specify organizational relationships helps tackling complexity by enabling a number of basic components to be grouped together and treated as a higher-level unit of analysis, and by providing a means of describing the high-level relationships between the various units.

Recently the agent-oriented approach [13] has been widely recognized as very suitable for the development of complex software systems since it fully exploits the techniques listed above. In particular in the context of complex software systems:

- the agent-oriented decompositions are an effective way of partitioning the problem space;
- the key abstractions of the agent-oriented mindset (agents, interactions, and organizations) are a natural means of modelling;
- the agent-oriented philosophy for modelling and managing organizational relationships is appropriate for dealing with the existing dependencies and interactions.

The development of complex software systems by using the agent-oriented approach requires suitable agent-oriented modelling techniques and methodologies which provide explicit support for the key abstractions of the agent paradigm.
Several methodologies supporting analysis, design and implementation of Multi-Agent Systems (MAS) have been to date proposed in the context of Agent Oriented Software Engineering (AOSE) [14]. Some of the emerging methodologies are Gaia [16], MaSE [7], Prometheus [15], Tropos [4], Message [5], Passi [6], and Adelfe [2]. Although such methodologies have different advantages when applied to specific problems it seems to be widely accepted that an unique methodology cannot be general enough to be useful to everyone without some level of customization. In fact, agent designers, for solving specific problems in a specific application context, often prefer to define their own methodology specifically tailored for their needs instead of reusing an existing one. Thus, an approach that combines the designer's need of defining his own method-ology with the advantages and the experiences coming from the existing and documented methodologies is highly required.
A possible solution to this problem is to adopt the method engineering paradigm so enabling designers of MAS to use phases or models or elements coming from different methodologies in order to build up a customized approach for their own problems [12].
In particular, the development methodology is constructed by assembling pieces of methodologies (**method fragments**) from a repository of methods (**method base**). The method base is built up by taking method fragments coming from existing agent-oriented methodologies (such as Adelfe, Gaia, Message, Passi, Tropos, etc.) or ad hoc defined methods. Currently this approach is adopted by the FIPA Methodology Technical Committee (TC) [20].
It is therefore crucial to define guidelines for methods integration in order to both construct the methodology (retrieving the method fragments from the method base and integrating them) and apply it in the actual development life cycle.
In this direction, the paper proposes two approaches for

the construction of agent-oriented methodologies by using methods integration: (i) *meta-model-driven*, which is based on the MAS meta-model adopted by the designer for the development of a MAS for a specific problem in a specific application domain; (ii) *development process-driven*, which is based on the instantiation of a software development process in which each phase is carried out using appropriate method fragments.

The remainder of this paper is organized as follows. In section II and III the meta-model-driven and the development process-driven approaches are respectively described. In section IV, conclusions are drawn and on-going research activities delineated.

II. The MAS meta-model-driven approach

A method fragment [18] is a portion of methodology which is composed of the following parts:

1) A process specification, defined with a SPEM diagram [21], which defines the procedural aspect of the fragment;
2) One or more deliverables such as AUML/UML diagrams and text documents [1];
3) Some preconditions which represent a kind of constraint since it is not possible to start the process specified in the fragment without the required input data or without verifying the required guard conditions;
4) A list of elements (which is a part of the MAS meta-model subsumed by the methodology from which it was extracted) to be defined or refined through the specified process;
5) Application guidelines that illustrate how to apply the fragment and related best practices;
6) A glossary of terms used in the fragment in order to avoid misunderstandings if the fragment is reused in a context that is different from the original one;
7) Composition guidelines which describe the context/problem that is behind the methodology from which the specific fragment is extracted;
8) Aspects of fragment which are textual descriptions of specific issues such as platform to be used, application area, etc;
9) Dependency relationships useful to assemble fragments.

It should be noted that not all of these elements are mandatory; some of them (for instance notation or guidelines) could be not applicable or not necessary for some specific fragment.

To build his own methodology by exploiting the *meta-model-driven* approach, the designer must:

- choose or define a MAS meta-model suitable for the specific problem and/or the specific application domain;
- identify the elements that compose the meta-model of the MAS under development;
- choose the method fragments that are able to produce the identified meta-model elements;
- defining a development process characterized by a *method fragments execution order* on the basis of the relationship existing among the meta-model elements produced by each fragment.

Hence, the obtained methodology is able to completely *cover* the MAS meta-model for a given problem in a specific application domain.

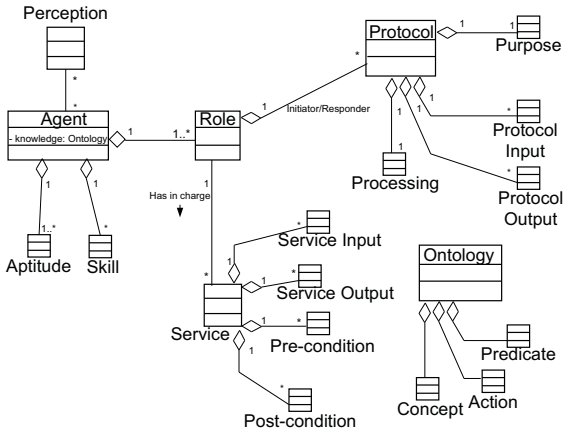

Fig. 1. An example MAS meta-model

An example MAS meta model is reported in Figure 1. Referring to the MAS meta-model of Adelfe, Gaia and Passi a set of methods fragments that are able to produce a piece of the MAS meta-model can be chosen. To completely cover the MAS meta-model selected fragments can be combined and, if necessary, new fragments can be defined (see Figure 2). Using this approach, the integration among the fragments is based on the relationships existing among the elements of the MAS meta-model. Thus, in order to obtain a completely and well-defined ad-hoc methodology, a proper *method fragments execution order* is to be defined.

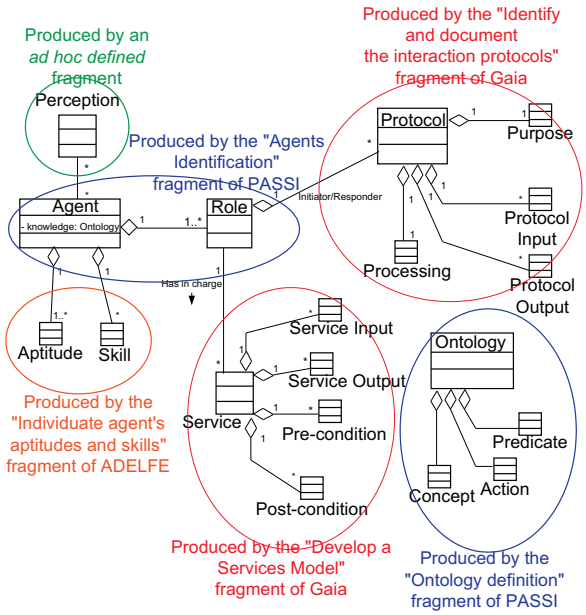

Fig. 2. An example of meta-model-driven methods integration

On the basis of the relationships shown in figure 2) the method fragments execution order is the following:

1) the Agents Identification fragment of Passi [19];
2) the concurrent execution of the ad-hoc defined fragment and the Individuate agent's aptitudes and skills fragment of Adelfe [17];
3) the concurrent execution of the Develop a Services Model fragment of Gaia and the Identify and document the interaction protocols fragment of Gaia [11];
4) the Ontology definition fragment of Passi [19].

III. The development process-driven approach

The development process-driven approach focuses on the instantiation of a software development process that completely covers the development of MAS (see Figure 3).

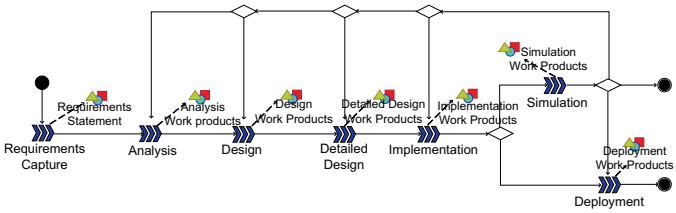

Fig. 3. An example of software development process

To build his own methodology by exploiting the *development process-driven* approach, the designer must:

- choose or define a software development process suitable for the specific problem and for the specific application domain;
- instantiate the development process by selecting, for each phase, suitable method fragments, chosen from agent-oriented methodologies proposed in the literature or ad-hoc defined.

An example software development process [8] is reported in Figure 3. Referring to the development phases specified by Tropos, Gaia, Passi and by a Statecharts-based methodology [10], a set of methods fragments that are able to carry out each phase of the development process are to be chosen. To completely cover the development process the selected fragments can be combined and, if necessary, new fragments can be defined (see Figure 4). Using this approach, the integration between the fragments is achieved by individuating and/or defining dependencies among work products produced by the fragments of the instantiated process. Notice that the work products produced in a given fragment might constitute the input for the next fragment provided that they contain all the in-formation required to its initialization (see Figure 5).

IV. Conclusions

This paper has proposed two approaches to the integration of methods fragments: *meta-model-driven* and *development process-driven*. These approaches are not mutually exclusive; rather, hybrid approaches containing features of both of them might be defined as well.

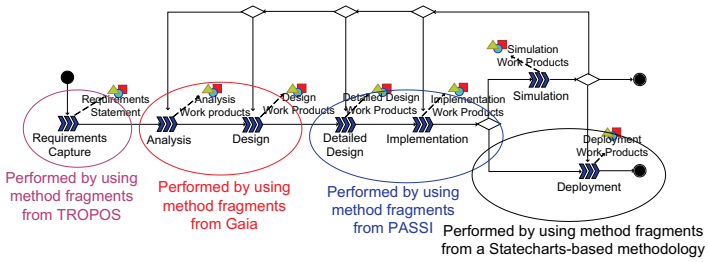

Fig. 4. Development process-driven methods integration

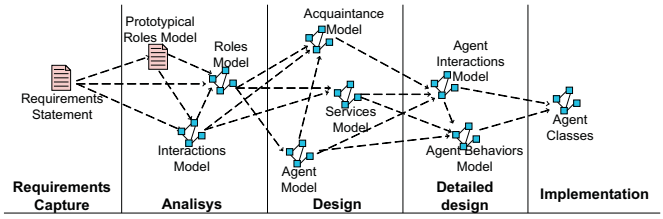

Fig. 5. Dependencies among work products of the instantiated process

The *meta-model-driven* approach provides the following advantage: flexibility for the definition of methodologies and meta-models of the MAS to be developed. Conversely, it has some drawbacks: (i) difficulty of integration of different fragments due to different semantics of the meta-model concepts; (ii) selection and/or definition of the meta-model to adopt for the specific problem and/or application domain.
The *development process-driven* approach is characterized by the following advantages: flexibility for the construction of methodologies by means of the instantiation of each stage of the development process. On the other hand, the disadvantages are the following: (i) process rigidity; (ii) low flexibility of the system meta-model since the meta-model of the adopted methodology must be used; (iii) adaptation among the work products which is sometimes difficult to achieve; (iv) choice and definition of the process to instantiate for the specific problem and/or application context. On going research activity is being focused on:

1) definition of adaptation techniques among work products produced by different methods and/or method fragments;
2) extraction from and definition of method fragments of already existing methodologies and the mutual adaptation among the defined method fragments. This activity is being carried out in the context of the FIPA Methodology TC;
3) the experimentation of the two presented approaches for the e-Commerce application domain [9].

References

[1] B. Bauer, J.P. Muller, and J. Odell. Agent UML: A Formalism for Specifying Multiagent Interaction. In Paolo Ciancarini and Michael Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 91–103. Springer-Verlag, Berlin, 2001.

[2] C. Bernon., M.P. Gleizes, G. Picard, and P. Glize. The Adelfe Methodology For an Intranet System Design. In *Proc. of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS)*, Toronto, Canada, 2002.

[3] G. Booch. Object-Oriented Analysis and Design with Applications. Addison Wesley, 1994.

[4] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology, *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[5] G. Caire, F. Leal, P. Chainho, R. Evans, F. Garijo, J. Gomez, J. Pavon, P. Kearney, J. Stark, and P. Massonet. Agent Oriented Analysis using MESSAGE/UML. In *Proc. of the 2nd In-ternational Workshop on Agent-Oriented Software Engineering (AOSE)*, LNCS 2222. Springer-Verlag, Berlin, 2002.

[6] M. Cossentino, P. Burrafato, S. Lombardo, and L. Sabatucci. Introducing Pattern Reuse in the Design of Multi-Agent Systems. In Ryszard Kowalczyk, Jorg P. Muller, Huaglory Tianfield, Rainer Unland, editors, *Agent Technologies, Infrastructures, Tools, and Applications for E-Services* , Lecture Notes in Artificial Intelligence (LNAI) volume 2592, pages 107–120, Springer-Verlag, Berlin Heidelberg, Germany, 2003.

[7] S. A. DeLoach, M. Wood, and C. Sparkman. Multiagent system engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):231–258, April 2001.

[8] G. Fortino, A. Garro, and W. Russo. From Modeling to Simulation of Multi-Agent Systems: an integrated approach and a case study. In Gabriela Lindemann, Jorg Denzinger, Ingo J. Timm, Rainer Unland, editors, *Multiagent System Technologies*, Lecture Notes in Artificial Intelligence (LNAI) volume 3187, pages 213–227, Springer-Verlag, Berlin Heidelberg, Germany, 2004.

[9] G. Fortino, A. Garro, and W. Russo. Modelling and Analysis of Agent-Based Electronic Marketplaces. *IPSI Transactions on Advanced Research*, 2004, to appear.

[10] G. Fortino, W. Russo, and E. Zimeo. A Statecharts-based Software Development Process for Mobile Agents. *Information and Software Technology*, 46(13):907–921, 2004.

[11] A. Garro, P. Turci, and M.P. Huget. Expressing Gaia Methodology using Spem. *FIPA Methodology TC, working draft v. 1.0/04-03-15, [http://fipa.org/activities/methodology.html]*.

[12] B. Henderson-Sellers. Method Engineering for OO Systems Development. *Communications of the ACM*, 46(10), 2003.

[13] N. R. Jennings. An Agent-Based Approach for Building Complex Software Systems. *Communications of the ACM*, 44(4), 2001.

[14] J. Lind. Issues in Agent-Oriented Software Engineering. In *Proc. of the First International Workshop on Agent-Oriented Software Engineering (AOSE)*, LNCS 1957, pages 45–58. Springer-Verlag, Berlin, 2001.

[15] L. Padgham and M. Winikoff. Prometheus: A methodology for developing intelligent agents. In *Proc. of the Third International Workshop on Agent-Oriented Software Engineering (AOSE)*, LNCS 2585, Springer-Verlag, Berlin, 2003.

[16] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[17] M. P. Gleizes et al. Adelfe fragments, rel.0, March 2004. *[http://www.pa.icar.cnr.it/ cossentino/FIPAmeth/docs/adelfe_fragments_v0.pdf]*

[18] Method Fragment Definition. *FIPA Methodology TC, working draft, Nov. 2003, [http://fipa.org/activities/methodology.html]*.

[19] M. Cossentino. PASSI fragments: All fragments, draft. rel 0.1, Feb. 2004. *[http://www.pa.icar.cnr.it/ cossentino/FIPAmeth/docs/passi_fragments_0_1.zip]*

[20] Foundation for Intelligent Physical Agents (FIPA) Specifications. *[http://www.fipa.org]*.

[21] Software Process Engineering Metamodel Specification, Version 1.0, formal/02-11-14. Object Management Group Inc. , November 2002.