Plugin-Agents as Conceptual Basis for Flexible Software Structures

Lawrence Cabac, Michael Duvigneau, Daniel Moldt and Benjamin Schleinzer

University of Hamburg, Department of Computer Science,
Vogt-Kölln-Str. 30, D-22527 Hamburg
http://www.informatik.uni-hamburg.de/TGI/

Abstract. To allow for flexibility in software structures (architectures) especially plugins and agents are proposed solutions. While plugins are used to support the conceptual and practical issues within component oriented software environments, agents are used in software areas where social metaphors like (self-)adaptability, flexibility, mobility, interactivity etc. are of interest. Common to both approaches is a strong relation to a service-oriented view on exporting functionality. This contribution illustrates the idea of the integration of both concepts on the formal basis of high-level Petri nets.

Keywords: High-level Petri nets, Nets-within-nets, reference nets, Renew, plugins, Mulan, multi-agent systems, plugin-agents

1 Introduction

While plugins and agents differ considerably in the current common perspective of software developers, there are some important similarities. On one side the area of component oriented computing usually concentrates on the flexible composition of software parts. Additional functionality is the main focus (see [1,2] for a thorough discussion of the concept). On the other side agents focus on the powerful modeling metaphors and concepts of autonomous behavior, mobility, pro-activeness, and communicative abilities [3]. There have been approaches to combine (mainly technical) aspects from both areas, plugin systems and agents technology [4,5]. This contribution now provides a more conceptual integration based on the formal model of high-level Petri nets (reference nets [6]). The integration can be considered as plugin-agents.

2 Conceptual and Technical Background

The three parts *Java*, Renew and Capa provide a runtime environment for plugin-agents and applications build on top. Renew can execute a multitude of net formalisms like reference nets concurrently and is written in *Java*. Reference nets combine object-based and nested modeling power with true concurrency. Capa is a Fipa compliant multi-agent-system that's implemented in reference nets and (pure) *Java*.

Burkhard et al. (Eds.): Multi-Agent Systems and Applications V, LNCS 4696, pp. 340–342, 2007.

Reference nets [6] are based on the colored Petri net formalism. They extend the colored Petri net formalism by combining the concepts of synchronous channels and the nets-within-nets concept by Valk [7]. In addition, reference nets [8] allow for net instances, dynamically changing nested net structures and the ability to include *Java* inscriptions. Renew [9] is a tool for the construction and execution of reference nets and their inspection (monitoring / debugging) at runtime. Due to the nets-within-nets paradigm, thus allowing to model active tokens embedded in an environment, the nested structures and the ability to dynamically change that structure, reference nets are a good choice to model plugins and agents.

Agents are designed to handle problems of conflicting functionality, service dependencies, locality and privacy, compatibility and dynamic extensibility [10]. Agents have the ability to act independently and autonomously (i.e. they decide on their reactions to inputs and act pro-actively, see [3]). Based on [11] Mulan can be used to build highly abstract models based on agent concepts.

Plugins can be seen as dynamic components [12]. In the same way as components, plugins provide functionality extension to enhance the software system [2]. However, plugins can achieve this goal in a dynamical manner by the mechanism described through the *plugin* metaphor. All this is supported by a plugin management system (PMS), which provides an interface to register, deregister and publish services of the plugins in the system. Each service has a description of how other plugins can utilize that service. The PMS also handles dependencies between plugins (compare [13]).

We combine advantages of both areas: agents and plugins. Agents do not allow to extend the functionality of other agents (entities in the system). They can nevertheless add to the functionality of a platform. By adding platform functionality to an agent, it can be extended by a pluggable agent. This idea leads to a natural inclusion of the plugin concept into agent technology.

Several advantages arise from the concept of a plugin-agent. We can assure locality for an agent. Usually in a multi-agent system, it is not possible to ensure that an addressed agent is located at a given platform. Response times can be reduced by using direct (synchronous) communication instead of time consuming (asynchronous) message handling in interactive systems. Nowadays, hardly any user accepts GUI interfaces that are not responsive. Locality and direct (proprietary) communication can ensure a secure connection between two entities. In mobile environments (mobile multi-agent system) user-personalized plugins are able to provide sensitive personal data and personalized functionality to mobile users, which can also connect at distinct and versatile platforms to the multi-agent system.

3 Conclusion

The combination of agent technology and plugin technology seems tempting. Advantages of both worlds can be used together to achieve more flexibility while increasing clearness at the same time. Agent technology provides autonomous

behavior, pro-activity, distribution and even mobility, while plugin technology provides well established concepts for light-weight extensible architectures.

Through explicit modeling of the concepts (plugin, agent and plugin-agent) we achieve a clear and well founded architecture. The visual models enable us to easily transfer conceptual knowledge to other developers. The operational semantics of the Petri net models allows us to refine the models to obtain a full fletched running agent framework providing those concepts. The executable versions allow for application development (rapid prototyping).

References

1. Sametinger, J.: Software Engineering with Reusable Components. Springer (1997)
2. Schumacher, J.: Eine Plugin-Architektur für Renew – Konzepte, Methoden, Umsetzung. Master's thesis, University of Hamburg (2003)
3. Wooldridge, M.: Intelligent agents. Multiagent systems: a modern approach to distributed artificial intelligence (1999) 27–77
4. Minh Vu, C.T.: E2 agent plugin architecture. In: 2005 International Conference on Integration of Knowledge Intensive Multi-Agent Systems, KIMAS'05: Modeling, Exploration, and Engineering. (2005)
5. Tu, M.T., Griffel, F., Merz, M., Lamersdorf, W.: A plug-in architecture providing dynamic negotiation capabilities for mobile agents. Lecture Notes in Computer Science **1477** (1998) 222–231
6. Kummer, O.: Referenznetze. Logos Verlag, Berlin (2002)
7. Valk, R.: Petri nets as token objects - an introduction to elementary object nets. In Desel, J., Silva, M., eds.: 19th International Conference on Application and Theory of Petri nets, Lisbon, Portugal. Number 1420 in LNCS, Berlin, Springer-Verlag (1998) 1–25
8. Kummer, O.: Introduction to Petri nets and reference nets. Sozionik Aktuell **1** (2001) 1–9 ISSN 1617-2477.
9. Kummer, O., Wienberg, F., Duvigneau, M.: Renew – User Guide. University of Hamburg, Faculty of Informatics, Theoretical Foundations Group, Hamburg. Release 2.0 edn. (2004) Available at: `http://www.renew.de/`.
10. Cabac, L., Duvigneau, M., Moldt, D., Rölke, H.: Agent technologies for plug-in system architecture design. In: Proceedings of the Workshop on Agent-oriented Software Engineering (AOSE), Utrecht, Netherlands (2005)
11. Rölke, H.: Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen. Volume 2 of Agent Technology – Theory and Applications. Logos Verlag, Berlin (2004)
12. Eichler, C.: Entwicklung einer Plugin-Architektur für dynamische Komponenten. Master's thesis, University of Hamburg (2002)
13. Cabac, L., Duvigneau, M., Moldt, D., Rölke, H.: Modeling dynamic architectures using nets-within-nets. In: Applications and Theory of Petri Nets 2005. 26th International Conference, ICATPN, Miami, USA, 2005. Proceedings. (2005) 148–167