Multi-agent Concepts as Basis for Dynamic Plug-in Software Architectures

Lawrence Cabac, Michael Duvigneau, Daniel Moldt, Heiko Rölke
University of Hamburg, Faculty of Informatics, Group TGI
Vogt-Kölln-Strasse 30
22527 Hamburg, Germany
cabac,duvigneau,moldt,roelke@informatik.uni-hamburg.de

ABSTRACT

In this work we present the basic concepts for a dynamic plug-in-based software architecture using concepts from the Petri net-based MAS framework MULAN. By transferring the concepts of agent-orientation to a plug-in-based architecture we are able to design our application and the plug-in-based system on an abstract level. Moreover, general problems that evolve from a highly dynamic and configurable architecture have been solved by basing the conceptual design on multi-agent principles. The introduced concepts have been applied to the software architecture of the tool RENEW.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Petri nets*

General Terms

Design

Keywords

Components, dynamic software architectures, high-level Petri nets, modeling, Mulan, multi-agent systems, nets-within-nets, plug-in architectures, reference nets, Renew

1. INTRODUCTION

Today's application software has to be adaptable, configurable and customizable to fulfill the needs of the users. Many system developers approach this challenge by introducing plug-in systems to extend or alter the functionality of these applications. Some systems are augmented with simple plug-in mechanisms, others reorganize the architecture of the application towards a system that consists exclusively of plug-ins. While applications of the first category usually resort to simple designs with extremely restrained possibilities of extending, the applications of the second category have to face many challenges to assure consistency and inter-operablility of plug-in components.

Plug-in-frameworks like those of Eclipse [2] or NetBeans [5] provide elaborated plug-in features in their practical environment. However, a conceptual model of a plug-in system is not discussed in this context.

By examining plug-in systems from the agent-oriented perspective, restrictions and problems of current systems can be discovered and attacked. Moreover, by basing the architectures of plug-in systems on agent-oriented principles, the designs of the application architectures adopt the advantages of multi-agent systems. These advantages are the handling of concurrency, conflicting functionality, service dependencies, locality and privacy, compatibility and dynamic extensibility. Challenges of plug-in systems can therefore be addressed in a more general way in MAS. One of the foremost benefits is that the plug-in architecture becomes dynamically extensible, i.e. functionality can be altered, added or removed at runtime without the need to restart the application.

Based on agent-oriented Petri nets we present a conceptual model for plug-in based systems. The idea is to structure and improve such systems using concepts from the agent-oriented area.

2. THE MULTI-AGENT SYSTEM MULAN

The multi-agent system architecture MULAN [4, 7] is based on the nets-within-nets paradigm by Valk [9], which is used to describe the natural hierarchies in an agent system. MULAN is implemented in the object-based reference net formalism using RENEW [6]. MULAN has the general structure as depicted in Figure 1: Each box describes one level of abstraction in terms of the net hierarchy. Each upper level net contains net tokens, whose structures are made visible by the ZOOM lines. Communication is carried across these

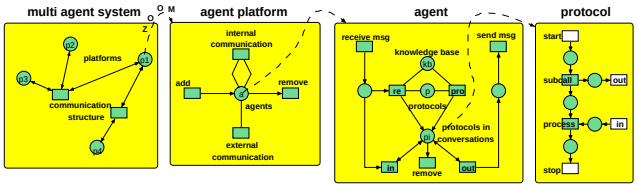

Figure 1: Agent system as nets-within-nets

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.
Copyright 2005 ACM 1-59593-094-9/05/0007 ...$5.00.

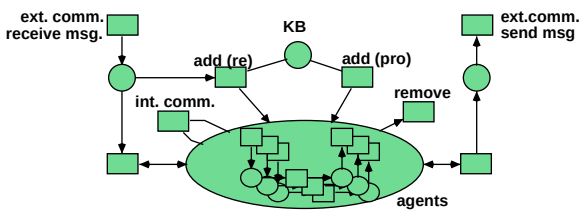

Figure 2: Agent as a platform

levels by channels at transitions. The figure shows a simplified version of Mulan, since several inscriptions are omitted. Nevertheless, Mulan is an executable model, and with the FIPA-compliant MAS framework Capa [1] a full-fledged agent platform implementation.

Platforms in a full featured Mulan system may act like agents and encapsulate the hosted agents. It is possible to exclusively use these platform agents as agents in a MAS. Following this approach leads to a dynamically reconfigurable MAS structure, i.e. a new hierarchy level may be introduced at run-time simply by creating a new (platform) agent and migrating other agents onto it. Figure 2 shows an agent that may serve as a platform for other agents. Those agents in turn serve as description of the platform agent's behavior.

3. DYNAMIC PLUG-IN ARCHITECTURE

A dynamic software architecture is characterized by extensibility and adaptability. We relate concepts of plug-in systems as extensible component systems as e.g. defined by Sametinger [8, p. 68] and agent-oriented systems. Both require self-contained, clearly identifiable artifacts that describe and/or perform specific functions and have clear interfaces. Extensibility is contained in both paradigms as a first-order concept: an agent system is extended by creating or migrating agents onto a platform. The life cycle of plug-ins matches well with the life cycle of agents as defined in the FIPA agent management.

In this work we conceive extensibility as a recursive feature. We apply the idea of nested platform agents to our concept, which leads to recursive extensibility. A system is extended by components, which again are extended by plug-ins, which are (specialized) components. The recursive agent-oriented plug-in model is a full-fledged plug-in system that allows for dynamic configuration. The similarity of structures on the top level and all other levels allows for the introduction of independent service and extension management units on every level. Our model is capable of describing a pluggable plug-in mechanism. Such a model is useful to merge multiple systems with independent management architectures.

In addition to the conceptual modeling of dynamic software architectures, we provide a practical example where the concept has been successfully applied in the development of the latest release of Renew (Version 2 of the multi-formalism Petri net IDE [3]). The former monolithic architecture has been exchanged with a system that is divided into a plug-in management system and plug-ins that provide (user) functionality. Through the introduction of the new architecture it is – at runtime – possible to dynamically extend the tool by registering plug-ins with the management system.

4. CONCLUSION

The presented generic – reference net multi-agent based – concept model for a dynamic architecture proves to be an approach that is both, sufficiently abstract for expressive modeling and sufficiently concrete to be able to transfer it to a real-world application. Moreover, it is the only modeling technique – to our knowledge – that is able to represent a flexible, adaptable and dynamic design of an application architecture. The level of abstractness is a benefit to the general design decisions. The level of concreteness helps the architect and developer to experiment and evaluate the model prior to the implementation.

We believe that this approach can be transferred to other application areas and applied as a general concept for various kinds of domains. Moreover, it is possible to generalize the implementation of the concept model as done in Renew to achieve a generalized core application that together with a conceptual approach can form a base for component-based application design of any kind.

5. REFERENCES

[1] M. Duvigneau, D. Moldt, and H. Rölke. Concurrent architecture for a multi-agent platform. In F. Giunchiglia, J. Odell, and G. Weiß, editors, *Proc. of AOSE 2002*, volume 2585 of *LNCS*, Berlin, 2003. Springer Verlag.

[2] Eclipse Homepage. `http://www.eclipse.org`, 2004.

[3] O. Kummer et. al. An extensible editor and simulation engine for Petri nets: Renew. In J. Cortadella and W. Reisig, editors, *Proc. of ICATPN 2004*, number 3099 in LNCS, pages 484–493, Berlin, 2004. Springer Verlag.

[4] M. Köhler, D. Moldt, and H. Rölke. Modelling the structure and behaviour of Petri net agents. In *Proc. of ICATPN 2001*, volume 2075 of *LNCS*, pages 224–242, Berlin, 2001. Springer Verlag.

[5] NetBeans Homepage. `http://www.netbeans.com`, 2004.

[6] Renew Homepage. `http://www.renew.de`, 2004.

[7] H. Rölke. *Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen*, volume 2 of *Agent Technology – Theory and Application*. Logos Verlag, Berlin, 2004.

[8] J. Sametinger. *Software Engineering with Reusable Components*. Springer Verlag, Berlin, 1997.

[9] R. Valk. Petri Nets as Token Objects - An Introduction to Elementary Object Nets. In J. Desel and M. Silva, editors, *Proc. of ICATPN'98*, number 1420 in LNCS, pages 1–25, Berlin, 1998. Springer Verlag.