6
Related Work

The main contribution of this dissertation is the proposal of a domain engineering process for developing MAS-PLs. Therefore, approaches that address the modeling and documentation of MAS-PLs are presented in this Chapter (Section 6.1). We have analyzed these approaches and pointed out their identified weaknesses.

In addition, we also describe architectures and frameworks that aim at incorporating agents into web applications (Section 6.2). These are alternative approaches to our Web-MAS architectural pattern. So, we discuss the problems we found on these approaches, and detail their differences compared to our proposal.

6.1
Multi-agent Systems Product Lines Approaches

In the literature, there are only few attempts that integrate SPL and MAS. Both existing approaches that address this integration – MaCMAS Extension (Pena 2006a) and GAIA-PL (Dehlinger 2007) – are based on existing MAS methodologies. The main problems that we have observed on these MAS-PL methodologies to model and document are: (i) they do not offer a complete solution to address the modeling of agent features in the whole engineering process; (ii) they suggest the introduction of complex and heavyweight notations that are difficult to understand when adopted in combination with existing notations (e.g. UML); and (iii) they do not capture explicitly the separated modeling of agent features.

6.1.1
MaCMAS Extension

Methodology Fragment for Analysing Complex MultiAgent Systems (MaCMAS) (Pena 2005) is an AOSE methodology that is designed to deal with complex unpredictable systems, which uses UML as a modeling language. In (Pena 2006a), it is proposed an approach to developing the core architecture of a MAS-PL. It consists of using goal-oriented requirement documents, role models, and traceability diagrams in order to build a first model of the system, and later use information on variability and commonalities throughout the products to propose a transformation of the former models that represent the core architecture of the family.

The software process proposed is composed by four activities: (i) build acquaintance organization – consists of developing a set of models in different layers of abstraction where we obtain a MaCMAS traceability model and a set of role models showing how each goal is materialized. This is achieved by applying the MaCMAS software process; (ii) build features model – responsible for adding commonalities and variabilities to the traceability model; (iii) analyze commonalities – it is performed a commonality analysis to find out which features, called core features, are more used across products; and (iv) compose core features – the role models corresponding to these features are composed to obtain the core architecture.

In (Pena 2006c, Pena 2007), Pena et al. use this approach to describing, understanding, and analyzing evolving systems. The approach is based on viewing different instances of a system as it evolves as different "products" in a SPL. That SPL is in turn developed with an AOSE approach and views the system as a MAS-PL.

One deficiency that we have identified in this approach is that the variable aspects of the systems are analyzed after modeling the MAS, and this can lead to undesirable situations, such as, high coupling between mandatory and optional features and inadequate modularization of agent features. Inadequate feature modularization results in non-effective application engineering and this increases time and costs for deriving new products, which contradicts the main reason for SPLs. In addition, the approach does not detail how the MAS-PL assets can be implemented so that they can be assembled to derive a product.

6.1.2 GAIA-PL

Dehlinger & Lutz (Dehlinger 2007) have proposed an extensible agent-oriented requirements specification template for mission-critical, agent-based distributed software systems that supports safe reuse. Their proposal adapted portions of the Gaia (Wooldridge 2000a, Zambonelli 2003) agent-oriented methodology and integrated them with a product-line-like approach, which exploits component reuse during system evolution. It is rooted within the requirements engineering phase. The proposed template allows ready integration with an existing tool-supported, safety analysis technique sensitive to dynamic variations within the components (i.e., agents) of a system.

The requirements specification template allows to capture dynamically changing configurations of agents and reuse them in future similar systems. The requirements are documented in two schemata: (i) Role Schema – defines a role and the variation points that the role can assume during its lifetime; (ii) Role Variation Point Schema – captures the requirements of a role variation point's capabilities. A Role Schema has an associated set of Role Variation Point Schemata. These schemata specify all the possible variation points of the roles, thus a new member to be added to the distributed system can be instantiated by specifying each new member to be deployed in the Agent Deployment Schema, which describes how different members of the distributed system that are to be deployed shall be instantiated.

The scope of this approach just cover a small part of the domain engineering process, focusing only on the requirements engineering. In addition, even though different capabilities can be specified for an agent, it is not possible to combine them. Thus, if a role schema has two variation points and one wants to choose both of them to compose an agent, a third variation point must be created defining both capabilities combined. Furthermore, the approach does not explicitly account for feature/role interactions.

6.2 Web-based Systems and Agent Integration Approaches

Several agent frameworks and platforms (Bellifemine 2007, Bordini 2007, Pokahr 2005) have been developed to ease the implementation of MASs. However, most of them do not address applications whose interface is the web. In this Section, we present some approaches that emerged in order to integrate agents and web applications. We also detail weaknesses of these approaches, which we aimed to solve with the Web-MAS architectural pattern (Section 5.1.1).

6.2.1 TaMEX Framework

Stroulia & Hatch (Stroulia 2003) propose a software framework called TaMeX, which supports the development of intelligent multi-agent applications that integrate existing web-based applications offering related services in a common domain. The TaMeX applications rely on a set of specifications of the domain model, the integration workflow, their semantic constraints, the end-user profiles, and the services of the existing web applications; all these models are declaratively represented in the XML-based TaMeX integration-specification language. At run-time, the TaMeX agents use these models to interact with the end users, monitor and control the execution of the underlying applications' services and coordinate the information exchange among them, and to collaborate with each other to react to failures and effectively accomplish the desired user request.

The TaMeX framework provides: (i) a language with well-defined semantics for specifying the application domain, the existing services and the service composition; (ii) a suite of intelligent methods for adapting existing web-based applications so that they "speak" the same – in terms of syntax and semantics-language; and (iii) a robust run-time environment for executing the integrated applications and deliver the desired overall service. The TaMeX architecture is a distributed multi-agent architecture consisting of two types of agents: task agents, which are responsible for interacting with the end users; and application wrappers, which are responsible for executing existing web applications and translating between the domain model of the integrated application and the individual domain models of the wrapped applications.

The Web-MAS architectural pattern also proposes the addition of software agents on existing web applications. However, we aimed at proposing a simple solution for that, which is founded on existing object-oriented design techniques and can be integrated with current adopted web technologies. On the other hand, the use of the TaMeX framework implies learning of a new programming language and specific methods. Besides, it obligates the use of its language to implement the software agents.

6.2.2
Choy et al. Approach

Choy et al. propose in (Choy 2005) the use of software agents to make the communication in a distance learning community more effective. They focus on the communication between teachers and students. Their software agents are designed to work on behalf of teachers, assisting them in communicating more effectively and closely with students, saving lot of time by delegating routine jobs. Basically, the software agents monitor the system and send alert e-mails in specific situations. The main elements that were introduced in the web application are: (i) `Schedule Control` class, which controls the agent's life and behavior; (ii) `Job Listener` classes that generate notifications when exceptional events occur; (iii) `Data Retrieval` class, which retrieves all required information at once for the `Job Central Processing` class; and (iv) `Job Central Processing` class, which hosts the criteria to generate e-mail alerts and sends them.

This work does not actually integrate agents into the web application.

The agents run in a parallel way with the system and consult the same database that is manipulated by the web application. Agents do not communicate with the rest of the system. Moreover, it does not allow the web application to access information directly from the agents. Both situations are addressed by the proposed Web-MAS architectural pattern.

6.2.3
Jadex Webbridge framework

A new architecture is proposed in (Pokahr 2007a) and (Pokahr 2007b), whose purpose is to seamlessly integrate agent technology and web applications. This architecture is in accordance with the Model 2 (Ford 2004) design pattern and the primary objective of the approach is to separate the agent-specific parts of an application from the web-specific parts such as HTML pages. An extra layer was introduced, performing the necessary mediation operations, to achieve the desired independence between the web front-end and the agent application.

The agentified Model 2 architecture refines only a small part of the original architecture by further developing the controller component, which serves as a connector that translates interactions with the view into actions to be performed on the model. This enables using agents for all aspects related to application functionality while preserving the usage of the existing and well suited Model 2 techniques for rendering (JSPs) and model representation (JavaBeans). One crucial aspect of this extended architecture is the partitioning of the controller into three distinct functionalities: delegate servlet, coordinator agent and application agents. The delegate has the main purpose to forward business tasks that originate from browser requests to the coordinator agent. The coordinator processes requests by distributing them to domain-dependent application agents.

This architecture is the basis for the Jadex Webbridge framework, which provides ready-to-use and extensible functionalities realizing the delegate servlet and the coordinator agent. Additionally, a web interaction module (capability) is provided that encapsulates the generic functionalities needed by application agents.

The Web-MAS architectural pattern also aims at seamlessly integrating agent technology and web applications; however this approach proposes that all the application logic is performed by software agents. Therefore, simple functionalities (e.g. create new system entities) that can be easily implemented with usual patterns and frameworks will not take advantage of these technologies. In addition, our pattern allows the integration with existing web applications developed with typical WAFs; furthermore Jadex Webbridge framework can not be integrated with this kind of frameworks, widely used in the web application development.

6.3 Final Remarks

This Chapter presented works related to two different topics of this dissertation: (i) approaches for developing MAS-PLs; and (ii) approaches that incorporate software agents into web applications. Only two recent research works have explored the integration synergy between SPLs and MASs. Pena et al. proposed a process based on the MaCMAS methodology to build the core architecture of MAS-PLs. Dehlinger & Lutz proposed a requirements specification template for agent-based applications that supports safe reuse. Both approaches present some deficiencies, which were described in this Chapter. Additionally, we have detailed approaches for integrating agents and web applications, indicating their weaknesses and how they were addressed in the Web-MAS architectural pattern.