Helper Agents as a Means of Structuring Multi-Agent Applications

Kolja Markwardt and Daniel Moldt

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

Abstract The Paose methodology of software engineering uses Multi Agent Systems as its main way of structuring applications. However as systems get larger and more complex, additional layers of abstraction are needed. Therefore we propose the HeRA-system (short for HElper and Resource Agents) to structure agent systems. In this paper we introduce the main concepts of HeRA and illustrate via a small example the usage of its prototypical implementation.

1 Introduction

Petri Nets provide a powerful formalism for modelling and implementing distributed concurrent systems. The Paose methodology (Petri net based Agent- and Object-Oriented Software Engineering [1]) uses software agents and Multi Agent Systems to develop distributed systems with reference nets [6]. Applying our approach showed problems when developing larger systems. The need for additional abstraction and structuring was identified in this context. First proposal have been made on this topic in [8,7]. Here we now show the final result as a consolidation of the former attempts.

When developing larger Multi Agent Systems (MAS) the question needs to be answered what kind of functionality we have to assign to an agent. To ease this, here we propose to use two types of agents to implement in the system. Doing so should give some hints how they should interact in order to achieve the intended goal of the system. We have experienced that having a type of agent eases the modeling of a system.

One way to distinguish elements of an MAS is between active and passive entities. Usually agents are considered active components. In [11] artifacts are introduced as another type of element in MAS, that agents can use and interact with. As similar approach based on Petri nets but not covering enriched social concepts has been proposed in [13] with the term of units.

The tools and materials approach (T&M) [14] for object-oriented software engineering distinguishes tools and materials as different artifact types which users can interact with in a software system. HeRA tries to adopt and extend ideas from the former three proposals for the creation of distributed user-centered agent systems. In the HeRA-system a user can access functionality by using

helper agents, that can act like tools to work on resource agents, who in turn can act like materials.

In the following, section 2 will give an overview over the PAOSE methodology and the MULAN/CAPA MAS that is used in the development of the HERA-system. Section 3 will then go on to describe the different concepts used in the HERA-system, which types of agents it consists of and how they interact with each other.

Section 4 explains these concepts by means of a simple example application built on HERA. Finally in section 5 we draw a conclusion and give an outlook on future work.

2 Developing applications with the PAOSE methodology

The PAOSE methodology of software engineering uses reference nets [6] for the modelling and implementation of software. Using interacting nets, the MULAN MAS [12] has been used for agent-oriented software development for years now.

In MULAN an MAS consists of agent platforms, which are connected with a communication infrastructure. Agents reside on platforms. A platforms manages the creation and deletion of agents and the communication between agents on a platform as well as between platforms. The behaviour of agents is determined by protocols and decision components within the agents. All these components are implemented as reference nets, interacting over synchronous channels.

A number of different modelling techniques are used in this approach, describing the system from a number of different yet linked perspectives [1]. The main focus in developing MAS applications with this approach is describing the different agents and agent roles within the system, the internal processes within the agents as well as the interactions between them, and the ontology used for representing concepts in the system. Interactions, internal processes and ontologies are implemented directly in petri nets (features structure nets for the ontology, reference nets for everything else).

The HERA-system now aims at further establishing an application-oriented perspective. By focussing on domain objects and supporting the users of the system, we hope to improve the usability and overall quality of software systems.

3 Helper and Resource Agents

In order to use an MAS application, users can interact with other agents within the MAS. This is often accomplished by means of a user agent, which represents the user within the system and usually provides the user with some kind of user interface that translates his input into agent activity. If new functionality is added to the system, the user agent, too, needs to be augmented, so that the user can access it.

In dynamic distributed systems, where new functionality is added frequently, this can be quite challenging. And if a typical user only needs a fraction of

the functionality, it is advisable to provide some kind of extension or plug-in mechanism that allows easy integration of new functionality on demand [2,3,4]. In HeRA we use helper agents, that plug into the user agent to provide new functionality where needed. The basic concepts of the HeRA system have been introduced in [8,7].

3.1 Overview

Figure 1 shows the different types of agents and platforms in the HeRA-system. A user connects to the system using his two-part user agent. The GUI is used for user interaction, while the agent part represents the user within the MAS. The user agent is connected to a number of helper agents that provide functionality and resource agents that represent resources and documents the user can work with using his helper agents.

The configuration of helper and resource agents a user has on his agent platform represents his personal workplace. He can use helpers to communicate and exchange resources with other users within his greater work environment. Agents can also meet and interact with each other on collaboration platforms, which represent e.g. location, places or groups within the system. Service platforms host agents that provide services, like the helper factory which is used to create new helper agents. Other examples could be a workflow management system or a security subsystem.

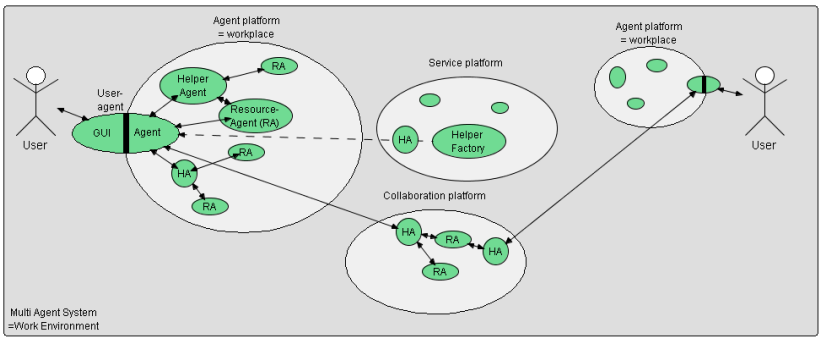

Figure 1. Agents and Platforms in HeRA

3.2 Useragent

The user agent consists of two parts, the GUI and the agent. The agent part is a Mulan agent that knows all the interactions necessary for handling helper agents and uses an RMI connection to communicate with the GUI part. The

GUI can therefore be located on the same or on a different computer than the agent part.

The GUI displays to the user a list of available helper agent types in the MAS, which he can choose to request from the helper factory (see below). If a new helper agent is registered with the user agent, it sends a description of his own user interface, which can then be integrated into the user interface of the user agent. That way the generic user interface of the user agent can be enhanced in any way needed to provide the functionality of the helper agent.

3.3 Helper Agents

In the T&M approach, users use tools to interact with materials and their environment. HERA turns these tools into agents, who can actively support a user in their work. This is reflected by the name helper agent, which emphasizes the more active role of the agent.

Helper agents are responsible for providing any kind of functionality to the user. They can provide a service all by themselves, encapsulate a legacy application, interact with other helper agents or display and manipulate resource agents.

3.4 Helper Factory

The helper factory is an agent used for creating new helper agents. It holds a list of helper agent types which it offers to user agents to choose from. On request it gathers all the information needed for creating the new helper agent and orders the agent management system to create it. The new agent can be customized for the user, for example only including functionality that the user can access according to his user permissions.

3.5 Resource Agents

Resource agents represent materials and resources in the work environment of a user. Instead of adding new concepts to the agent platform, resources are modelled as active components as well. A resource agent acts pretty much like an object, which can be handled by a helper agent, but it can also enforce its own rules. For example a material can decide which helper agents can access it or decide conflicts in concurrent access.

Helper and resource agents need to understand the same usage patterns, it is not possible to use a hammer to fasten a screw for example. These usage aspects represent a m:n mapping between different helper and resource types. As long as a helper understands the usage protocols of a resource, it can use that resource. Here the work on service manuals [5] can be used to improve the mutual dependencies with respect to the behavior.

4 Example: a Whiteboard Application

To illustrate these concepts, we provide an example in the form of a simple whiteboard application. A whiteboard is a common medium used by multiple people to communicate about ideas etc. While multiple people can read the content of the whiteboard, only one person at a time can write on it.

The whiteboard itself is modelled as a resource agent. It manages the content of the board and accepts requests for changes to this content. Helper agents can register with the whiteboard material agent to receive updates on the content whenever it changes.

The whiteboard helper allows a user to use the whiteboard application. The user requests a new whiteboard helper from the helper factory, which creates the agent for him. After registration the user agent loads the GUI extension for the helper (see figure 2) and connects it to the helper agent.

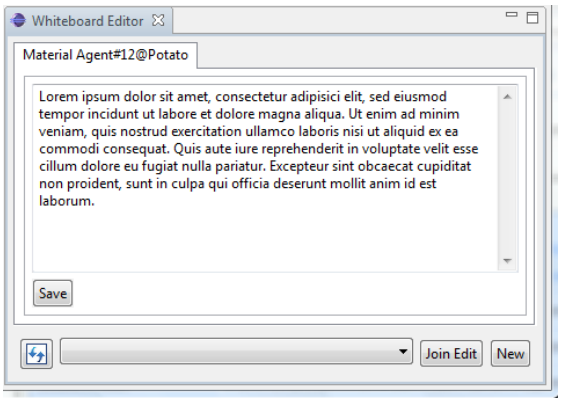

Figure 2. Whiteboard application

Using the agent by means of the GUI extension, the user can create new whiteboards or access a list of whiteboards already existing in the system and subscribe to them. He can then edit the content of the resource agent, which results in updates to all connected helpers. In some way this supports an event driven perspective.

The whiteboard example illustrates one of the possible arrangements for the collaboration platform in Figure 1. In general resource agents should be placed on such platforms if they do not belong to a single agent. Implicitly these platforms become collaboration platforms if several agents use the resource.

5 Conclusion and Outlook

Petri net based Multi Agent Systems can be used to structure net-based application development. In this paper we presented further structural elements within

MAS by introducing the concepts of helper and resource agents. These concepts provide the possibility to design applications that are more focussed on providing functionality to individual users collaborating within a distributed system. Additionally work objects can be modelled explicitly as first-order objects within the system.

In [10] it has been shown how these concepts can be used to leverage agent-based workflow management systems [9]. Future work focusses on combining these aspects further into an application development platform for complex distributed systems.

What has not been discussed here is the possibility of "feeding" agents with roles, goals, obligations etc. This allows for declarative style of programming which is nicely integrated due to the nature of agents. So also the use of social models that are currently discussed for the organization of agents can be applied to improve the overall architecture. Again this is inherently covered by MAS in general and hence can also be used in our helper and user agents.

References

1. Lawrence Cabac. *Modeling Petri Net-Based Multi-Agent Applications.* Dissertation, April 2010. http://www.sub.uni-hamburg.de/opus/volltexte/2010/4666/.
2. Lawrence Cabac, Michael Duvigneau, Daniel Moldt, and Heiko Rölke. Multi-agent concepts as basis for dynamic plug-in software architectures. In *AAMAS 2005*, pages 1157–1158, 2005.
3. Lawrence Cabac, Michael Duvigneau, Daniel Moldt, and Benjamin Schleinzer. Plugin-agents as conceptual basis for flexible software structures. In *Multi-Agent Systems and Applications V. CEEMAS'07, Leipzig.*, volume 4696 of *LNCS*, pages 340–342. Springer, 2007.
4. Michael Duvigneau. *Konzeptionelle Modellierung von Plugin-Systemen mit Petrinetzen.* Dissertation, October 2009.
5. Kathrin Kaschner, Peter Massuthe, and Karsten Wolf. Symbolische Repräsentation von Bedienungsanleitungen für Services. In *AWPN'06*, September 2006.
6. Olaf Kummer. *Referenznetze.* Logos Verlag, Berlin, 2002.
7. Kolja Lehmann, Lawrence Cabac, Daniel Moldt, and Heiko Rölke. Towards a distributed tool platform based on mobile agents. In *MATES'05*, volume 3550 of *LNAI.* Springer, September 2005.
8. Kolja Lehmann and Vanessa Markwardt. Proposal of an agent-based system for distributed software development. In *MOCA 2004*, Aarhus, Denmark, 2004.
9. Kolja Markwardt, Lawrence Cabac, and Christine Reese. A process-oriented tool-platform for distributed development. In *MSVVEIS 2009*, pages 44–52, 2009.
10. Kolja Markwardt, Daniel Moldt, and Thomas Wagner. Net agents for activity handling in a wfms. In *AWPN 2009, Karlsruhe, Germany*, 2009.
11. A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, 2008.
12. Heiko Rölke. *Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen*, volume 2 of *Agent Technology – Theory and Applications.* Logos Verlag, Berlin, 2004.
13. Volker Tell and Daniel Moldt. Ein Petrinetzsystem zur Modellierung selbstmodifizierender Petrinetze. pages 36–41, 2005.
14. Heinz Züllighoven. *Object-Oriented Construction Handbook.* dpunkt Verlag, 2004.