A Domain Engineering Process for Developing Multi-agent Systems Product Lines

(Extended Abstract)

Ingrid Nunes
PUC-Rio, LES
Rio de Janeiro, Brazil
ioliveira@inf.puc-rio.br

Uirá Kulesza
Federal University of Rio Grande do Norte (UFRN)
Natal, Brazil
uira@dimap.ufrn.br

Camila Nunes, Carlos J.P. Lucena
PUC-Rio, LES
Rio de Janeiro, Brazil
{cnunes,lucena}@inf.puc-rio.br

ABSTRACT

Multi-agent Systems Product Lines (MAS-PLs) have emerged to integrate two promising trends of software engineering: agent-oriented software engineering and software product lines. In this paper, we propose a domain engineering process to develop MAS-PLs, built on top of agent-oriented and software product line approaches.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; D.2.8 [**Software Engineering**]: Reusable Software—*Domain engineering*

General Terms

Design, Documentation

Keywords

Software Product Lines, Multi-agent Systems, Process, Agent-oriented software engineering, Domain Engineering

1. INTRODUCTION

Agent-oriented software engineering has emerged as a new software engineering paradigm to help on the development of complex and distributed systems, based on agent abstraction. Several works were proposed in this direction, such as methodologies and modeling languages. However, most of agent-oriented methodologies do not take into account the adoption of extensive reuse practices, which have been widely used in the software engineering context to provide reduced time-to-marked and lower development costs.

In the context of software reuse, the concepts of system families and software product lines (SPLs) have gained a significant popularity throughout the software industry and research community, leading to the emergence of a new field called software product line engineering (SPLE). Although many SPL methodologies have been proposed, they do not detail or barely detail the modeling and documentation of SPLs that take advantage of agent technology. Only some recent research [3, 9] has explored the integration of SPL and Multi-agent System (MAS) technologies, by incorporating their respective benefits and helping the industrial exploitation of agent technology.

This paper presents a domain engineering process for developing MAS-PLs. Our approach is the result of an investigation of how current SPL, MAS and MAS-PL approaches can model MAS-PLs. Based on this experience, we propose our process, built on top of some MAS and SPL approaches.

Cite as: A Domain Engineering Process for Developing Multi-agent Systems Product Lines (Short Paper), Ingrid Nunes, Uirá Kulesza, Camila Nunes and Carlos J.P. Lucena, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

2. THE DOMAIN ENGINEERING PROCESS

SPLE is in general organized in two main processes: domain engineering and application engineering. Domain engineering is the process of SPLE in which the commonalities and variabilities of the SPL are identified, defined and realized. Our proposal is a domain engineering process that defines the stages and their respective activities to develop MAS-PLs. It aggregates some activities that are specific to model software agents and their variabilities. Table 1 summarizes the stages and activities that compose our process, and the artifacts produced in each one of them as well.

Our domain engineering process was conceived by the integration of existing works in the context of MASs and SPLs, which are: (i) PLUS [4] method; (ii) PASSI [1] methodology; and (iii) MAS-ML [2] modeling language. In addition, we propose additional adaptations and extensions for them.

We have investigated the use of current SPLs and MAS-PLs approaches for modeling and documenting agency features in MAS-PLs [7]. SPL methodologies provide useful notations to model the agency features. However, none of them completely covers their specification. Particularly, the PLUS (Product Line UML-based Software engineering) [4] approach was very useful for documenting agency features. PLUS provides a set of concepts and techniques to extend UML-based design methods and processes for single systems to handle SPLs. As a consequence, this approach was the base for the elaboration of our process.

Nevertheless, agent technology provides particular characteristics that need to be considered in order to take advantage of this paradigm. In order to model agency features in MAS-PL, we have adopted the phases of the System Requirements model of PASSI [1] methodology, which is agent-oriented. We made some adaptations in these phases

Stage	Activity		Artifacts
Domain Analysis	Early Requirements		
		Feature Modeling	Feature Model
		Use Case Modeling	Use Case Diagram, Use Case Descriptions
		Feature/Use Case Dependency Modeling	Feature/Use Case Dependency Model
	Late Requirements		
		Agent Identification	Agent Identification Diagram
		Role Identification	Role Identification Diagram
		Task Specification	Task Specification Diagram
		Feature/Agent Dependency Modeling	Feature/Agent Dependency Model
Domain Design	Static Modeling		Class Diagrams, Role Diagrams, Organization Diagrams
	Dynamic Modeling		Sequence Diagrams
	Feature/Agent Dependency Modeling		Refined Feature/Agent Dependency Model
Domain Realization	Assets Implementation		Reusable Software Elements
	Design/Implementation Elements Mapping		Implementation Model, Configuration Model

Table 1: The Domain Engineering Process.

and they were incorporated into the domain analysis stage. The incorporated phases with our extensions are described in Table 2. PASSI follows the guideline of using standards whenever possible; and this justifies the use of UML as modeling language. This helped us to incorporate some PLUS notations into PASSI diagrams.

Table 2: Our extensions to the PASSI Approach.

Phase	Extensions
Agent Identification	Only use cases in <<agency feature>> stereotyped packages are distributed among agents Use of Stereotypes (kernel, alternative or optional) Use of colors to trace features
Role Identification	Use of UML 2.0 frames (crosscutting features) Use of colors to trace features
Task Specification	One diagram per agent and feature Use of UML 2.0 frames (crosscutting features) Use of colors to trace features

At the domain design stage, PASSI uses conventional class, sequence and activity UML diagrams to design agents, and this approach has been successfully used in the development of embedded robotics applications. However, our focus is to allow the design of agents that follow the BDI (belief-desire-intention) model. Moreover, some important agent-oriented concepts, such as environment, cannot be modeled with UML and the use of stereotypes is not enough because objects and agency elements have different properties and different relationships. Thus, our process uses the MAS-ML [2] modeling language, with some extensions, to model agents. MAS-ML extends the UML meta-model in order to express specific agent properties and relationships. As discussed in [2], others MAS modeling languages do not allow to model some agency concepts.

To address variability in MAS-ML diagrams, we adopted four different adaptations: (i) use of the <<*kernel*>> , <<*optional*>> and <<*alternative*>> stereotypes to indicate that diagram elements are part of the core architecture, present in just some products or vary from one product to another, respectively; (ii) use of colors to indicate that an element is related to a specific feature; (iii) model each feature in a different diagram, whenever possible. It is not possible to be done when dealing with crosscutting features; however the use of colors helps to distinguish the elements related to these features; and (iv) introduction of the capability [8] concept to allow the modularization of variable parts in agents and roles. Capabilities have been introduced into some MASs as a software engineering mechanism to support modularity and reusability.

3. CONCLUSION AND FUTURE WORK

In this paper, we presented a domain engineering process to develop MAS-PLs. Our approach incorporates activities and notations of different well-succeeded works in the context of SPLs and MASs: PLUS provides notations for documenting variability; PASSI methodology diagrams are used to specify agency features; and MAS-ML is the modeling language used in the domain design stage. We also proposed some adaptations and extensions to these approaches to address agency features. We have developed our process with the experience of two cases studies: the OLIS [5], a product line of web applications that provide personal services to users, and the ExpertCommittee [6], a conference management systems product line.

We are currently working on the development of other case studies to evaluate our process. In addition, we aim at formalizing our process using a process modeling language.

4. REFERENCES

[1] M. Cossentino. *From Requirements to Code with the PASSI Methodology*, chapter IV. Idea Group Inc., 2005.

[2] V. T. da Silva, R. Choren, and C. J. P. de Lucena. A uml based approach for modeling and implementing multi-agent systems. In *AAMAS '04*, 2004.

[3] J. Dehlinger and R. R. Lutz. A Product-Line Requirements Approach to Safe Reuse in Multi-Agent Systems. In *SELMAS'05*, 2005.

[4] H. Gomaa. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison Wesley, USA, 2004.

[5] I. Nunes, U. Kulesza, C. Nunes, E. Cirilo, and C. Lucena. Extending web-based applications to incorporate autonomous behavior. In *WebMedia 2008*, pages 115–122, 2008.

[6] I. Nunes, C. Nunes, U. Kulesza, and C. Lucena. Developing and evolving a multi-agent system product line: An exploratory study. In *AOSE '08*, pages 177–188, 2008.

[7] I. Nunes, C. Nunes, U. Kulesza, and C. Lucena. Documenting and modeling multi-agent systems product lines. In *SEKE '08*, pages 745–751, 2008.

[8] L. Padgham and P. Lambrix. Agent capabilities: Extending bdi theory. In *AAAI '00*, pages 68–73, 2000.

[9] J. Pena, M. G. Hinchey, and A. Ruiz-cortés. Building the core architecture of a nasa multiagent system product line. In *AOSE'06*, 2006.