Workflow Management Principles for Interactions Between Petri Net-Based Agents (Extended Abstract)

Thomas Wagner[1] Daniel Moldt[1]

Abstract: Agents provide suitable mechanisms for modelling complex interactions in distributed systems. In some cases, though, classical agents interactions may exhibit issues of complexity and rigidness. These issues can lead to cumbersome and inefficient implementations. Introducing workflow and especially task principles into agents interactions extends the mechanisms available to agent modellers. With these improved mechanisms it is possible to avoid issues found in classical agents interactions. This contribution describes a Petri net-based approach on this topic. The work summarised in this extended abstract has been published in [WM15b].

Keywords: Workflows, Agents, Integration, Interaction, Communication, Petri Nets

1 Introduction

Agents interactions provide the means to design, model and implement complex relations between components (i.e. agents) in distributed systems. Agents interaction patterns are usually highly complex and describe which agents are exchanging what data at what precise point in their execution. The ability to model this complexity is beneficial for most scenarios. However, the strict association of specific agents or roles of agents to particular data and/or functionality also contains an inherent rigidity. Especially in an open system, agents with capabilities unknown beforehand may be available. Incorporating these agents into a system is challenging, due to the rigidly defined interactions.

One approach to solve and avoid these kinds of issues originates from the field of workflow management. Agents interacting to provide the functionality within a system are conceptually very similar to humans doing work facilitated by a workflow. Both, agents and humans, serve as resources to a set of related processes. These processes consist of tasks that need to be accomplished in order for the process to finish. For agents as resources, these tasks are usually implicit within their behaviour. At some point during execution they receive a message and react to it by performing the pre-defined behaviour. Another difference is that there is no separate and explicit workflow or workflow engine(s) present in an agent system. Instead, the agents themselves serve as the engines whenever they request work from other agents. Altogether, this means that agents serve as both workflow resources and workflow engines within an agent system considered in workflow terms. The approach described in this abstract implements this idea to improve agents interactions.

This extended abstract is structured as follows: After this introduction the overall approach is presented in Section 2, a prototype in Section 3 and the abstract concludes in Section 4.

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

2 Approach

The general approach takes the perspective that an agent system is a workflow system with a specialised form and categorisation of components. The behaviour of and between agents is considered in the form of workflow processes modelled as workflow Petri nets [vdA97]. These processes partition the functionality within the behaviour into explicitly modelled tasks. Tasks are intended to be executed by any agent that features the required functionality. The interface to the tasks is kept as generic as possible, simply requiring the form of input and output/result data. Task parameters describe the required functionality for the task as well as data formats. The understanding of the task concept follows the idea of the task transition [Ja02] for reference Petri nets [Ku02].

Figure 1 gives an overview of the approach. Some agents initiate, control and manage parts of the behaviour (i.e. the workflow processes) of a system. In workflow terms these agents act as workflow engines. Whenever a task becomes available, the information about it is sent to the so-called intermediary system. The intermediary system is a subsystem of agents providing standardised functionality. In workflow terms the intermediary system serves a function similar to a workflow enactment service (cf. [Ho95]). It gathers the information about available tasks from all engines and forwards it to the appropriate resources. Any agent of the system is a potential resource and registers itself and its capabilities with the intermediary system. The intermediary system filters available tasks for resources according to their capabilities. Resources can decide to request the tasks offered to them. When that happens the request is routed via the intermediary system to the engine. Note, that all communication between engines and resources happens with the involvement of or via the intermediary system. The engine has autonomous control over its workflows and can decide whether to agree to the request or not.

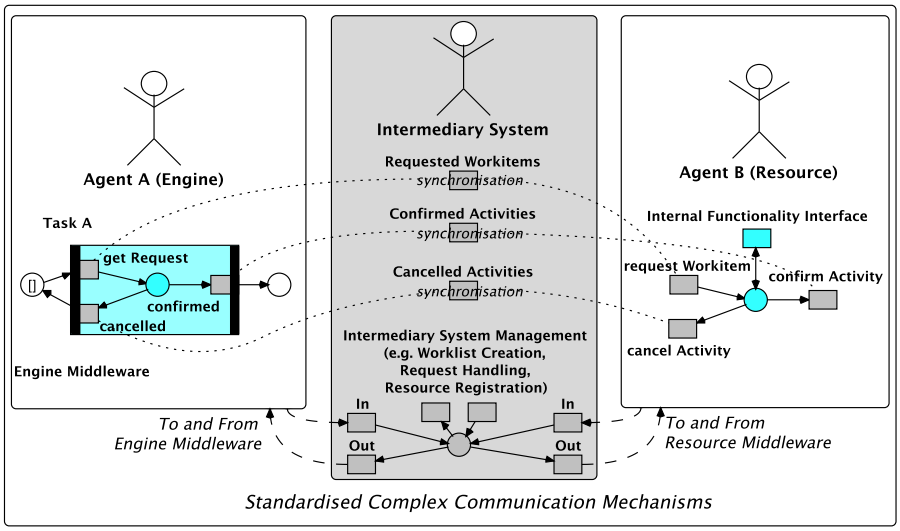

Fig. 1: Overview of the Approach (Edited from [WM15b])

If it agrees, the request transitions in all three components fire (conceptually[2]) synchronised (upper part of Figure 1). The resource then uses its internal functionality to execute the work associated with the task. If at any point the engine or the resource decide to cancel the task execution, the cancel transition in all three components is synchronised. If the resource is finished, it informs the engine of the result, but the engine has control whether or not to accept the result (or to cancel the task). Accepting the result confirms the task and fires and synchronises the corresponding confirm transitions in all three components.

The main effect of the approach is a decoupling of agents as interaction partners. Engines do not need to know beforehand which other agent is capable of and available to execute their requests. Agents simply send the task information to the intermediary system, which forwards it to all eligible resources. In cases of high workload the intermediary system may (in e.g. an open system) consider alternative resources and evaluate their functionality for a task. The benefits of the decoupling are increased flexibility and efficiency. There are also some disadvantages, especially w.r.t. the centralised nature of the intermediary system. These issues can mostly be solved in the implementation.

3 WORKBROKER-Prototype

The approach described above has been implemented in the WORKBROKER-prototype. The WORKBROKER realises an intermediary system using MULAN (**Mul**ti-**A**gent **N**ets, [Rö04]) and CAPA (**C**oncurrent **A**gent **P**latform **A**rchitecture, [DMR03]) agents. It is integrated as a proof of concept into our software development support system (SDCC) for the PAOSE approach (Petri net based, Agent- and Organization-oriented Software Engineering[3]). Agents (as engines) representing development teams execute development workflows. Tasks are e.g. requests for other development teams to fix bugs, change a common interface or clarify design choices. The engines create these tasks and send the information to the WORKBROKER intermediary system, which forwards them to the affected other development team agents. These agents then serve as the resources for the tasks with the WORKBROKER handling communication and management of the interactions.

4 Conclusion

In conclusion, the approach presented in this abstract enhances the way agents interact and cooperate. Utilising the idea of agents as engines as well as resources and considering interactions as (parts of) tasks and workflows allows modellers to exploit the advantages of workflows regarding behaviour modelling. It does not limit or prohibit any options regarding classical agents interactions and is thus a straightforward extension of capabilities for modellers. One can flexibly choose the best mechanism for each interaction.

In fact, the approach brings the concepts agents and workflows closer together. It is part of larger, ongoing work to combine and integrate the concepts of agent and workflow on the

[2] Actual implementations would only enact a pseudo-synchronisation within distributed settings.
[3] see `http://www.paose.de`

abstract level. Through such an integration, the strengths of both, agents and workflows, can be made available to system modelling. Introducing workflow principles into agent behaviour is an important step for this integration. More information about the overall work on the integration can be found in [WM15a].

Applying the agent/workflow approach to our own SDCC provides relevant insights into the complex but powerful mechanisms: Currently, a new version of the WORKBROKER is being implemented in the prototype for the overall integration of agents and workflows. This new WORKBROKER realises an intermediary system between entities that can dynamically act and be regarded as agents, workflows or both. These entities are used for modelling systems in which structural and behavioural modelling abstractions are considered equally. The new prototype not only considers agents interactions in workflow terms, but instead the entirety of the agent. Using these integrated entities makes it easier and more directly possible to exploit workflow principles for agents. The effects are currently being researched, but are, from a modelling point of view, very promising.

Literaturverzeichnis

[DMR03] Duvigneau, Michael; Moldt, Daniel; Rölke, Heiko: Concurrent Architecture for a Multiagent Platform. In (Giunchiglia, Fausto; Odell, James; Weiß, Gerhard, eds): Agent-Oriented Software Engineering III. Third International Workshop, Agent-oriented Software Engineering (AOSE) 2002, Bologna, Italy, July 2002. Revised Papers and Invited Contributions. volume 2585 of LNCS, Springer, Berlin, pp. 59–72, 2003.

[Ho95] Hollingsworth, David: The Workflow Reference Model. Technical report, WfMC, 1995. Available at http://www.wfmc.org.

[Ja02] Jacob, Thomas: Implementierung einer sicheren und rollenbasierten Workflowmanagement-Komponente für ein Petrinetzwerkzeug. Diploma thesis, University of Hamburg, Department of Informatics, Vogt-Kölln Str. 30, D-22527 Hamburg, 2002.

[Ku02] Kummer, Olaf: Referenznetze. Logos Verlag, Berlin, 2002.

[Rö04] Rölke, Heiko: Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen, volume 2 of Agent Technology – Theory and Applications. Logos Verlag, Berlin, 2004.

[vdA97] van der Aalst, Wil M. P.: Verification of Workflow Nets. In (Azéma, Pierre; Balbo, Gianfranco, eds): Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings. LNCS 1248, Springer, Berlin, pp. 407–426, 1997.

[WM15a] Wagner, Thomas; Moldt, Daniel: Integrating Agent Actions and Workflow Operations. In (Müller, Jörg P.; Ketter, Wolf; Kaminka, Gal; Wagner, Gerd; Bulling, Nils, eds): Multiagent System Technologies - 13th German Conference, MATES 2015, Cottbus, Germany, September 28-30, 2015, Revised Selected Papers. volume 9433 of Lecture Notes in Computer Science. Springer, pp. 61–78, 2015.

[WM15b] Wagner, Thomas; Moldt, Daniel: Workflow Management Principles for Interactions Between Petri Net-Based Agents. In (Devillers, Raymond; Valmari, Antti, eds): Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015. volume 9115 of LNCS. Springer, pp. 329–349, 2015.