Automatic Product Derivation of Multi-agent Systems Product Lines

Elder Cirilo, Ingrid Nunes
PUC-Rio, LES
Rio de Janeiro, Brazil
{ecirilo,ioliveira}@inf.puc-rio.br

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

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

ABSTRACT
Multi-agent Systems (MASs) development and Software Product Lines (SPLs) are two consolidated software engineering techniques. Recent research work explores the integration between them by proposing new templates and adaptations to document SPL variability in the context of MASs. However, the automatic product derivation process is not addressed in these works. In this paper, we propose a new extension to our existing model-based product derivation tool, called GenArch, in order to enable the automatic instantiation and customization of Multi-agent Systems Product Lines (MAS-PLs). A case study illustrates how the proposed extension can be used to derive products (instances) from a MAS-PL.

Categories and Subject Descriptors
D.2.13 [**Software Engineering**]: Reusable Software; I.2.11.d [**Artificial Intelligence**]: Multiagent systems

General Terms
Multi-agent Systems Product Lines

Keywords
Multi-agent Systems, Software Product Lines, Model-Driven Software Development, BDI Agents.

1. INTRODUCTION
Agent technology has emerged as a prominent technique to address the design and implementation of complex distributed systems. It provides a way to solve complex problems based on higher level abstractions (e.g. agents, roles). On the other hand, Software Product Lines (SPLs) have become a mainstream practice that addresses the design and implementation of a set of related software products, promoting large-scale reuse and mass customization. Recent research [3][4] has explored the integration between SPLs and Multi-agent Systems (MASs). However, the automatic application engineering process is not addressed in these works. The Application engineering, also called product derivation, specifies how the assets developed during the domain engineering process can be composed and customized to build a product of the SPL. Ideally, this process would be accomplished with the help of instantiation tools to facilitate the selection, composition, configuration and integration of SPL assets and their respective variabilities.

In this context, this paper presents an extension of the GenArch tool that addresses the application engineering of MAS-PLs. The original functionalities of GenArch are extended to incorporate a new Domain-Specific Architecture Model (DSAM) that models the abstractions and concepts of the MAS domain.

2. GENARCH
In our previous work [1][2], we have proposed and developed the GenArch, a model-based product derivation tool that aims at defining a lightweight process to the application engineering process. GenArch provides a set of derivation models that enable the domain engineer to model: (i) the problem space (feature model) –represents the concepts and features from a specific domain; (ii) the solution space (implementation model) –consists of the implementation artifacts used to build products of a SPL; and (iii) the configuration knowledge (configuration model) – defines how specific feature combinations in the problem space are mapped to a set of software components in the solution space.

The GenArch approach is driven by these models and works basically on three main steps: (i) automatic models construction; (ii) artifacts synchronizations; and (iii) product derivation. GenArch provides two Java annotations that can be used inside the source code of the SPL variation and variation points for enabling the automatic construction of the models. The purpose of these annotations is twofold: (i) characterize that a particular Java code asset addresses a specific feature (@Feature); and (ii) characterize that an annotated implementation element represents a variation point, such as a hotspot framework class (@Variability). Due to the fact that changes in the SPL implementation artifacts can occur during the SPL development, GenArch also implements capabilities that synchronize the derivation models with changes made in the SPL code assets.

The GenArch product derivation step, i.e. the customization and compositions of the SPL architecture, is driven by a configuration of the feature model. Based on a feature model configuration, GenArch performs models customization. It covers the creation, in memory, of an instance of derivation models that only contain elements that respect the feature model configuration and the configuration knowledge. According to the derivation models instances, GenArch decides which code assets need to be

SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

instantiated and customizes classes, aspects or configuration files. Template technology is used to implement the elements that must be customized. It collects information from the derivation models to customize their respective variable parts of code. The GenArch derivation process is concluded with code generation based on processing of templates and loading the selected and generated code assets in specific folders of a new Eclipse Java project.

2.1 GenArch Extensible Infrastructure

In order to enable variability management and automatic product derivation of complex SPL architectures, such as MAS-PLs, we considered necessary to extend the GenArch infrastructure [2]. Therefore, we develop a new extensible infrastructure whose purpose is to allow SPL engineers customize the tool through the addition of new DSAMs. This kind of model can be used to express a specific part of the SPL architecture that is dependent of a technology domain, for instance, Multi-Agent Systems in the case of MAS-P, and not is cover by implementation model..

For creating a domain architecture model, the domain engineer needs to create a meta-model defining the vocabulary of concepts provided by the platform and to define how these concepts can be put together. To integrate new domain architecture models with the GenArch infrastructure, a set of functionalities is provided, which encompasses an API and three Eclipse extensions points: (i) domain model; (ii) domain model extractor; and (iii) domain model derivation processors.

3. MAS-PL AUTOMATIC PRODUCT DERIVATION

In order to enable the automatic product derivation of MAS-PLs, we extended the GenArch base models to incorporate the Jadex DSAM. Jadex is a framework for MAS development based on principles of the Belief-Desire-Intention (BDI) model [15]. The Jadex first-class elements are agents, believes, goals, plans, capabilities, events and expressions. In particular, a Jadex capability allows that believes, goals and plans can be placed together in a separated module, in such a way that this module can be reused by different agents. Following this definition, the Jadex DSAM specifies how Jadex common and variable concepts are realized in the MAS-PL architecture and implemented by code assets. In the next sections, we outline how the OLIS case study, a MAS-PL of web applications that provide different services to the users, was modeled in GenArch (Section 3.1) and how it can be used to derive different products from this MAS-PL (Section3 .2).

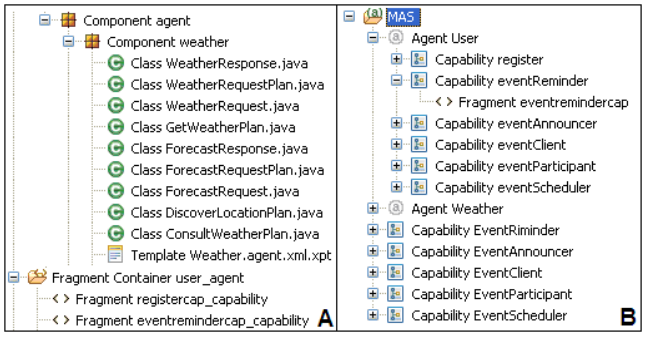

Figure 1. OLIS Derivation Models.

3.1 Modeling OLIS MAS-PL

The GenArch modeling process consists on the creation of the implementation, feature, configuration and DSAM models (Section 2). Figure 1(a) shows a partial view of the implementation model of the OLIS MAS-PL. The Weather package, for example, contains some Java classes that implement the Weather Agent. Figure 1(b) shows an instance of the Jadex DSAM that models the OLIS agent architecture based on Jadex concepts. This model only contains the OLIS variabilities (User Agent capabilities and Weather Agent).

The OLIS DSAM also specifies the mapping between its elements and elements of the architecture implementation model. These mappings indicate to GenArch how the DSAM elements are implemented. The relationships between domain variability and OLIS agency functionalities are specified in the configuration model. These models encompass the GenArch configuration knowledge, which is necessary to enable automatic selection and customization of the code assets during the product derivation process (Section 2).

3.2 OLIS MAS-PL Derivation

The GenArch derivation process for the OLIS MAS-PL is accomplished by the following steps: (i) selection of the Agents, Capabilities, Plans, Goals, Events, Expression that will compose the architecture of the product; (ii) selection of the code assets (class, aspects, files, components, folders) that will be part of the derived product; and (ii) customization of the Jadex Agent and/or Capabilities XML files (ADFs). The selection of the implementation elements is accomplished based on a feature model instance and configuration model. The customization of the OLIS Jadex Agent and/or Capabilities defined in ADFs is realized by means of template files and Jadex DSAM.

4. CONCLUSION AND FUTURE WORK

In this paper, we presented a GenArch extension that supports automatic product derivation of Multi-Agent Systems Product Lines. Essentially, we have extended the basic functionalities of GenArch tool to incorporate a new DSAM for the Jadex Framework. As a future work, we plan to extend GenArch to address: (i) the composition of different DSAMs; and (ii) to allow the domain engineer deal with meta-model constraints in an easy way. Finally, we intend to apply the tool in more complex MAS-PLs case studies with the presence of both coarse and fine grained variabilities.

5. REFERENCES

[1] E. Cirilo, et al. A Product Derivation Tool Based on Model-Driven Techniques and Annotations. JUCS, v. 14, n. 8, pp. 1344-1367, 2008..

[2] E. Cirilo,U. Kulesza, R. Coelho, C. Lucena, A. Staa. Integration Component and Product Lines Technologies. H. Mei (Ed.): ICSR 2008, LNCS 5030, pp. 130–141, 2008.

[3] J. Dehlinger and R. R. Lutz. A Product-Line Requirements Approach to Safe Reuse in Multi-Agent Systems. In SELMAS 2005, pages 1-7, USA, 2005. ACM Press.

[4] J. Pena, M. G. Hinchey, M. Resinas, R. Sterritt, and J. L. Rash. Designing and managing evolving systems using a MAS product line approach. Science of Computer Programming, 66(1):71-86, 2007.