7
Conclusion and Future Work

On the one hand, SPL is a new trend of software reuse, which investigates methods and techniques in order to promote reduced time-to-market, lower development costs and quality improvement on the development of applications that share common and variable features. On the other hand, AOSE is a new software engineering paradigm to allow the development of distributed complex applications, which are characterized by a system composed of many interrelated subsystems. In order to incorporate the benefits of these two software engineering disciplines and to help the industrial exploitation of agent technology, Multi-agent System Product Lines (MAS-PLs) have emerged integrating Software Product Lines and Agent-oriented Software Engineering techniques.

In this dissertation, we presented a domain engineering process for developing MAS-PLs. We specified our process according to SPEM, defining its phases, activities, tasks, roles and work products. We provided specific activities concerned in analyzing, designing and implementing agent features. Our approach was defined based on existing best practices of existing SPL and MAS approaches: PLUS is a SPL method based on UML that provides notations for documenting variability; PASSI is an agent-oriented methodology, which proposes some diagrams that are used to specify agent features at analysis level; and MAS-ML is the modeling language used in the domain design phase to model agents, roles and organizations. We also proposed adaptations and extensions to these approaches to address variability and traceability of agent features. Additionally, some design and implementation guidelines are proposed in order to help in agent features modularization. An advantage of the approach resides in the fact that we separate the modeling of agent features, and it makes it possible to evolve existing systems to incorporate new features that take advantage of agent abstraction. We have also discussed limitations of our approach, which include techniques to verify consistency among our process models. We developed our process with the experience of two cases studies: the ExpertCommittee case study, which is a product line of conference management systems and the OLIS case study, which is a product line of web

applications that provide several personal services to users. These case studies were modeled and documented with our domain engineering process and were used to evaluate and illustrate our approach.

7.1 Contributions

As the result of the work presented in this dissertation, the following contributions can be enumerated:

SPL and MAS-PL Approaches Comparison. Several SPL approaches have been proposed. In order to understand these approaches and analyze how they can deal with the documentation and modeling of agent features, we studied and compared some existing SPL and MAS-PL approaches (Nunes 2008a). In our comparative study (Chapter 2), we used the same evaluation framework of (Matinlassi 2004) to compare them. The goal of this evaluation was to obtain an overview of these approaches and not necessarily to rate them.

Domain Engineering Process. The main contribution of this dissertation is the definition of a systematic process (Nunes 2009b, Nunes 2009c) to perform domain engineering, which includes the phases of Domain Analysis, Domain Design, and Domain Realization, based on a set of activities, tasks, work products, roles. The domain engineering phases specify: (i) how to analyze, understand and document MAS-PL agent and non-agent features (Domain Analysis) (Nunes 2009d, Nunes 2009e); (ii) how to design a MAS-PL reference architecture by modeling its features (Domain Design); and (iii) how to implement MAS-PLs (Domain Realization). Moreover, the process was specified using SPEM (OMG 2008), which was proposed by OMG. It provides a way of documenting processes so that they may be studied, understood, and evolved in a logical, standardized fashion.

Documenting and Tracing Agent Features. In order to provide agent features documentation and tracing, we integrated notations from SPL and MAS approaches, and incorporated additional extensions. Stereotypes, UML 2.0 frames and structured activities were introduced in agent specific diagrams, such as role identification, task specification and organization diagrams, to provide explicit variability information. Additionally, the capability concept was used to modularize agent and role variabilities. A Feature/Agent dependency model was proposed providing agent features traceability.

Separated Modeling of Agent Features. MAS methodologies usually propose to distribute all system functionalities / responsibilities among agents. Agents are an abstraction that provides some particular characteristics, such as autonomy and pro-activeness. Therefore, in our process, we define an activity whose purpose is to identify agent features, which are features that the agent abstraction is appropriate, and then they are modeled in particular activities. We claim that features of MAS-PLs that do not take advantage from agent technology can be modeled and implemented using existing SPL approaches. This explicit separation allows non-agent features to take advantage of other existing SPL techniques adopted to implement variabilities.

Two MAS-PL Case Studies Development. In order to identify problems on MAS-PL development and deficiencies of current approaches to modeling MAS-PLs, we developed the ExpertCommittee case study (Nunes 2008d, Nunes 2009f) (Chapter 4). This case study was later used to evaluate our process using an extractive SPL development strategy. In addition, a second case study, the OLIS, was developed using a reactive approach with the purpose of evaluating and testing the effectiveness of our process. Given that little research effort has been made in MAS-PL development, there are only a few case studies in this context, which are essential to validate proposed approaches.

Design and Implementation Guidelines. Based on our experience on MAS-PL development, we stated design and implementation guidelines with the goal of providing a better agent features modularization. These guidelines were also derived from the development experience of our case studies. We presented guidelines in three directions: (i) integration of web applications with software agents; (ii) implementing roles with two widely used agent platforms (JADE and Jadex); and (iii) use of Aspect-oriented Programming to provide a better feature modularization. Among these guidelines, the major contribution is the proposal of the Web-MAS Architectural Pattern (Nunes 2008b), which defines a general structure to add autonomous behavior to existing web applications using agent technology.

Besides the contributions of this dissertation, we also conducted research in cooperation with other members of our research group related to the work presented here. The use of AOP to implement agent features is exploited in (Nunes 2008c) and (Nunes 2009a). (Nunes 2008c) presents a quantitative study of development and evolution of the EC case study. (Nunes 2009a)

presents a comparison of the stability of implementation techniques for MAS-PLs. In the context of application engineering for MAS-PLs, (Cirilo 2008c) proposes an extension of the GenArch tool (Cirilo 2008a) in order to provide automatic product derivation.

7.2 Future Work

These contributions represent a first effort at supporting the development of MAS-PLs. In spite of the benefits identified in the use of the proposed approach, there are many ongoing and future works, some of which are described in the following.

Dealing with Fine-grained Features. In the literature, there are many examples of SPLs with coarse-grained features. This means that these features can be implemented wrapped in a specific unit, such as a class, a method or an agent. However, fine-grained extensions, e.g. adding a statement in the middle of a method, usually require the use of techniques, like conditional compilation, which obfuscates the base code with annotations. Though many SPLs have been implemented at the coarse granularity level using existing approaches, fine-grained extensions are also essential when extracting features from legacy applications (Kästner 2008). Our scope in this dissertation was dealing with coarse-grained features; however it is important to study ways of completely modularizing fine-grained agent features at analysis, design and implementation levels.

Use of Aspect-oriented Analysis and Design Approaches. Our domain engineering process provides support to crosscutting features by using specific notations to indicate variability introduced by them. Nevertheless, some aspect-oriented analysis and design approaches (Chitchyan 2005) have been proposed in order to identify and modularize crosscutting features, which are in this case seen as crosscutting concerns. Crosscutting concerns (Rashid 2003) refers to such quality factors or functionalities of software that cannot be effectively modularized using existing software development techniques. So, we are currently investigating how existing aspect-oriented modeling approaches (Jacobson 2004, Clarke 2005) can help the visual documentation of the agent features.

Experimental Studies to Evaluate our Process. We applied and evaluated our process based on the development of two case studies. Furthermore, some

activities of our approach, mainly related to the modeling of agent features, were adopted in the development of design and implementation projects in the Software Systems Design graduate course at PUC-Rio. However, empirical studies are very important to be performed given that the progress in any discipline depends on our ability to understand the basic units necessary to solve a problem (Basili 1996). Moreover, experimentation provides a systematic, disciplined, quantifiable, and controlled way to evaluate new theories. As a consequence, our future work includes making experimental studies to see how and when our process really works, to understand its limitations and to understand how to improve it.

Framework for Integrating Web-applications and Software Agents. Based on the development of our case studies, we have derived the Web-MAS Architectural Pattern. Besides these case studies share the same architecture, there are several elements such as classes that are used in both of them. Examples are the environment and facade agents, and classes that implement the Observer pattern. Therefore, a framework for developing web applications that have software agents on their architecture may be built to facilitate the development of this kind of application.

Tool Support. Our domain engineering process proposes the use of MAS-ML to model agent features and some models to provide feature traceability. However, none of them have tool support to facilitate the generation of diagrams. This is an essential motivation for the adoption of an approach. In the Software Engineering Laboratory at PUC-Rio, a prototype for designing MAS-ML diagrams has already been developed. In addition, some model-based derivation tools have been proposed in order to automate the derivation process of SPLs. One of them is the GenArch (Cirilo 2008a) tool, which was recently extended (Cirilo 2008c) by the incorporation of a new domain-specific architecture model for Jadex to enable the automatic instantiation and customization of MAS-PLs. There are some ongoing research work, whose aim is to integrate our models with GenArch.