Net Agents for Activity Handling in a WFMS

Kolja Markwardt, Daniel Moldt, and Thomas Wagner

University of Hamburg - Department of Informatics
http://www.informatik.uni-hamburg.de/TGI

Abstract. Workflow Management Systems (WFMS) are used to organize work processes between different people within an organization or between organizations. In this paper we will describe an agent-based WFMS, built with Petri nets, to utilize the formal soundness of Petri nets and the flexibility of multi-agent systems to enhance the usefulness of a WFMS. The focus of this paper lies in the way activities are handled in the WFMS. We will first discuss the way this works in the system as of now. Then we will go on and describe a way to use Activity Agents to add a further flexibility to the activity handling of the system. These Activity Agents will be responsible for providing the functionality, including materials and user interface associated with an activity.

Keywords: Activity Agents, Reference Nets, Workflow Management Systems

1 Introduction

Workflow Management Systems (WFMS) are used regularly within companies. In research and practice the question of coupling the different WFMS of cooperating organizations arises, leading to inter-organizational WFMS. Agent-based WFMS are one answer in this field. In [6, 7] we proposed an agent-based WFMS and a process infrastructure for agent systems. The general model is quite elaborated, while the implementation has not been discussed deeply. Following the proposal of [5] the ideas of tools and materials (see [10] for this general approach) is introduced to multi-agent systems (MAS). Adding the concept of tools and materials to our Mulan reference model adds further structuring capabilities for the modeller.

In this contribution we will explain in Section 2 the technological basics of our WFMS. Section 3 explains our current agent-based WFMS. Then Section 4 provides the information about the tool and material approach and its adaptation to our agent-based approach. On top that the central idea, the Activity Agent is introduced. How to implement the afore mentioned Activity Agent concept is described in Section 5. The paper concludes with a short summary and outlook in Section 6.

2 Basics

The technological foundation for this contribution are the Mulan and Capa agent architectures. Mulan stands for **Mul**ti-**a**gent **n**ets, which is also the main

idea behind it. It was described in [8]. Every element of MULAN, including agent behavior, agent knowledge and agents themselves, is modelled using the reference net formalism introduced in [4].

CAPA (**C**oncurrent **A**gent **P**latform **A**rchitecture) is an extension of MULAN, which was introduced in [2]. Its main focus is on communication between platforms and making the MULAN principles fully compliant with the FIPA standards. The reference net tool RENEW serves as both development and runtime environment. A description of RENEW can be found in [4].

Within the WFMS workflows are modelled using a variation of the workflow nets described in [9]. This variation of workflow nets uses the task transition introduced in [3]. A task transitions actually represents three regular transitions. The three transitions model the request of a work item and the cancellation or confirmation of an activity.

3 An Agent-Based WFMS

In its current version the system supports basic WFMS functionality. It provides the means to administrate the system, add and edit workflow definitions and instantiate and execute workflow instances.

The functionality of the WFMS is provided by a number of different agent types. The system's core is made up of a trio of agents, which provide the internal function of the WFMS. These agents are the Workflow Engine agent (WFEngine agent), the Workflow Enactment Service agent (WFES agent) and the Workitem Dispatcher agent (WiDi agent). The WFEngine agent is responsible for firing the internal transitions of the tasks and for initiating the workflow instances. The WiDi agent distributes work items and activities to the users, if they are eligible for them. The WFES agent is located between the other two agents. It manages different workflow instances on the workflow engine. These three agents interact with each other to organize the functionality to provide work items and activities to the user. For each user a user agent exists, which manages the interactions between the WFMS and the user's GUI. Its main responsibility lies in invoking interactions upon actions of the user within the GUI and in initiating display updates within the GUI. The other agent types offer the functionality to manage and authenticate logged in users as well as access to the database.

Currently the WFMS supports two kinds of tasks. Simple tasks can only be accepted and then be marked as completed or canceled. They represent actions, which have to be completed outside of the functionality the WFMS offers. The other kind of tasks is form tasks. When a form task is assigned to a user, a form window is opened, in which the user enters the necessary information. Forms can consist of an arbitrary number of labels, text boxes, check boxes and radio buttons. When the form task is completed the data entered into a form is read into the system and can be used in later form tasks.

The subject of this contribution concerns the the way activities are handled within the system. Because of this it is important to give a description of the way activities are assigned in the current version.

While workflow instances are active within the system, eligible users can request the work items, which are currently activated in the workflow nets. When a user wishes to request a work item he initiates the associated interaction. If this interaction is successful the WFEngine agent fires the internal request transition of the task and creates the activity.

When any changes in a workflow net occur a decision component (DC) net, a special part of the WFEngine agent, is automatically informed. This listener DC net contains a cycle, which is responsible for handling reported changes in the set of activities and is always repeated when a new change occurs. The cycle checks which activities have changed and need to be updated. The cycle ends with the initiation of the UpdateActivityList interaction. The UpdateActivityList updates the internal lists of the WFEngine agent, the WFES agent and the WiDi agent.

After the UpdateActivityList interaction has been completed, the OfferActivityList interaction is started, in which the WiDi agent informs all user agents connected to him about the previously updated status of their activities. These updated activities are then displayed for the user and can be executed.

4 Tool- and Activity-Agents

In this paper we propose a new way of handling activities in the WFMS by using a special kind of Tool Agents, called Activity Agents. We will first describe the notion of Tool Agents and how they can be used to build flexible tool-based applications. Then we will describe the special incarnation of Activity Agents and the way they will be integrated into the WFMS architecture.

4.1 Tool Agents

Tool Agents are a way to use multi-agent systems to build tools for supporting individual users work as well as collaborative efforts. This follows the notions of the tools and materials approach [10], applied to multi-agent systems to address distributed workplaces.

Overview The main idea about the tool agent concept is that each user controls a user agent (UA), which can be enhanced by different tool agents (TA) as shown in [5]. The user agent provides basic functionality like a standard user interface and the possibility to load new tool agents. Those tool agents can then plug into the user agents UI with their own UI parts, offering their functionality to the user. By choosing the specific set of tool agents, the user can tailor his work environment to his specific needs.

Material agents (MA) are used to represent and encapsulate the materials or work objects that are currently worked on, like an insurance claim or a text file. Materials are manipulated by tools and can be created, deleted and moved between workplaces. Tools and materials populate the workspace of the user.

An agent called the Tool Factory is used to manage the different types of tool agents known to the system. It is called by the user agent to create a ne instance of a tool agent to use. Figure 1 shows how these agents work together.

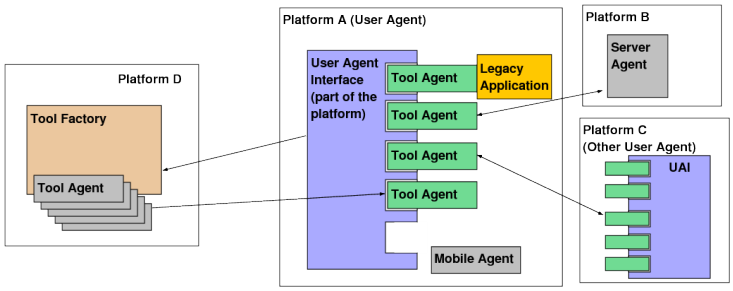

Fig. 1. User and Tool agents

4.2 Activity Agents

The definition of a workflow can contain any number of different types of tasks for a user to perform. In the current systems, all these tasks have to be defined in advance and the user needs to know how to handle them. It seems clear that in this way the user can be a real bottleneck in the deployment of new workflows, if these contain new types of tasks.

Tool Agents already provide a way to enhance the functionality of a User Agent. Therefore, an adaptation of Tool Agents for WFMS will provide a way to handle this problem, this adaptation is called an **Activity Agent**.

Like any Tool Agent it can be used to manipulate materials, but in this case the material and the context for its manipulation is provided by the workflow. An Activity Agent is not requested by the User to perform some task but it is rather assigned to it by the workflow engine to handle an activity. Once it is connected to the User however, it functions like another Tool Agent. It only needs a way to determine that the work on the activity is done, so that it can return the material and feedback on activity completion back to the workflow engine, for example a “finish” or “abort “ button in the GUI.

4.3 The Activity Agent in the Activity handling process

Activity Agents represent activities within the running WFMS. When a user successfully request a work item available to him, the system will automatically start a new Activity Agent. This Activity Agent will be responsible for this activity alone, and will only be active while the activity is being executed by the user. During the execution of the activity the user will exchange information with the Activity Agent, in order to work on the activity. When the activity has

been finished, the Activity Agent will transmit the relevant data back to the workflow engine and terminate.

5 Design and Implementation

In this section we will describe our proposed way to implement the Activity Agent. The Activity Agent will be started after the listener DC net of the WFEngine has detected that a new activity has been created. Before the Update-ActivityList interaction is started the listener will check if the activity is flagged as an Activity Agent activity. If the activity is to be executed by an Activity Agent, a new DC net is entered. The purpose of this new DC will be to start the new Activity Agent and add the agent's information to the activity, so that the executor's user agent knows how to interact with the Activity Agent. After this is done, the listener DC net can continue and start the UpdateActivityList interaction. Within The UpdateActivityList and OfferActivityList interactions only the internal lists have to be modified, in order to incorporate the added information.

While the Update- and OfferActivityList will not have to be changed much, the handling of activities in general and within the user GUI require changes. Concerning the handling of activities changes have to be made, because in the proposed version the actual handling of the activity will be done by the Activity Agent. In the current version the activities are manipulated within the GUI and then passed to the user agent, who, upon completion of the activity, sends them to the WFEngine agent. In the new version the user agent will not directly communicate with the WFMS's core in this matter, but will only communicate with the Activity Agent in order to manipulate materials involved in the activity (e.g. forms). When the user has finished his work on the activity he will inform the Activity Agent, who then informs the WFMS's core.

The reason for changes to the GUI is, that in the current version tasks are simply displayed in a list (simple tasks) or a generic form window is opened (form tasks). In both cases, the interface to complete the activity is embedded into the general GUI. Since the Activity Agent is designed to be able to offer arbitrary functionality with a possibly specialized GUI, this GUI has to be offered by the agent itself, because otherwise the main user GUI has to be changed and enhanced, whenever a new type of task is added. In order to support this, the GUI has to be changed so that selecting an activity will contact the Activity Agent who will in turn invoke his own GUI.

6 Summary and Outlook

We have proposed in this paper a way of enhancing an Agent-based WFMS with flexible activity-handling procedures. Specialized agents will be used to plug into the users' system and provide them with the functionality needed to perform their tasks within the workflow process. The underlying Petri nets allow first of all an appropriate modelling of the system and its processes. In addition

the agents and their behaviour become a well defined semantics. Following our Paose approach (see [1]) models are continuously transformed, so that they can be executed. Reference nets support this directly. Introducing the tool and material ideas into our multi-agent systems provides us with a highly expressible modelling basis. Activity agents directly rely on this. The flexibility introduced into the agent-based WFMS by the addition metaphor of the tool allows to introduce new activities much easier than the former way of explicitly defining all necessary parts redundantly.

Activity agents will directly be used in our next version of our distributed and no longer centralized agent-based WFMS. There it will take care in a quite generic way of activities of the workflows, the core of workflows in general.

References

1. Lawrence Cabac, Till Dörges, Michael Duvigneau, Christine Reese, and Matthias Wester-Ebbinghaus. Application development with Mulan. In Daniel Moldt, Fabrice Kordon, Kees van Hee, José-Manuel Colom, and Rémi Bastide, editors, *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'07)*, pages 145–159, Siedlce, Poland, June 2007. Akademia Podlaska.
2. Michael Duvigneau. Bereitstellung einer agentenplattform für petrinetzbasierte agenten. Diploma thesis, University of Hamburg, Department of Computer Science, December 2002.
3. Thomas Jacob. Implementierung einer sicheren und rollenbasierten workflowmanagement-komponente für ein petrinetzwerkzeug. Diploma thesis, University of Hamburg, Department of Computer Science, 2002.
4. Olaf Kummer. *Referenznetze*. Logos Verlag, Berlin, 2002.
5. Kolja Lehmann and Vanessa Markwardt. Proposal of an agent-based system for distributed software development. In Daniel Moldt, editor, *Third Workshop on Modelling of Objects, Components and Agents (MOCA 2004)*, pages 65–70, Aarhus, Denmark, October 2004.
6. Christine Reese, Jan Ortmann, Daniel Moldt, Sven Offermann, Kolja Lehmann, and Timo Carl. Fragmented workflows supported by an agent based architecture. In Manuel Kolp, Paolo Bresciani, Brian Henderson-Sellers, and Michael Winikoff, editors, *Agent-Oriented Information Systems III 7th International Bi-Conference Workshop, AOIS 2005, Utrecht, Netherlands, July 26, 2005, and Klagenfurt, Austria, October 27, 2005, Revised Selected Papers*, volume 3529 of *Lecture Notes in Computer Science*, pages 200–215. Springer-Verlag, 2006.
7. Christine Reese, Matthias Wester-Ebbinghaus, Till Dörges, Lawrence Cabac, and Daniel Moldt. Introducing a process infrastructure for agent systems. In Mehdi Dastani, Amal El Fallah, João Leite, and Paolo Torroni, editors, *LADS'007 Languages, Methodologies and Development Tools for Multi-Agent Systems*, volume 5118 of *Lecture Notes in Artificial Intelligence*, pages 225–242, 2008. Revised Selected and Invited Papers.
8. Heiko Rölke. *Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen*, volume 2 of *Agent Technology – Theory and Applications*. Logos Verlag, Berlin, 2004.
9. Wil M.P. van der Aalst. Verification of workflow nets. *Lecture Notes in Computer Science*, 1248/1997:407–426, 1997. Application and Theory of Petri Nets 1997.
10. Heinz Züllighoven. *Object-Oriented Construction Handbook*. dpunkt Verlag, 2005.