Analysis of Multi-Agent Interactions with Process Mining Techniques

Lawrence Cabac, Nicolas Knaak, Daniel Moldt, and Heiko Rölke

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

Abstract. Process mining and multi-agent models are powerful techniques for the analysis of processes and organizations. However, the integration of both fields has seldom been considered due to the lack of common conceptual background. We propose to close this gap by using Petri nets as an operational semantics and consider process mining a useful addition to monitor and debug multi-agent systems in the development phase. Mining results can be represented in the formalized form of Petri nets that allows to validate or verify the actual behavior.

On our way to mining complex interactions within (simulated) organizations, we present a plug-in extension of our Petri net-based agent platform Mulan/Capa for recording interaction logs. Using process mining, the logs can be mapped by some intermediate steps to agent protocols e.g. represented as AgentUML interaction protocol diagrams. These diagrams are a descriptive representation form that combines organizational and control flow information. Furthermore, they can be mapped to executable Petri net, thus allowing to feed mining results back into the design phase.

Keywords: agent interactions, conversations, high-level Petri nets, interaction mining, mining, multi-agent systems, Mulan, modeling, nets-within-nets, process mining, reference nets, Renew, simulation.

1 Introduction

The concept of Multi-Agent Systems (MAS) has gained increasing importance in computer science during the last decade. MAS research considers systems as aggregations of goal-oriented, autonomous entities (agents) interacting in some common environment (see e.g. [1]). Since no or only minor central control is exposed on the agents, a coherent global system behavior emerges merely from their cooperative or competitive interactions.

The design, implementation, and validation of MAS still remains a demanding task. Petri nets are frequently applied for modelling agent behavior due to the

K. Fischer et al. (Eds.): MATES 2006, LNAI 4196, pp. 12–23, 2006.

typical combination of formal conciseness and visual clearness as well as the possibilities of displaying and formally analyzing concurrent systems [1]. Petri nets also support the verification and validation of MAS, since formal methods can be applied to assess liveness and safety properties of such models.

Unfortunately, the applicability of formal verification techniques is limited to simple and often practically irrelevant classes of MAS [2]. Furthermore such techniques can only be applied in a confirmative fashion; i.e. to verify (or falsify) previously posed hypotheses about a system's behavior. Agent-oriented software engineering (AOSE), however, is primarily an experimental process [2] consisting of prototypical design, simulation, observation and a-posteriori analysis in order to explore the system's behavior. Since the observation of even simple MAS might produce large and complex amounts of data [3], data mining has occasionally been proposed as a support technique for such analysis (see e.g. [4,5]).

To aid the understanding of dynamic processes – in particular interactions – in MAS, it seems straightforward to apply techniques from *process mining* originally developed in the domain of business process intelligence (see e.g. [6,7]). These techniques seem especially appropriate in Petri net-based AOSE due to their ability to reconstruct concurrent Petri net models from execution traces. This leads to a number of potentially interesting applications during the AOSE development cycle.[1] (1) In the *system analysis phase*, process mining can be employed to aggregate behavior or interaction traces of relevant agents from the real system to Petri net models that flow into the design phase. (2) In the *design phase*, process mining seems to be a promising approach to integrate adaptability into Petri net-based agents by providing them with the ability to learn executable models of behavior from the observation of other agents' interactions. (3) In the *validation phase*, process mining can be used to aggregate large amounts of trace data observed from the running system. These models can be visualized, formally analyzed or compared to design models to validate the system's behavior. Also, process mining might support the detection of unforeseen, implicit interaction patterns emerging at runtime.

In this paper, we present an approach towards the application of process mining techniques to the analysis, design and validation of multi-agent interactions. In particular, we pursue the goal of reconstructing models of agent interaction protocols from sample interactions. Our approach is integrated into the FIPA-compliant, Petri net-based agent platform Mulan/Capa.

The paper is organized as follows: Section 2 briefly introduces Mulan and Capa. In Section 3 we review existing work on agent interaction analysis and introduce process mining as an advanced analysis technique. In Section 4 we present our approach towards analyzing agent interactions by means of process mining, where Petri nets build an important intermediate representation. In Section 5 we discuss our prototypical implementation of a tool for interaction monitoring, debugging and validation. Finally, Section 6 concludes the paper with a discussion of our results reached so far and of possible future research.

[1] Similar applications of general data mining to MAS are discussed in [5].

2 The Multi-Agent System Architecture Mulan

The MAS architecture Mulan (MULti Agent Nets) [1] is based on the nets-within-nets paradigm [8], which is used to describe the natural hierarchies in an agent system. It includes four system views depicted in Figure 1: MAS infrastructure (1), platform (2), agent (3), and protocol (4). These are related by the mechanism of *token refinement*. A MAS is modelled as a Petri net whose places contain tokens representing platforms. Each platform is itself a net whose central place hosts all agents that currently reside on this platform. An agent consists of exactly one *agent net* that is its interface to the outside world and an arbitrary number of *protocol nets* defining its behavior. The variety of protocols ranges from simple linear step-by-step plans to complex dynamic workflows.

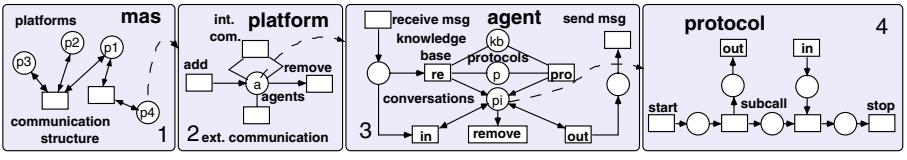

Fig. 1. Agent system as nets-within-nets

Mulan is implemented in reference nets and Java using the Petri net simulator Renew [9,10]. Reference nets provide a concurrency semantics and a natural concept of distribution and locality. Java allows us to seamlessly include external functionality such as graphical user interfaces. Compatibility of the Mulan framework to the FIPA specifications [11] is ensured through its partial re-implementation Capa (Concurrent Agent Platform Architecture) [12], which provides a seamless integration with other FIPA-compliant MAS frameworks. It also allows us to participate in heterogeneous environments such as OpenNet [13]. Especially in these environments, the determination, analysis and coordination of interactions are challenging tasks. In the following sections we show how these challenges can be approached using techniques from process mining.

3 Interaction Analysis and Process Mining

Interaction analysis is currently an important topic in MAS research for the reasons mentioned above. In the following, we review related work on interaction analysis, and introduce process mining as an advanced analysis technique.

3.1 Interaction Analysis in Multi-Agent Systems

Many frameworks for multi-agent application development include debugging tools that allow to monitor the message traffic on the agent platform. An example is the *Sniffer agent* integrated into the JADE framework [14]. This tool

displays observed message sequences as UML sequence diagrams and provides basic filtering capabilities. Monitoring agent interactions leads to large amounts of data. Important behavior patterns are in danger to go unrecognized when the analysis is performed by hand. Therefore data mining techniques are increasingly applied in this context (see e.g. [5]).

The algorithms for this task are mostly based on computational logic and stochastic automata: Nair et al. [4] e.g. propose an approach towards team analysis in the domain of (simulated) robot soccer (*RoboCup*). They consider three complementary perspectives: The *individual agent model* is a situational decision model of a single agent represented by means of association rules. The *multiple agent model* represents agent interactions as a stochastic automaton. The *global team model* shows relations between team properties (e.g. ball possession time) and game results in a rule-based fashion.

Botia et al. [15] focus on mining social networks at multiple resolutions from message logs using the *ROCK* cluster algorithm. In addition, their monitoring tool *ACLAnalyser* can automatically observe the execution of predefined interaction protocols on the JADE platform. Mounier et al. [16] present an approach towards *agent conversation mining* using stochastical grammar inference. Mining results are represented as a stochastic automaton whose edges are labelled with message performatives. The approach neglects concurrency and interaction roles. Hiel [17] applies extended Hidden Markov Models for the same task; also neglecting the aforementioned aspects. However, he suggests to improve the reconstruction of (concurrent) protocols by process mining techniques as a possible direction for future research. Parallel to the publication at hand, Dongen et al. report on the application of process and decision tree mining to communication logs observed in an auctioning simulation based on ad-hoc agent concepts [18]. This work proposes the introduction of adaptability by means of process mining.

3.2 Process Mining

Process mining (*workflow mining*) is a subfield of data mining concerned with "method[s] of distilling a structured process description from a set of real executions" [19]. The task is – given an *event log* recorded during process execution – to reconstruct properties of the generating processes. While most research is done in the area of *business process management* [6], other application domains such as the analysis of *web service interactions* [20] have recently been considered.

A large number of process mining techniques are available, that can be classified by the *perspective* that the analysis focuses on. The most prominent perspectives are control flow and organizational perspective [7]. The objective in the *control flow perspective* is to reconstruct the observed process' control structure – i.e. sequences, branches, loops and concurrency. The *organizational perspective* focuses on the "structure and the population" of the organization in which the processes are observed. This covers "relations between roles [...] groups [...] and other artifacts" [7]. Tool support for process mining is increasingly becoming available. Aalst et al. developed the *ProM* process mining tool that is extensible through a plugin mechanism by mining, import, export and analysis plugins [21].

An often-cited mining technique for the control flow perspective is the α-Algorithm: From an event-based process log, this algorithm builds a concurrent Petri net model on the basis of a *direct successor* relation. An extension of the algorithm can be proven to reconstruct any net belonging to the class of *extended sound workflow nets* [22], but it cannot cope with noise, hidden tasks, and duplicate tasks.[2] Herbst [6] developed an algorithm for mining process models containing duplicate tasks from activity-based logs.[3] Research on mining in the organizational perspective has so far focused on the reconstruction of *role assignments* [23,24] and *social networks* [25]. Further tasks in process mining are *log segmentation* (i.e. the mapping of messages from the process log to process instances and process classes) and *condition mining* (i.e. inference of branching conditions in the process model). Both are covered in an approach by Schütt [26].

Interaction mining – i.e. the reconstruction of interaction models from message logs – covers aspects of both control flow and organizational structure. Gombotz et al. [20] apply interaction mining to analyze the operation of web services at different levels (operation, interaction and workflow). One of the mining results is a so-called *web service interaction graph* representing the relations of a particular web service and its neighbors. Aalst [24] shows that the α-algorithm can be used to mine sequence diagram-like Petri net-structures from message logs. The approach is restricted to 1:1 interactions and does not explicitly abstract from senders and receivers to interaction roles.

4 An Approach Towards Agent Interaction Mining

Though the similarities between the analysis of multi-agent interactions and the research field of process mining have recently been recognized in the literature (see above), the integration of process mining into practical methods and tools for AOSE is still in its infancy. In the following, we present our approach towards analysing agent interactions with process mining techniques.

4.1 Context

Our approach towards Agent Interaction Mining (AIM) is integrated into a larger framework for Process Mining in (Agent Oriented) Software Engineering (ProMiSE, see [27]). This framework covers several analysis perspectives related to the four conceptual levels of Mulan: (1) the *decision perspective* focusing on decision models encoded in an agent's knowledge base, (2) the *internal control perspective* regarding the processes running within a single agent, (3) the *external control perspective* concerned with multi-agent interactions, (4) the *structural perspective* focusing on (static) platform and MAS structures, and (5) the *multi-level perspective* regarding relations between the perspectives mentioned before.

[2] A *hidden task* is a nameless activity not registered in the log. *Duplicate tasks* occur if the same activity is executed under different preconditions.

[3] In an activity-based log we can identify start and end events of activities which eases the detection of concurrency.

On our way to applying mining techniques to the analysis of MAS on multiple levels, we choose the *external control perspective* as a starting point. From the observation of message traffic, we proceed *bottom up*, i.e. we try to reconstruct basic interaction protocols in the first step. Through the recursive application of mining techniques to the results of the previous level, we aim to proceed to hierarchical protocols and higher level dynamical and structural patterns.

4.2 Techniques

The task of AIM at the protocol level is formulated as follows: Given a message log recorded during the execution of a MAS, find the unknown set of interaction protocols involved in the generation of this log. This task can be divided into several sub-phases depicted in Figure 2. Generally, we consider the FIPA ACL message attributes `performative`, `sender`, `receiver`, and some conversation control tags. By masking message content, we keep the following stages application-independent.

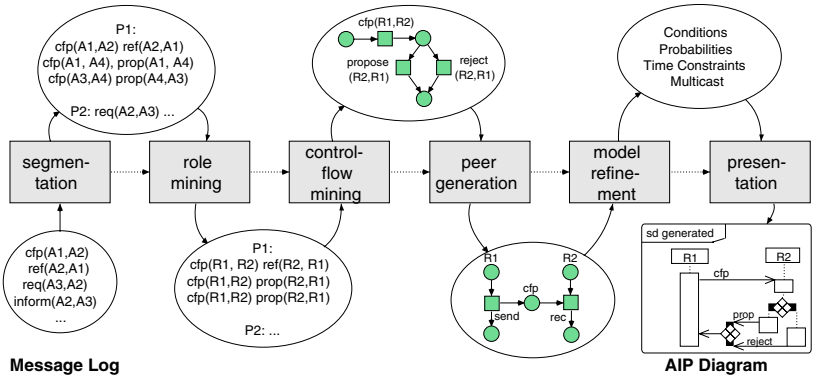

Fig. 2. A mining chain for agent interaction mining

The first phase – log segmentation – is necessary because a log normally contains messages from several conversations, generated by multiple protocols. These messages must be sorted by assigning them to a *conversation*; and by assignging each conversation to a *protocol type*. Given the information available in FIPA ACL messages (e.g. `conversation-id`) this segmentation is trivial.

However, these tags are not excessively used on the Capa platform and might generally prove as too inflexible for detecting complex patterns of interaction. Therefore, we reconstruct conversations by *chained correlation* [28] of messages based on the `in-reply-to` tag: Messages without this tag are assumed to start a new conversation. Other messages are appended to the conversation currently ended by a message with a corresponding `reply-with` tag. In doing so, we obtain $1:1$ *conversation threads*. However, these might be part of a larger multi-party

conversation that we reconstruct by merging all conversation threads sharing at least one `reply-with` or `in-reply-to` tag.

Assigning conversations to protocol types is a clustering task. For each conversation, we build a feature vector representing a *direct successor* relation of performatives.[4] Each vector component represents one possible succession of two performatives. It is assigned a value a counting the number of appearances of this succession in the conversation. To regard for protocols with a typically branched control structure, combinations of performatives appearing near the start of a conversation are weighted stronger than those appearing at the end. Finally, we apply the *nearest neighbour* algorithm [30] to cluster similar vectors based on the Euclidian distance.

The result of the segmentation phase are traces of conversations ordered by protocol types. In the second phase – role mining – we further abstract the messages by replacing names of sender and receiver agents with conversation roles. We currently apply a simple unification algorithm that binds agent names to role names in the order of their appearance in the conversation. However, this simple approach might fail in branched or concurrent protocols. Alternatively, we consider using a role detection mechanism based on sets of sent and received performatives similar to the approach described in [29].

In the third phase – control flow mining – we collect the abstracted conversation traces of each protocol type and try to induce a model of the protocol's control flow. Interaction protocols such as those specified in AgentUML might contain concurrent, hidden, and duplicate tasks. Therefore, the algorithm by Herbst [6] seems to be a good choice at first sight. However, this algorithm requires an activity-based log, while the message log is event-based.

Based on ideas from [6] and [26], our preliminary process mining technique consists of two stages – automata inference and concurrency detection: First, we reconstruct a deterministic finite automaton (DFA) from each set of samples using the k-RI algorithm [31]. The edges of the DFA are labelled with message performatives and sender/receiver roles. The k-RI algorithm can detect loops and duplicate tasks, but not concurrency. We therefore apply a modified version of the α-algorithm to the DFA next. Based on the successor relation of labelled transitions, the algorithm detects hints for concurrency in the DFA's structure.

Control flow mining results in an overall Petri net model of each protocol. This model can be split straightforwardly into protocol templates for every conversation role. Each of these peers corresponds to one lifeline in an AgentUML interaction protocol diagram (AIP, see [32]), that might be used to visualize the mining results. Additionaly, we are planning to refine the reconstructed model by infering temporal relations between messages with techniques described in [7]. We will also apply the C 4.5 decision tree learning algorithm [30] to reconstruct branching conditions from message content attributes as proposed in [6]. The attachment of branching conditions to the protocol templates leads to executable Mulan protocols.

[4] A similar metric is used in a preliminary approach by Vanderfeesten towards detecting conversation roles [29].

5 A Tool for Agent Interaction Mining

In this section, we present a prototypical tool and show an example for the application of our interaction mining techniques.

5.1 Monitoring Tool

To integrate process mining facilities into the CAPA platform, we developed a monitoring tool named MULAN *Sniffer* as a RENEW plugin [33]. The name indicates that the tool's functionality was derived from typical MAS debugging tools such as the *JADE Sniffer* [14]. The MULAN Sniffer monitors all ACL messages sent between agents on the platform during a simulation. The resulting message log is displayed textually as a list or graphically as a UML sequence diagram. Filters can be applied to select messages containing certain performatives, etc.

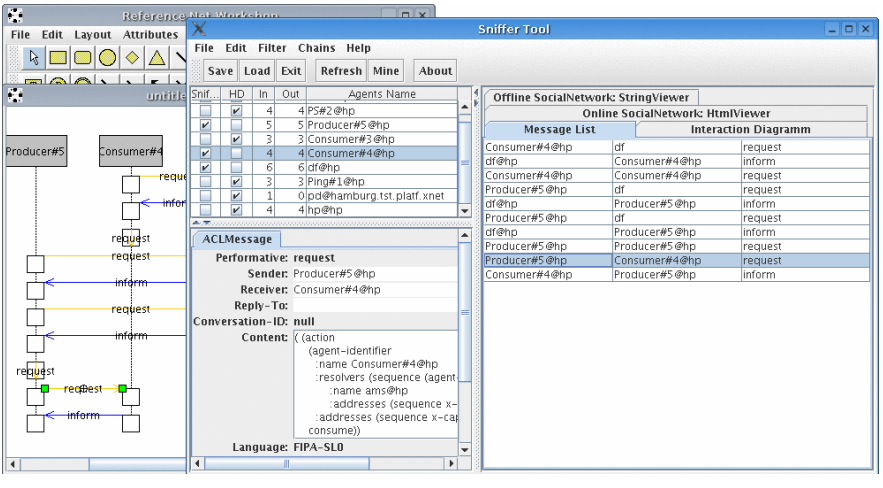

Fig. 3. MULAN Sniffer UI with observed interactions and RENEW UI

Figure 3 shows the user interface of the Sniffer with an observed message log. The messages in the diagram are color coded to ease the monitoring of the MAS. They can be inspected in the bottom left view of the Sniffer window. The upper left view shows a list of observed agents which can be sniffed or blocked. It also shows the numbers of messages sent and received per agent. The tool allows to observe changes in the diagram on the fly, i.e. when the message is sent.

The MULAN Sniffer differs from its 'ancestors' in two aspects that are important for our approach: (1) The recorded sequence diagrams are stored in the same format used by the MULAN design tools. They can therefore be edited and mapped to executable agent protocols. (2) More important, the *Sniffer* is a pluggable RENEW plugin [33] that can be extended by plugins for process mining and filtering itself.

The interfaces for filtering and mining plugins are reminiscent of similar tools such as *ProM* [21]. Special emphasis is put on the *recursive* character of process mining algorithms: These algorithms operate on data and provide data for higher-level analysis. We therefore introduce the concept of *mining chains*. Complex process mining algorithms are constructed by combining basic building blocks in data flow networks as proposed in [34]. This visual modeling technique is frequently used in data mining tools and can be supported by the Petri net editor of Renew.

5.2 AIM Plugin and Example

The example in Figure 4 shows a plugin that applies the algorithms described in Section 5 to the message log provided by the Sniffer. The messages partly result from multiple executions of a concurrent protocol simulating negotiations between a customer, a mediator and a service provider to allocate an order.

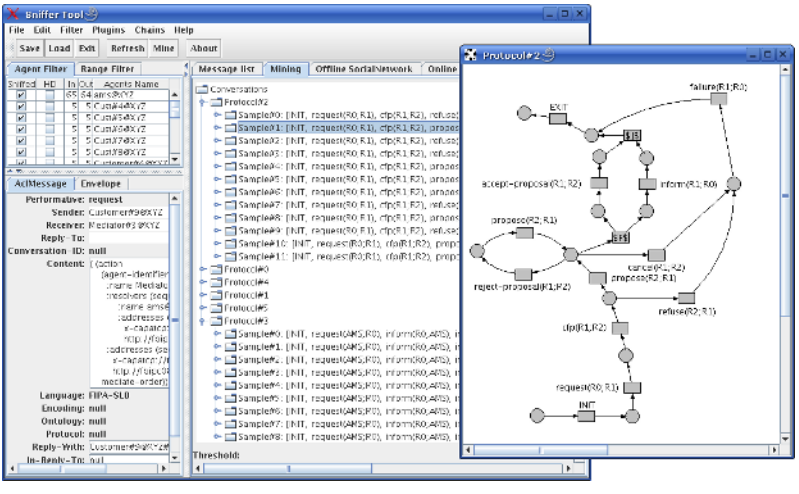

Fig. 4. AIM Plugin of the Mulan Sniffer showing mined conversations

In Figure 4 the Sniffer UI can be seen (background) with a tree-view that shows the results of the log segmentation. Each tree node represents one identified protocol type with the respective conversations as children. On selecting a conversation, the associated messages are automatically highlighted in the sequence diagram view (see also Figure 3).

In the example, the messages belonging to the order allocation protocol where successfully separated from the surrounding 'noise', i.e. conversations executed during the initialization of agents and platform. However, the performance of the clustering procedure strongly depends on a threshold for cluster similarity that needs careful calibration. The window in the foreground shows the correctly reconstructed Petri net model of the order allocation protocol.

6 Conclusion

The Mulan / Capa framework offers an integrated tool set supporting the development of Petri net-based MAS. It includes features for the specification, creation, documentation, monitoring and debugging of multi-agent applications. However, in concurrent, distributed and heterogeneous environments the analysis of multi-agent interactions is extremely difficult. Thus there is a need for elaborated techniques to handle large amounts of data. Process mining is one technique that can be successfully applied. The more abstract view of interaction mining allows to emphasize the desired perspectives (e.g. external control perspective) that are important for agent-based development and analysis.

This paper shows how to embed interaction mining into agent-oriented software engineering. We have developed an approach to reconstruct interaction protocols from message logs; integrating and extending several process mining techniques. It allows us to structure message logs by means of clustering and to reconstruct non-trivial concurrent protocols. However, we have encountered several cases the techniques cannot handle yet. Enhancing and validating them in greater detail is an important topic at issue. We have furthermore presented the Mulan Sniffer, a monitoring tool that is extensible by mining and filtering plugins. It is also applicable to many other FIPA-compliant MAS, which allows to monitor and mine in heterogeneous multi-agent environments and thereby evaluate our mining techniques in numerous real-world situations.

In our future work, we will validate the presented process mining techniques in empirical and analytical studies and provide necessary extensions. An important drawback of the current approach is the inability to appropriately handle multicast-protocols where the number of agents bound to each conversation role is not constant (e.g. the well-known *ContractNet* protocol). Furthermore, we are planning to tackle the challenging problem of reconstructing hierarchical protocols (see also [17]) and develop mining techniques for the other perspectives described in Section 4.1. In this context, the potential of Mulan as a conceptual framework for process mining will be investigated further in two directions: for providing a classification of techniques and for providing background knowledge to improve mining results.

Besides the analysis and validation of MAS, we are planning to apply process and interaction mining in a broader context of AOSE: On the one hand, we will improve the adaptability of Petri net-based agents through mining capabilities within our *Socionics* project [35]. On the other hand, the developed techniques might support online monitoring as well as the work of software developers in terms of reflecting their behavior to provide a feedback about best practices. Again this requires a sufficiently powerful toolset which we have sketched in [36]. In the context of inter-organizational processes the agent metaphor is highly applicable when considering the actors. Techniques that are applied to automated systems can be applied to the users in such environments as well. Therefore the legal, social, ethical and practical issues resulting from the application of process mining within the environment of people's computer support (with or without their knowledge) require urgent investigation.

References

1. Rölke, H.: Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen. Volume 2 of Agent Technology – Theory and Applications. Logos Verlag, Berlin (2004)
2. Edmonds, B., Bryson, J.: The insufficiency of formal design methods - the necessity of an experimental approach - for the understanding and control of complex MAS. In: AAMAS. (2004) 938–945
3. Sanchez, S.M., Lucas, T.W.: Exploring the world of agent-based simulations: Simple models, complex analyses. In Yücesan, E., Chen, C.H., Snowdon, J.L., Charnes, J.M., eds.: Proceedings of the 2002 Winter Simulation Conference. (2002) 116–126
4. Nair, R., Tambe, M., Marsella, S., Raines, T.: Automated assistants for analyzing team behaviors. In: Autonomous Agents and Multi-Agent Systems 8. (2004) 69 – 111
5. Remondino, M., Correndo, G.: Data mining applied to agent based simulation. In Merkuryev, Y., Zobel, R., Kerckhoffs, E., eds.: Proceedings of the 19th European Conference on Modelling and Simulation, Riga, SCS-Europe (2005) 374–380
6. Herbst, J.: Ein induktiver Ansatz zur Akquisition und Adaption von Workflow-Modellen. PhD thesis, University of Ulm (2001)
7. van der Aalst, W., Weijters, A.: Process mining: a research agenda. Computers in Industry **53**(3) (2004) 231–244
8. 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
9. Kummer, O.: Referenznetze. Logos-Verlag, Berlin (2002)
10. Kummer, O., Wienberg, F., Duvigneau, M.: Renew – The Reference Net Workshop. `http://www.renew.de` (2006) Release 2.1.
11. FIPA: Foundation for Intelligent Physical Agents. `http://www.fipa.org` (2005)
12. Duvigneau, M., Moldt, D., Rölke, H.: Concurrent architecture for a multi-agent platform. In Giunchiglia, F., Odell, J., Weiß, G., 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 Lecture Notes in Computer Science., Berlin Heidelberg New York, Springer-Verlag (2003)
13. openNet: Project. `http://www.x-opennet.org/` (2005)
14. JADE: Java Agent DEvelopment Framework. `http://jade.cselt.it` (2005)
15. Botía, J., A.López-Acosta, A.F.Gómez-Skarmeta: ACLAnalyser: A tool for debugging multi-agent systems. In de Mántaras, R.L., Saitta, L., eds.: Proceedings of the 16th Eureopean Conference on Artificial Intelligence, Valencia, IOS (2004) 967–968
16. Mounier, A., Boissier, O., Jacquenet, F.: Conversation mining in multi-agent systems. In: Proceedings of the CEEMAS 2003. (2003) 158 – 167
17. Hiel, M.: Learning interaction protocols by overhearing. Master's thesis, Utrecht University (2005)
18. van Dongen, B., van Luin, J., Verbeek, E.: Process mining in a multi-agent auctioning system. In Moldt, D., ed.: Proceedings of the 4th International Workshop on Modelling of Objects, Components, and Agents, Turku (2006) 145–160
19. Maruster, L., Weijters, A., van der Aalst, W., van den Bosch, A.: Process mining: Discovering direct successors in process logs. In: ICDS: International Conference on Data Discovery, LNCS (2002)

20. Gombotz, R., Baina, K., Dustdar, S.: Towards web services interaction mining architecture for e-commerce applications analysis. In: International Conference on E-Business and E-Learning, Amman, Jordan, Sumaya University (2005)
21. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A new era in process mining tool support. In: ICATPN. (2005) 444–454
22. Medeiros, A., Dongen, B., Aalst, W., Weijters, A.J.M.M.: Process mining: Extending the α-algorithm to mine short loops. BETA Working Paper Series, WP 113, Eindhoven University of Technology (2004)
23. Ly, T., Rinderle, S., Dadam, P., Reichert, M.: Mining staff assignment rules from event-based data. In: Workshop on Business Process Intelligence (BPI), in conjunction with BPM 2005, Nancy, France (2005)
24. van der Aalst, W.: Discovering coordination patterns using process mining. In Bocchi, L., Ciancarini, P., eds.: First International Workshop on Coordination and Petri Nets (PNC 2004), STAR, Servizio Tipografico Area della Ricerca, CNR Pisa, Italy (2004) 49–64
25. van der Aalst, W., Song, M.: Mining social networks: Uncovering interaction patterns in business processes. In: Proceedings of the 2nd International Conference on Business Process Management, Potsdam (2004)
26. Schütt, K.: Automated modelling of business interaction processes for flow prediction. Master's thesis, University of Hamburg, Department for Informatics (2003)
27. Cabac, L., Knaak, N., Moldt, D.: Applying process mining to interaction analysis of Petri net-based multi-agent models. Technical Report 271, University of Hamburg, Department of Informatics (2006)
28. van der Aalst, W., Dumas, M., Ouyang, C., Rozinat, A., Verbeek, H.: Choreography conformance checking: An approach based on BPEL and petri nets. Technical Report BPM-05-25, BPMcenter.org (2005)
29. Vanderfeesten, M.: Identifying Roles in Multi-Agent Systems by Overhearing. Master's thesis, Utrecht University (2006) in preparation.
30. Dunham, M.H.: Data Mining: Introductory and Advanced Topics. Prentice Hall, Upper Saddle River (NJ) (2003)
31. Angluin, D.: Inference of reversible languages. Journal of the ACM **29**(2) (1982) 741–765
32. Cabac, L., Moldt, D., Rölke, H.: A proposal for structuring Petri net-based agent interaction protocols. In van der Aalst, W., Best, E., eds.: Lecture Notes in Computer Science: 24th International Conference on Application and Theory of Petri Nets, ICATPN 2003, Netherlands, Eindhoven. Volume 2679., Berlin Heidelberg: Springer (2003) 102–120
33. 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 2005, Miami, USA, June 2005. Proceedings. (2005) 148–167
34. Jessen, E., Valk, R.: Rechensysteme: Grundlagen der Modellbildung. Studienreihe Informatik. Springer-Verlag, Berlin (1987)
35. Homepage: Socionics in Hamburg. `http://www.informatik.uni-hamburg.de/TGI/forschung/projekte/sozionik/` (2005)
36. Lehmann, K., Cabac, L., Moldt, D., Rölke, H.: Towards a distributed tool platform based on mobile agents. In: Proceedings of the Third German Conference on Multi-Agent System Technologies (MATES). Volume 3550 of Lecture Notes on Artificial Intelligence., Springer-Verlag (2005) 179–190