From Formal Workflow Models to Intelligent Agents

Markus Hannebauer
GMD — German National Research Center for Information Technology
Research Institute for Computer Architecture and Software Technology (FIRST)
hannebau@first.gmd.de

Abstract

Todays business workflows, e.g. in administration, have a strong need for support by information technology, because they heavily rely on the efficient and fluent interaction of human actors and artificial systems. At the same time, workflow tasks supported or automatically executed by computers get more and more complex. Intelligent agents try to cope with both of these settings by providing a Software Engineering abstraction, which incorporates complex functionality for task planning and execution with elaborate interaction capabilities. This paper presents a method for identifying and specifying such agents and their behavior by careful analysis of formal workflow models. It follows a small example out of an industrial-size case study, which is currently conducted at Europe's biggest hospital, Charité Berlin.

There are at least two aspects of todays business workflows, which have to be considered for support by information technology. The first aspect is the partially or fully automated execution of workflow tasks. These tasks get more and more complex and enforce the use of sophisticated methods, including planning, scheduling and constraint satisfaction. There are various successful contributions from Artificial Intelligence on these topics (rf. e.g. to (Allen et al. 1990; Zweben and Fox 1994; Yokoo et al. 1998)). The second aspect is the control of the task execution process itself. Since most real-world tasks and their execution are distributed spatially and among individuals, the coordination and interaction between the task execution entities is central to the management of business processes. Workflow Management has made successful contributions to this field since the beginning of 1990 (rf. e.g. to (Lawrence 1996; Jablonski 1995)).

The notion of an *Intelligent Agent* (rf. e.g. to (Huhns and Singh 1998; Jennings and Wooldridge 1998)) is a recent concept, which tries to incorporate the merits of classical AI methods with strong communication and interaction capabilities. Hence, it seems to be predestined for application to workflow management, especially under the assumptions proposed by HEWITT's *Open Systems* (Hewitt 1988). Unfortunately, it is not simple to identify possible agents in a given business context and to specify their behavior. Should agents represent certain individuals, roles or even organizational units in the business domain or should they rather implement representations of tasks, processes and resources? In my opinion, these questions cannot be answered generally but only in the context of the given application domain. Nevertheless, I will present a method, which describes the identification and the specification of agents and their properties by a careful analysis of formal workflow models. This method can be applied to a wide range of applications, which involve distributed process control.

The term "agent" can be interpreted as being a new abstraction in Software Engineering, just like objects. Even pioneers of object-oriented technology are aware of the usefulness of intelligent, goal-directed entities. A. COCKBURN states in (Cockburn 1997), "I hope that one day we shall have more formal tracking of goals and backup goals.", and I. GRAHAM can be quoted, "Clearly the goal or contract is in the mind of some agent rather than being something possessed by the task itself." Nevertheless, there has been a lot of confusion about this term, especially outside the Distributed Artificial Intelligence community. An interesting survey on possible agent definitions is given in (Franklin and Graesser 1996). Throughout this paper, I will use the term "agent" to denote an artificial representative, which autonomously acts in behalf of a human or organizational actor in the application domain.

For presentation, I will follow a small example out of an industrial-size case study. This case study is explained in the first section. In the following section a brief description of the used workflow model is given. The next section presents the method for identification and specification of agents and their skills. Concluding remarks follow in the last section.

The Case Study

Intelligent Agents have traditionally been applied to the control and optimization of industrial transport and production processes. Examples for such research can be found in (Burmeister et al. 1996) or (Liu and Sycara 1998). In contrary to that, research on workflow management and agents in business contexts is more in-

volved with human processes, typically in the domain of administration and services. Some results of applying agents to such domains are reported in the fields of meeting scheduling (Sen and Durfee 1995), telecommunications (Jennings et al. 1996) and health care management (Huang, Jennings and Fox 1995).

Several researchers from Humboldt University Berlin and GMD FIRST are currently carrying out an industrial-size case study at the cardiological clinic of Charité Berlin, Europe's biggest hospital. The cardiological clinic consists of **five wards** with a capacity of altogether over 80 patients, **four outpatients' facilities**, in which different types of medical consulting are done in parallel, and **eight diagnostic units**, some of which with several subunits. The diagnostic units perform over 100 diagnostic examinations each day. These examinations are requested by the wards, the outpatients' department and other clinics of Charité.

The present problem is the coordination between the requesting and serving units. Spatial and organizational distribution of the named units results in distributed knowledge, distributed control and hence suboptimal patient throughput and resource usage. Since most of the patients' care pathways are similar, this problem seems to be tractable by workflow management and optimization techniques, but traditional monolithic workflow management engines scale purely in measure of workflow instantiations and they usually ignore the problem of restricted information distribution. Therefor, a more local and flexible architecture is needed to control and optimize the requesting and serving workflow in diagnosis. We decided to design and realize a truly distributed multi agent system, which will (hopefully) run on 25 to 30 computers allover the whole cardiological clinic. This system is called ChariTime. It shall be permanently active to allow the dynamic allocation of actors and resources to diagnostic tasks, while coping with failures and emergency cases.

Modeling Workflows

The first step towards the design and realization of a system like ChariTime is a detailed analysis of the present situation. Dependent on the complexity of the analyzed domain, informal or formal models may be used to describe the observable workflows. The advantage of informal workflow models is the quick gain of results. In contrary to that, formal workflow models spend some additional time, but help to prevent missing or even incorrect analysis results. In both cases, the analysis must cover static aspects, dynamic aspects and the specification of the control objective.

Static Aspects

The target of the static model is the definition of all entities, which participate in the observed workflow, together with their signature. Entities, which have to be described this way, include tasks, jobs (composed of tasks), actors, resources, processes, process units and process systems. The latter two describe organizational groups of the other entities and help to structure the static model.

Within the ChariTime project all these entities have been formalized by algebraic specification, which means the signature and semantics. As an example I will present the static definition of a process p for a given task t. For briefness, the signature is left out.

Definition 1 (Process) *A* process p *is a 7-tuple*

$$p = (id, requ, cons_{succ}, cons_{fail}, compl_{succ}, compl_{fail}, P_{fail})$$

The semantics of the given constants and functions is as follows. id is a unique identifier of the process within the domain. requ provides the set of actors and resources, which are required to enact t. p may succeed or may fail w. r. t. the given task t, the involved actors and resources. In both cases, p has a certain impact on the involved actors and resources. $cons_{succ}$ *specifies the consequences of the process on actors and resources in case of a successful enactment,* $cons_{fail}$ *the consequence in case of a failed enactment. In a similar way* $compl_{succ}$ *provides a cumulative distribution function over completion time in case of a successful enactment.* $compl_{fail}$ *is defined accordingly for the failure case.* P_{fail} *denotes the failure probability of the process.*

□

Since the static definition of actors and resources includes role identifiers, the function *requ* might not base on the unique identifiers of actors or resources, but on their role identifiers, which would result in a relaxation of actor and resource requirements.

Dynamic Aspects

The target of the dynamic model is the description of the states of all involved entities and their interrelations over time. Petri Nets suite very well to this target, because they are tailored to the formal description of concurrent, distributed processes. Petri Nets can be used to describe and verify dynamic aspects of workflows as for example shown in (Aalst 1997) or (Graw and Gruhn 1995). *Colored Petri Nets* have been successfully applied to the modeling and evaluation of production systems (Kis, Neuendorf and Xirouchakis 1997; Zimmermann 1997). Since I have used algebraic specification techniques to describe the static aspects, I use *algebraic Petri Nets* (Reisig 1998), which are related to colored Petri Nets. I use a customized class, which has been extended with notions of time and stochastics.

Using this kind of Petri Nets, the dynamic state of a job can be described as well as the constraints given by the environmental system. Figure 1 illustrates a net representing a job, which consists of five tasks. Processes are denoted by transitions, task specifications and states by places. As one can easily see, the execution sequence of task 2 and 4 can concurrently run to the execution of task 3. Task 5 synchronizes the job execution and finishes it.

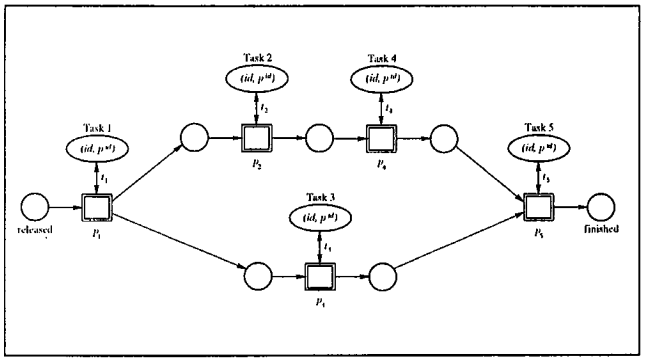

Figure 1: Sample Job Net with 5 Tasks

Figure 2 shows a simple system net, which specifies the dynamic interrelations between two process units for transformation of jobs and two process units for transportation of jobs and actors/resources. Transformation ($tr1$, $tr2$) as well as transportation processes (e.g. $tp1,2$) require certain resources and sometimes the involvement of actors. Any information on time and stochastic behavior is left out on this level of detail.

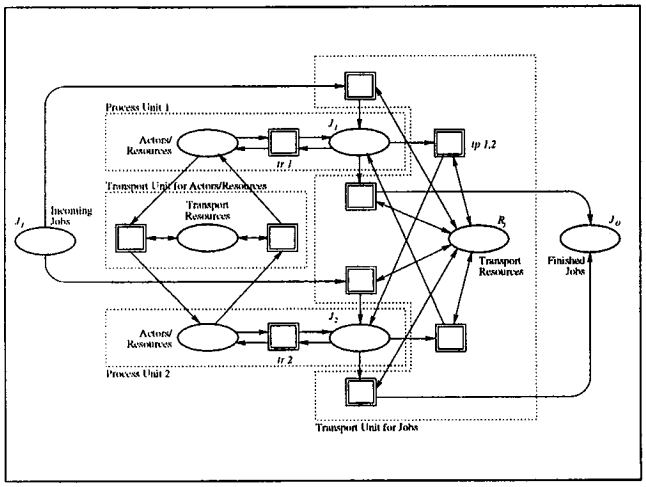

Figure 2: Sample System Net

Specification of the Control Objective

The general control objective in process control or optimization is the useful assignment of jobs (resp. their tasks) to actors and resources in time. The first part, the assignment to actors and resources, can easily be illustrated by defining a mapping, which *dynamically binds* process transitions in jobs nets to process transitions in the system net. This idea bases on *Object Systems* (Valk 1995), but in my case the binding is not static. For example, dynamically binding p_3 in figure 1 to the transformation process $tr1$ in figure 2 specifies the concrete use of certain resources and the involvement of certain actors (this is defined by the *requ* function of $tr1$). Hence, the objective of the control system is finding a dynamic binding between job nets and system nets.

The additional assignment to points in time can be illustrated by formalized observations of the concurrent run of the bound nets. *Concurrent runs* (Reisig 1998) are protocols of possible observable runs, which are again denoted by acyclic Petri Nets. Timing this protocols means to assign a certain starting time point to each bound process transition and finishing time points to the places in the post-set of the process transitions. Figure 3 illustrates the observable firing of the transport process transition $tp1,2$, which represents the transport of job j from storage J_1 to storage J_2 using the transport resource r. This notation is equivalent to GANNT plans, as the dashed box shows (the y-axis represents the actor/resource dimension, the x-axis the time dimension). Due to these timed concurrent runs optimization criteria, like minimum makespan, weighted makespan etc. can be specified canonically.

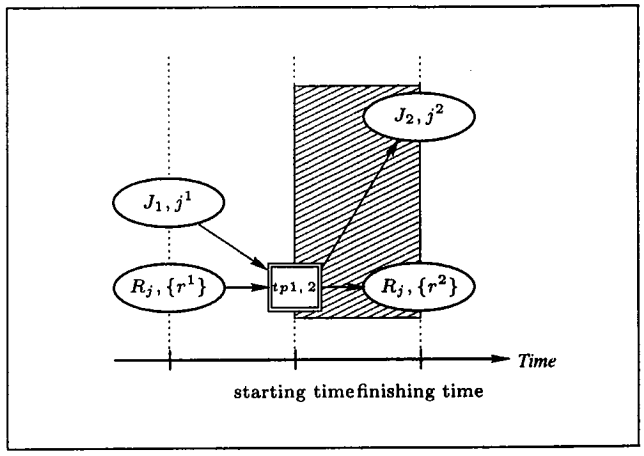

Figure 3: Observable Firing of a Transport Process Transition

An example

The following example is a simplified excerpt from the workflow models, which have been identified within the analysis phase of ChariTime. Figure 4 shows the typical in-patient flow in cardiology. After admission in the ward, the patient gets a first stationary examination. Requests for diagnostic examinations occur in the framework of this stationary examination. After transportation to the proper diagnostic units, the examinations can take place. In the best case, the patient gets all necessary examinations one right after the other. This prevents superfluous transports. Other cases are that the patient has some requested examinations left but nevertheless returns to the ward or that all examinations have been finished. Therapy is not modeled in this context, since it is not in the focus of the project.

Figure 4 illustrates a very coarse view on patients' flow. Every process transition in this flow can be detailed by further workflow submodels. Actors and re-

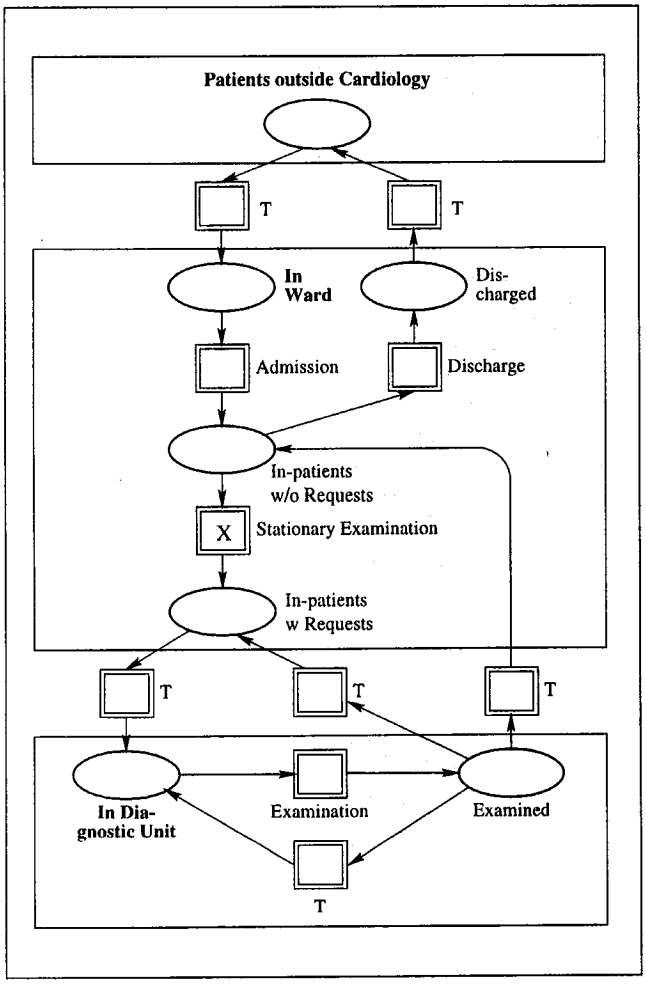

Figure 4: In-patient Flow

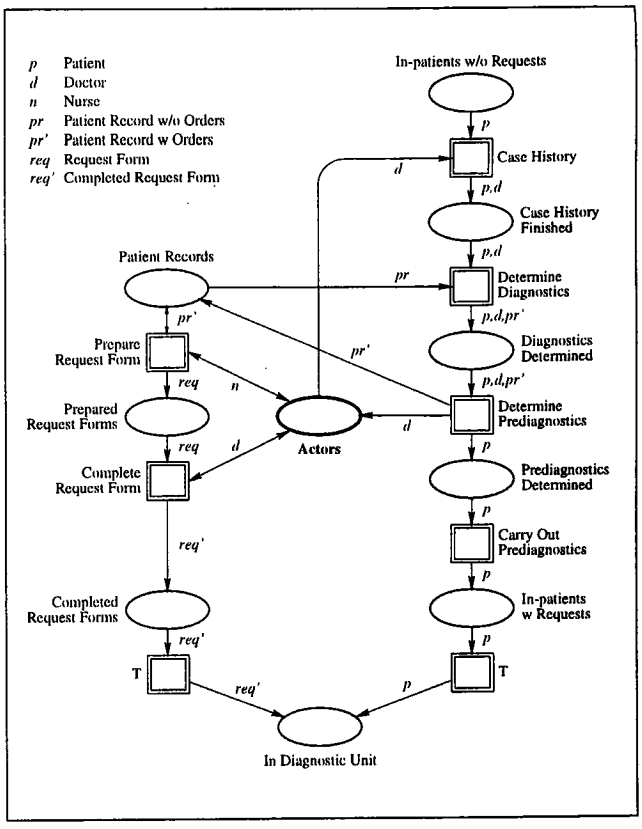

Figure 5: Stationary Examination

sources appear in these refinements. Figure 5 demonstrates the refinement of the stationary examination process. Central in this refinement are the active actors involved in the described processes. On the right the refined patient flow can be recognized, on the left the document flow has been added.

The first process in stationary examination is the taking down of the care history. This has to be done by a doctor. After this, the doctor determines certain diagnostic examinations, makes corresponding notes in the patient's record and determines additional prediagnostics. Concurrently to the prediagnostics a nurse picks out the patient record, which carries notes of ordered examinations. For every note she prepares a certain request form with the patient's name, address, weight and so on. After that, the prepared forms are handed over to the doctor, who completes the forms with medical information and signs them. They are transported to the diagnostic units, such that examinations can take place.

Identifying and Specifying Agents

After formally describing the observable workflows and thus analyzing the present situation, one has to specify the functionality, which shall be covered by the envisaged system. This is commonly known as requirements analysis. Focusing on the development of Multi Agent Systems, the functionality is encoded in agents skills. Since agents are representatives of human or organizational actors in this context, agent skills can be directly derived from *actor skills*. Actor skills denote the role competency of an actor to execute a given task within a business process. That means theoretically the competency of an actor to dynamically bind a process transition in the job net to a process transition in the system net.

Identifying Agents and Skills

Actor skills can be identified from the formal workflow model as follows:

- Consider a process transition p in the system net. Every actor, that can be found on the incoming arcs of p, is involved in the process P represented by p. Hence, he/she needs a certain skill to participate in the process. Such, for every actor, participating in P, an actor skill is identified, e. g. "Prepare Request Form".

- For further refinement of the actor skills, one takes a look at other specifying facts, like: Is the actor active of passive? In case of several involved actors, which interactions are implied by the common process P? Is the actor a client or a server in P? Which resources are used in P? This leads to a more specific identification of actor skills, like "Prepare Request Form with Patient Record".

The simple enumeration of actor skills can be structured by assigning actor skills to their actors and denoting this assignment by *actor diagrams*, which are known from the *Unified Modeling Language*. Figure 6 shows such a structured set of actor skills, which can be derived from the workflow given by figure 5. The relation lines between the actor "Patient" and the skills are dashed, because the patient is rather passively involved.

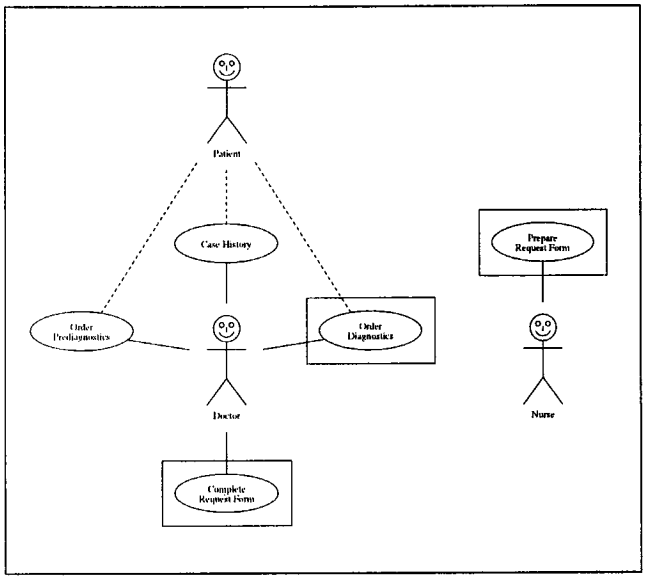

Figure 6: Actor Diagram for Stationary Examination

The envisaged functionality can now easily be specified by marking those actor skills, which shall be supported or automated by the system. In figure 6 this is done by rectangles. This method is highly incremental, since one may decide to realize the most important actor skills first, while the support of other actor skills may follow later. Those actor skills, which shall be supported, are then usefully grouped together and thus define agents and their functionality.

A good starting model for grouping supported actor skills to agents is the actor itself. That could mean, that the concrete actor is represented by a unique agent in the system, e. g. a personal agent for nurse Angela. Though, in most domains, actors with the same roles have similar skills. Hence, roles may define actor skill collections and thus agents better, e. g. a general nurse agent. Another influence factor in identifying agents is the quantity and quality of the supported skills. If there are many easy skills to support a grouping in a higher order agent, which represents a whole organizational actor, may be useful, e. g. a ward agent. If there is a very complex skill, which for example enforces the use of a sophisticated planning or scheduling algorithm, it may be in good keeping of a single specialized agent, e. g. a planner agent. The grouping of skills to agents should result in a good compromise between fine-grained "stupid" agents and a monolithic "big headed" agent.

Specifying Agents and Skills

Once agents and their skills have been identified, they can be specified concurrently and incremental. While objects commonly encapsulate data and methods, agents encapsulate **skills**, **knowledge** and **control**. These three components have to be defined to specify an agent.

The specification of skills consists mainly of the description of their atomic *activities* and the ordering relation between them. The complexity of activities corresponds to that of common methods. Though activities are domain dependent, they are similar in most domains. Typical activities are the sending of a certain message, the waiting for a message, the query of some data or the output of data on peripheral devices. Activities should not be complex, because complex behavior is realized by skills. To identify activities the observation of the problem domain can once again help. All elementary activities in the framework of a process should be analyzed. That includes the use of legacy systems and real-world communication protocols.

Activities, which could be identified for the skill "Prepare Request Form with Patient Record" could be querying the patient record, getting the right forms, putting patient information on the forms, adapting the state of the patient record and passing the prepared forms to the doctor. The description of the activities is intentionally vague, because specification does not prescribe the use of certain technologies.

The specification of knowledge is similar to the identification of domain classes in object-oriented methods. Hence, there are many heuristics for finding domain classes, which belong to the skill, that has to be specified. A good strategy is the analysis of used resources, like technical equipment, forms or media. The specification of knowledge should not include detailed information about the representation of the knowledge, since this is part of the design phase. In the presented example, knowledge can be identified by the domain classes `patient_record`, `form_A`, `form_B` and so on.

The statement on the quantity and quality of the agents skills mentioned earlier holds for agent control or architecture, too. Though specification determines the *what* of the given objective and not the *how*, the nature of the agents skills directly influences the choice of an appropriate control architecture. Simple skills with the need for timely execution may imply the use of a reactive agent architecture. On the other hand, complex skills, which involve planning or scheduling activities,

may enforce the use of a fully deliberative agent architecture. Hybrid architectures, like the BDI-architecture ((Rao and Georgeff 1995)), may be promising for unifying both demands. A first decision for a certain agent architecture can only be done, when all skills of the agent are known and specified. Hence, there is no way to determine the right control architecture for our toy example, since it is not complete.

Conclusion

In this paper I have presented a method for identifying and specifying intelligent agents and their functionality from formal models of the present workflow. Following this method, the static and dynamic aspects of the workflow and the specification of the control objective are described by a unique notation based on algebraic types and algebraic Petri Nets. From this description actor skills can be derived. Actor skills, that shall be supported by the envisaged system, are grouped to agents following the example of real actors, roles, organizational units or functionality. Agents are further specified by determination of the atomic activities of their skills, the knowledge, which is needed to perform the skills, and the agent control architecture.

The use of the presented method has been illustrated by a small example out of a real-world case study in medical administration. This case study is currently taken out at the cardiological clinic of Charité Berlin. The analysis and specification phase is nearly finished and the design phase has just begun. The results presented in this paper have been successfully used to find a lead-in to the realization of a distributed Multi Agent System, which is supposed to control request-and-serve relations in diagnostic examination. Further reports on this case study will follow.

References

Aalst, W. M. P. van der 1997. Exploring the process dimension of workflow management. Technical Report, Computing Science Reports 97/13, Eindhoven University of Technology.

Allen, J.; Kautz, H.; Pelavin, R. and Tenenberg, J. eds. 1990. *Reasoning about Plans.* Morgan Kaufmann Publishers.

Burmeister, B.; Bussmann, S.; Haddadi, A. and Sundermeyer, K. 1996. Agent-Oriented Techniques for Traffic and Manufacturing Applications: Progress Report. In (Jennings and Wooldridge 1998):161-174.

Cockburn, A. 1997. Goals and Use Cases. *Journal of Object-Oriented Programming* 10(5):35–40.

Franklin, S. and Graesser, S. 1996. Is it an Agent, or just a Program: A Taxonomy for Autonomous Agents. In Proceedings of the 3rd International Workshop on Agent Theories, Architecture, and Language (ATAL):193–206.

Graw, G. and Gruhn, V. 1995. Distributed Modeling and Distributed Enaction of Business Processes. In Schäfer, W. and Botella, P. eds. Proceedings of the 5th European Software Engineering Conference (ESEC '95):8–27.

Hewitt, C. 1988. Offices are Open Systems. In Bond, A. H. and Gasser, L. eds. *Readings in Distributed Artificial Intelligence*:321–329. Morgan Kaufmann Publishers.

Huang, J.; Jennings, N. R. and Fox, J. 1995. An Agent-based Approach to Health Care Management. *Applied Artificial Intelligence: An International Journal* 9(4):401-420.

Huhns, M. N. and Singh, M. P. eds. 1998. *Readings in Agents.* Morgan Kaufmann Publishers.

Jablonski, S. 1995. *Workflow-Management-Systeme: Modellierung und Architektur.* Thomson Publishers.

Jennings, N. R.; Faratin, P.; Norman, T. J.; Brien, P. O.; Wiegand, M. E.; Voudouris, C.; Alty, J. L.; Miah, T. and Mamdani, E. H. 1996. ADEPT: Managing Business Processes using Intelligent Agents. In Proceedings of the BCS Expert Systems Conference.

Jennings, N. R. and Wooldridge, M. J. eds. 1998. *Agent Technology — Foundations, Applications, and Markets.* Springer.

Kis, T.; Neuendorf, K.-P. and Xirouchakis, P. 1997. Scheduling with Chameleon Nets. In Proceedings of PNSE '97.

Lawrence, P. 1996. *Workflow Handbook 1997.* John Wiley in assoc. with WfMC.

Liu, J.-S. and Sycara, K. P. 1998. Multiagent Coordination in Tightly Coupled Task Scheduling. In (Huhns and Singh 1998).

Rao, A. S. and Georgeff, M. P. 1995. BDI Agents: From Theory to Practice. In Lesser, V. ed. Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS):312–319. MIT Press.

Reisig, W. 1998. *Distributed Algorithms: Modeling and Analysis with Petri Nets.* Springer.

Sen, S. and Durfee, E. H. 1995. Unsupervised Surrogate Agents and Search Bias Change in Flexible Distributed Scheduling. In Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS '95):336–343.

Valk, R. 1995. Petri nets as dynamical objects. In Proceedings of the 1st Workshop on Object-Oriented Programming and Models of Concurrency.

Yokoo, M.; Durfee, E. H.; Ishida, T. and Kuwabara, K. 1998. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. *IEEE Transactions on Knowledge and DATA Engineering* 10(5).

Zimmermann, A. 1997. Modellierung und Bewertung von Fertigungssystemen mit Petri-Netzen. Doctoral Diss., Fachbereich Informatik, Technische Universität Berlin.

Zweben, M. and Fox, M. eds. 1994. *Intelligent Scheduling.* Morgan Kaufmann Publishers.