Discovering distributed processes in supply chains

Laura Maruster, J.C. (Hans) Wortmann, A.J.M.M. (Ton) Weijters and W.M.P. (Wil) van der Aalst
Eindhoven University of Technology, Faculty of Technology Management, I&T Department, 5600 MB Eindhoven, The Netherlands.

Abstract
Processes such as tendering, ordering, delivery, and paying are executed by several parties in almost all supply chains. However, none of these parties has a proper overview over the whole set of activities executed. Therefore, none of the parties can take the lead in business process redesign. Business processes are often not described in an explicit manner, and therefore they are not available for analysis. However, in the information system of each supply chain party, partial information about the business process are recorded. We claim that the overall distributed process can be induced, by using this partial information of all involved parties.

In this paper we present an overview of methods available to discover processes across supply chains, based on the assumption that there is a common point of reference at all involved parties, e.g. an order number. Such an induced or discovered process enables analysis across the supply chain, and can become an important tool to facilitate business process redesign in networked organizations.

Keywords
Supply chain, workflow management systems, process mining, Petri nets.

INTRODUCTION

Despite of the wide use of the term "supply chain", there is not much known about the functioning of particular supply chains in practice. It is like travelling in the mountains: nowhere there is a place from which all slopes and gorges can be seen. When on route, it is only possible to see the path followed from one turn to the next one. Similarly, none of the partners in a supply chain has complete overview of what activities are executed by which partners.

Especially when it comes to quantitative determination of how much costs are incurred per process in the supply chain, there is hardly any material available to each of the participants. When extending the notion of supply chains to virtual enterprises, the lack of factual knowledge is even larger.

This paper is organized as follows: in next Section we will illustrate the above problem in a real life case study. This leads to the conclusion, that empirical analysis of processes in distributed environments is barely needed. In particular, none of the parties in a supply chain has any insight in the costs induced at other parties. More surprisingly, we will claim that these parties do not even have insight in their own part of the processes involved. Following Section introduces the notion of *process mining* as a method to discover processes from empirical data. Remaining Sections provide an overview of process mining techniques and discuss the applicability in the supply chain context.

REAL-LIFE CASE STUDY

In a study on Supply Chain improvement (see Wouters et al. 1999) a supply chain of electrical installation in building and construction was investigated. Such a supply chain consists of manufacturers delivering products, wholesalers who make these products available to installers who are contracted by customers to provide electrical installations according to documents provided by architects. One of the results, describing the processes between manufacturers, wholesalers and installers, is depicted in Figure 1.

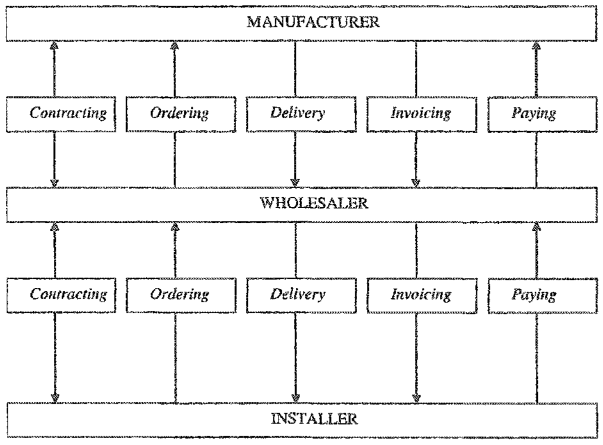

Figure 1 - Processes in the installation supply chain

Consider for example the contracting process in Figure 1. When an installer negotiates on products and work methods to be used for a particular prospect, the tendering will involve many requests for price quotations with the wholesalers. The wholesaler may, in turn, contact the supplier for technical specifications, possible delivery conditions, etc. However, the installer is not exactly aware of the activities triggered with the wholesaler and/or the manufacturer. For example, the installer may ask the same question to many wholesalers, and all these may contact various manufacturers for the same question on technical specifications. This is clearly a situation where the party which causes costs in the network (the installer) does not have visibility on the induced costs.

When the installer and a particular wholesaler are going to have as closer partnership, it is essential to be able to have a proper view on the costs encountered in these processes. It should be realized, that many organizations do not have their processes well described in process modelling formalisms. Rather, the work moves from one desk to another one, until a particular request is answered in a satisfactory way.

A first step towards proper workflow analysis would be, to make sure that the same *reference* (e.g. quotation number) is used throughout the supply chain in order to relate activities at partners to the same process identifier. When this is the case, then parties involved may start to collect data on the subsequent stages through which a particular process proceeds.

In the project performed with the installation supply chain, it was concluded that supply chain optimization is dramatically enhanced if partners understand each others processes and cost drivers. However, partners are extremely reluctant to allow each other to be enabled to calculate margins. Consequently, it is easy to convince the participants that quantitative analyses of processes make sense, but partners are generally unwilling to allow precise measurement of resource consumption. Therefore, joint process descriptions are welcomed by all parties in a supply chain, and participants are willing to submit descriptions of activities spent in the processes as depicted above in Figure 1, with clear reference to e.g. an identifier for tenders or orders.

INTRODUCTION TO PROCESS MINING

The managing of complex business processes calls for the development of powerful information systems, able to control and support the flow of work. These systems are called Workflow Management Systems (WfMS) and generally they are thought of as "a generic software tool which allows for definition, execution, registration and control of workflows" [1]. Despite the workflow technology promise, many problems are encountered when applying it. One of the problems is that these systems require a workflow model, i.e. a designer has to construct a detailed model accurately describing the routing of work. The drawback of such an approach is that a workflow model requires a lot of efforts from the workflow designers, workers and management, is time consuming and often subjective and incomplete.

The idea that we propose is to reverse the process; instead of designing a workflow model, why not collecting the information related to the process and inducing the underlying workflow model?

In the context of different partners which are participating in a supply chains of electrical installation, it means that the manufacturer, the wholesaler and the installer only needs to have a system of recording the process activities and to agree on a common order number and eventually on a common quotation numbers for the same type of processes. Subsequently, process-mining techniques can be used to reveal the overall distributed process. In the following subsection we provide a description of the process-mining problem.

What means process mining?

We use the term *process mining* for the method of distilling a structured process description from a set of real executions of a certain business process. To illustrate the idea of process mining, consider the process executions from Table 1.

Table 1 - An example of a process log

Case number	Executed tasks
Case 1	A F G H I K L
Case 2	A B C E J L
Case 3	A F H G I K L
Case 4	A F G I H K L
Case 5	A B C E J L
Case 6	A B D J L
Case 7	A B C E J L

In this example, there are seven cases that have been processed and twelve executed tasks. We can notice the following: for each case, the execution starts with task A and ends with task L, if C is executed, then E is executed. Also, sometimes we see task H and I after G and H before G. We assume that it is possible to record events such that (i) each event refers to a task, (ii) each event refers to a case and (iii) events are totally ordered. We call this information history the *process log*. The process log is used to construct a process specification, which adequately models the registered behaviour. Using the information shown in Table 1, we can discover the process model depicted in Figure 2. To represent the model a Petri net is used [1]. More details about the Petri net formalism are given later.

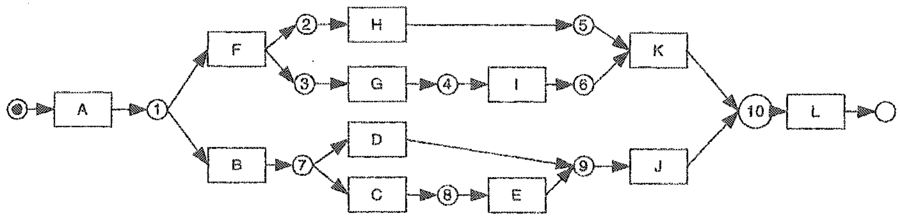

Figure 2 - A process model for the process log shown in Table 1

The Petri net from Figure 2 starts with task A and finishes with task L. In the Petri net formalism, all tasks are represented as transitions (marked as rectangular blocks). After executing A, either task B or task F can be executed. If task F is executed, tasks H and G can be executed in parallel. A parallel execution of tasks H and G means that they can appear in any order. In the Petri net formalism, the parallel construct "after task F, tasks H **and** G are executed in parallel" is represented with the aid of two circles called "places", labelled "2" and "3". The choice construct "after task B, task D **or** task C are executed" is represented with only one place, labelled "7".

From the information given in Table 1 is relatively simple to construct the Petri net from Figure 2. In the case of real-world processes where much more tasks are involved and with a high level of parallelism, the problem of discovering the underlying process becomes very complex. In our process log example from Table 1, it is easy to detect that the direct successors for task A are tasks B and F and for task B are tasks C and D. After task A, either task B or task F can be executed, thus there is a choice between B and F, that can be easily detected (the same holds for task B, where a choice can be made between the execution of C or D). However, in case of parallel tasks, it is not so easy to decide which the direct successor is. For example, for the log presented in Table 1, which is the direct successor for task G: I or H?

In practical situation it seems realistic to assume that workflow logs contain noise. Different situations can lead to noisy logs, like missing registration data or input errors. Moreover, logs can be incomplete. A log is *incomplete* if:

- The process that generates the log is too complex. In such a situation, it is possible that too few information are registered.
- There are tasks or combinations of tasks with a very long execution time. Let's consider the example presented in Figure 2. If we suppose to be always the case that event H is processed in 1 time unit, event G in 3 time units and I in 2 time units and H always finishes its execution before I starts, then we will always see the sequence "AFHIGKL" and never the sequence "AFGIHKL". Thus, we will not have that log information from where to determine that H is directly followed by K.
- Unbalance can exists between the probabilities of task executions. In Figure 2, after task A is executed, either task B or F can be executed. A task execution unbalance means, for example, that after task A, in 80% of the cases task B will be executed and only in the rest of 20% of the cases, task F will be executed.

In later section we will present some process mining methods which try to provide solutions in case of noise and incomplete logs.

Related work

The idea of process mining is not new; it is already applied in different domains as software engineering processes [4-6] and workflow management [3, 7-10, 15].

Cook and Wolf have investigated similar issues in contexts of software engineering processes. They provide methods for process discovery in case of sequential processes [4], concurrent processes [5] and also a technique which display the differences between the process model and the real behaviour [6]. In [4], Cook and Wolf describe three methods for process discovery in case of software engineer processes: one using neural networks, one using a purely algorithmic approach and one Markovian approach. These authors consider the latter the two the most promising approaches. The purely algorithmic approach builds a finite state machine where states are fused if their futures (in terms of possible behaviour in the next k steps) are identical. The Markovian approach uses a mixture of algorithmic and statistical methods and is able to deal with noise. Note that the results presented in [4] are limited to sequential behaviour. Cook and Wolf extend their work to concurrent processes in [5]. They propose specific metrics (entropy, event type counts, periodicity, and causality) and use these metrics to discover models out of event streams. However, they do not provide an approach to generate explicit representations for a broad rage of process models, i.e. we want to be able to generate a concrete Petri net rather than a set of dependency relations between events. In [6] Cook and Wolf provide a measure to quantify discrepancies between a process model and the actual behaviour as registered using event-based data.

The idea of applying process mining in the context of workflow management was first introduced in [3]. This work is based on workflow graphs, which are inspired by workflow products such as IBM MQSeries workflow (formerly known as Flowmark) and InConcert. In this last mentioned paper, two problems are defined. The first problem is to find a workflow graph generating events appearing in a given workflow log. The second problem is to find the definitions of the edge conditions. In [15], a tool based on these algorithms is presented. Herbst and Karagiannis also address the issue of process mining in the context of workflow management [7-10]. The approach uses the ADONIS modelling language and is based on hidden Markov models where models are merged and split in order to discover the underlying process. The work presented in [7, 9, 10] is limited to sequential models. A notable difference with the other approaches is that the same activity can appear multiple times in the workflow models. The result in [8] incorporates concurrency but also assumes that workflow logs contain explicit causal information. The latter technique is similar to [3, 15] and suffers from the drawback that the nature of splits and joins (i.e. AND and OR relations) is not discovered.

As we can observe from the mentioned literature, a robust process-mining method should fulfil certain requirements:

- Be able to detect concurrency. Generally, business processes are characterized by parallel processes. Thus, the algorithms which are limited to discover only sequential processes are not suitable to business processes.
- Detect AND/OR relations explicitly and to model these relations with the aid of an appropriate modelling formalism.
- Reflect the behaviour as registered in the process log.
- The resulting model to be as simple as possible.

We try to develop methods that respect these four requirements. As modelling formalism we chose the Petri nets. Petri nets provide a graphical but theoretical robust founded language for modelling concurrency. Petri nets (PN) have been successfully used to model and analyze processes from many domains, like for example, software and business processes. Processes can be modelled by WF nets, which form a subclass of PN [1]. In the following subsection we provide some basics about Petri nets.

Workflow models and workflow nets

A classical Petri net is a directed graph with two kinds of nodes, *places* and *transitions*, where *arcs* connect a place to a transition or a transition to a place. Each place can contain zero, one or more tokens. The state of a classical PN is determined by the distribution of tokens over places. A transition can fire if each of its input contains tokens. If the transition fires, i.e. it executes, it takes one token from each input place and puts one token on each output place.

Workflows are *case* oriented, which means that each *activity* executed in the workflow corresponds to a case. For example, a *case* corresponds with an insurance claim. The process definition of a workflow assumes that a partial order exists between activities, establishing the execution order of the activities. Referring to the Petri net formalism, workflow activities are modelled as transitions and the causal dependencies between activities are modelled as places and arcs.

A workflow net (WF) net is a classical PN with one *source* place (i.e. a place without incoming arcs), that represents the beginning of the case in the workflow, and a *sink* place (i.e. a place without outgoing arcs), which represents the end of the case in the workflow. Each transition and place in the WF net is on a path from source place to sink place.

The routing in a workflow assumes four kinds of routing constructs: *sequential*, *parallel*, *conditional* and *iterative* routing [1]. *Sequential* routing concerns ordered causal relationships between tasks. For example, if we consider tasks A and B, we have a sequential routing construct when task B is executed only after task A is executed. *Parallel* routing is used when the order of execution is less strict. A parallel routing is modelled by *AND-split* and *AND-join* blocks. An AND-split corresponds to a transition with two or more output places and an AND-join corresponds to a transition with two or more input places. *Conditional* routing allows the modelling of a choice between two or more alternatives. To express the conditional construct, *OR-split* and *OR-join* blocks are used. An OR-split corresponds to two or more alternative output transitions and an OR-join corresponds to two or more alternative input transitions. For example, in Figure 2 we can identify the following routing constructs: transitions *H* and *G* are AND-splits, *K* is an AND-join, the places *1* and *7* (represented as circles) are OR-splits and *9* and *10* are OR-joins.

OVERVIEW OF PROCESS MINING TECHNIQUES

One of our first process mining results are based on some empirical experiments. We observed that simple process mining techniques work in case of some sorts of Petri nets and do not in case of others. A very natural question arose: which kind of processes can be rediscovered? Subsequently, we split the process mining problem in two approaches:

1. The "theoretical" approach: find the classes of workflow models for which it is possible to accurately construct the model by merely looking at their logs, assuming noise free and complete log.
2. The "practical" approach: given a process log, construct the underlying Petri net, (i) in the presence of noise and (ii) incomplete log.

The theoretical approach is very useful for providing insights with respect to the limitations of the process mining methods also in case of the practical approach. In the following two subsections we provide an overview of both theoretical and practical approaches.

The theoretical approach

Our experiments started with the construction of different WF-nets and for each WF-net we generated a workflow log [11]. We showed that in case of sound and acyclic workflow nets involving parallel, conditional and sequential construct, our method was able to rediscover the original WF-nets. However, in case of a not-free choice WF-net the method was not able to find all connections between events.

Subsequently, in [2] we formulated the "theoretical approach" of the process mining problem as a twofold objective. First, assuming the ideal situation without noise, we are looking for a mining algorithm that is able to rediscover sound WF-nets, i.e. based on a complete workflow log the corresponding workflow process can be derived. *Sound* WF-nets are those WF-nets where no deadlocks, infinite loops, etc. can happen, thus the soundness property insures a correct behaviour of the WF-net. To reason about the quality of a workflow mining algorithm, we need to make assumptions about the completeness of a log. For a complex process, a handful of traces will not suffice to discover the exact behaviour of the process. Thus, a workflow is *complete* if all tasks that potentially directly follow each other in fact directly follow each other in some trace in the log. Second, given such an algorithm we want to indicate the class of workflow nets which can be rediscovered, which of course, should be as large as possible.

We found a mining algorithm for which it was possible to prove that: given a complete workflow log and assuming zero noise, it is possible to rediscover sound Petri nets, with the requirements that the underlying Petri net should belong to the class of structured Petri nets (SWF) (for more details about structured Petri nets, see [2]).

Although such a result can leave the impression that only a small class of Petri nets can be rediscovered, we explained that from a practical point of view it is not the case. First, SWS-nets allow for all routing constructs encountered in practice, i.e. sequential, parallel, conditional and iterative routing are possible and the basic workflow building blocks (AND-split, AND-join, OR-split and OR-join) are supported. Second, workflow nets that are not SWF-nets are typically difficult to understand and should be avoided if possible.

Although there are necessary some requirements in order to prove that the class of SWF-nets can be rediscovered on the basis of a complete workflow log, the applicability is not limited to SWF-nets. We also showed in [2] that in many situations a behaviourally equivalent workflow net can be derived. Even in the cases when the resulting workflow net is not behaviourally equivalent, it is meaningful and captures most of the behaviour, thus provides insights into the considered process.

Also, we made the remark that these theoretical results are consistent with other empirical results. For example, the fact that workflow process exhibiting non-free choice behaviour are difficult to mine was observed both in theory and practice [2, 11, 13].

The practical approach

The conclusions provided by the theoretical approach are very useful because we become aware of some of the possible limitations of the process mining problem, (e.g. the classes of WF-nets that cannot be mined correctly). However, one of the "strongest" assumption in the case of the theoretical approach is the absence of noise and the completeness of the process log, which is very difficult to find in practice. Therefore, in the practical approach, we concentrate on situations with noise and incomplete information in the process log.

We present two heuristic methods. The first method is using a parameter which is set in function of the amount of noise. The second method is going in the same direction: it is based on a learning approach for finding a threshold value which is used to detect for each task its direct successors.

The first heuristic method

In [13] is presented a method where the inducing of the WF-net depends on a parameter set in function of noise level. The method is based on (i) the dependency/frequency table and (ii) the causality metric. The *dependency/frequency table* (D/F-table) contains information about frequencies of events, frequencies of an event directly following all the other events, etc. The *causality* metric indicates the strength of the causality relation between two events and is calculated based on the information provided in the D/F table. The WF-net is constructed considering three rules where the causality metric and the noise level should exceed some threshold values. The experimental results shown that the method was able to discover the WF-net when the noise parameter varied. Thus, it seems that the method was robust in case of noise for the considered WF-nets. However, in case of non-free choice WF-nets, this heuristic method was not able to discover all connections, which is again in line with our theoretical results.

The results from [13] were preliminary; therefore we considered that more experimental work should be done in order to provide a robust method for process mining when noise is present. Such a method is explained in the next subsection.

The second heuristic method (the learning approach)

Performing more experiments, we remarked that the previous method based on the noise parameter and the threshold value was not providing always good results. We tried to improve it by developing a learning method for finding automatically a threshold that can be used to induce the WF-net.

Our idea was to use the same information from the D/F table and to develop additional metrics that can help to discover a process from the workflow logs. Subsequently, we split the process mining problem in two subproblems: (i) finding the direct successors for each event and (ii) finding the AND/OR relation between a pair of two events. For solving the two mentioned subproblems, we planned to use a global learning approach, namely to develop a logistic regression model. Once we know for all events their direct successors and the AND/OR relations between events, we can construct the WF-net.

In [12], we provided a method for solving the first process mining subproblem. Namely, given a workflow log, we have to find the direct successor task(s) for each event, (i) in the presence of noise and (ii) incomplete log.

In addition to the causality metric already introduced (see [13]), we built two new metrics which express the succession relation: a local metric and a global metric. The *local* metric is comparing the frequency of an event X directly preceded by an event Y with the frequency of event Y directly preceded by event X. The *global* metric compares the two frequencies mentioned before by taking into account the overall frequencies of events X and Y. The three metrics, i.e. the causality metric and the local and global metrics were used to develop a logistic regression model that can be used to predict when two events are in the direct succession relation. In order to develop the logistic regression model, a learning material was produced, by varying the number of event types, the amount of information contained in the workflow log, the amount of noise and the unbalance of task execution probabilities.

In conclusion, the method was able to find almost all direct connections in the presence of parallelism, unbalance and noise. Also, we tested our model on a workflow log generated by a more complex Petri net than the learning material, resulting in a close performance to the learning material. The next step is to use the same learning approach for finding the AND/OR relations between two events, which is actually an ongoing work. In this way, we are able to build the WF-net which is our final goal.

APPLICABILITY IN THE CONTEXT OF SUPPLY CHAIN OPTIMIZATION

Let us now return to the problem with which this paper started: redesigning business processes in networked enterprises. Recall that each participant in a supply network has very limited information on the process steps executed with his suppliers, customers, or subcontractors. In other words, no party in the supply network has proper view on the whole set of activities related to e.g. a quotation, an order, or a shipment.

On the other hand, experience as described in [14] learns the following. Each chain, i.e. each business process involving a particular manufacturer, wholesaler, and installer, when analyzed in detail, showed considerable opportunity for improvement.

For example, ordering by wholesalers to manufacturers turns out to be much lower than ordering costs by installers to manufacturers. This is due to the fact that wholesalers have optimized their logistics and installers have optimized their engineering and installing activities. However, this does not mean that it always makes sense to have the shipments physically flow through the warehouse of the wholesaler. It could be more attractive to move the goods directly from the manufacturer to the construction site where the installer is working. Such ideas require careful analysis of what happens in practice, not only in terms of financial transactions, but also in terms of frequencies of information exchange and cost driving activities. It took many man-months of work to investigate such facts in the project reported in [14], and the results were sometimes unsatisfactory because the ultimate insight in the complete business process could not be gained.

The idea is to provide a solution for collecting the necessary information about the whole business process, with less effort and with more reliable results, and to use efficiently these data. Our method consists on two basic steps. First, each part involved into the supply chain has to enable a task registration system that allows recording all the tasks or activities that have been executed into the process. An essential requirement is that all involved parts have to agree on a common order number and eventually on common quotation numbers for the same type of processes. Second, the process logs produced by the task registration system are used by the process mining techniques described in following sections and a Petri net for the whole distributed process is induced. Further on, the obtained model can be analyzed by the involved parties and eventually changes can occur in order to optimize the supply chain and to diminish the overall process costs.

By using such a method we expect that the supply chain activity can be improved because the parties (i) will benefit of a deeper understanding of the entire distributed process and (ii) they will be not obliged to allow that precise measurements for resource consumption that they are not willing to make them public.

CONCLUSIONS

There is considerable need in practice to discover business processes, which cross the boundaries of individual companies. Description of such processes in terms of frequencies and costs constitute a sound basis for supply chain optimization. It was argued, that such a discovery is by no means trivial.

An overview is given about new techniques that induce workflows from a workflow log, in which traces of individual workflows are shown, even if there is some noise in the data or the log is incomplete. However, in order to apply these techniques it is essential that companies involved in the supply chain use a common denominator for describing their activities, such as an order number. Only than it is possible to create Petri-nets, which fit the workflow log.

These techniques cannot determine distributed workflows in such a way that they completely cover all aspects of reality in all detail. The induced workflows cannot be better than the data they stem from. However, these workflows represent reality with sufficient precision to form a basis for redesign of cross-organizational processes. This is exactly the purpose they should serve!

REFERENCES

[1] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *J. of Circuits, Systems, and Computers*, 8(1): 21 - 66, 1998.

[2] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: which processes can be rediscovered? Beta publication, WP 75 2002, Eindhoven, The Netherlands.

[3] R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process models from Workflow Logs. In *Sixth International Conference on Extended Database Technology*, pg. 469 - 483, 1998.

[4] J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data, *ACM Transactions on Software Engineering and Methodology*, 7(3): 215 - 249, 1998.

[5] J.E. Cook and A.L. Wolf. Event-Based Detection of Concurrency. In *Proceedings of the Sixth International Symposium on the Foundations of Software Engineering* (FSE-6), Orlando, FL, November 1998, pp. 35 - 45.

[6] J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the correspondence of a Process to a Model. *ACM Transactions on Software Engineering and Methodology*, 8(2): 147 - 176, 1999.

[7] J. Herbst. A Machine Learning Approach to Workflow Management. In 11th European Conference on Machine Learning, volume 1810 *of Lecture Notes in Computer Science*, pages 183 - 194, Springer, Berlin, Germany, 2000.

[8] J. Herbst. Dealing with Concurrency in Workflow Induction In U. Baake, R. Zobel and M. Al-Akaidi, *European Concurrent Engineering Conf.*, SCS Europe, Gent, Belgium, 2000.

[9] J. Herbst and D. Karagiannis. An Inductive approach to the Acquisition and Adaptation of Workflow Models. In M. Ibrahim and B. Drabble, editors, *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, pg. 52 - 57, Stockholm, Sweden, August 1999.

[10] J. Herbst and D. Karagiannis. Integrating Machine Learning and Workflow Management to Support Acquisition and adaptation of workflow Models. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 9: 67 - 92, 2000.

[11] L. Maruster, W.M.P. van der Aalst, T. Weijters, A. van den Bosch, W. Daelemans. Automated discovery of workflow models from hospital data. In Kröse, B., de Rijke, M., Schreiber, G. and van Someren, M. (eds.): *Proceedings 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'01)*, 25-26 October 2001, Amsterdam, The Netherlands, pp. 183 - 190.

[12] L. Maruster, W.M.P. van der Aalst, A.J.M.M. Weijters and A. van den Bosch-Process mining: discovering direct successors in process logs, submitted to the *5th International Conference on Discovery Science 2002,* November 24-26, 2002, Lubeck, Germany.

[13] T. Weijters, W.M.P. van der Aalst. Process Mining: Discovering Workflow Models from Event-Based Data. In Kröse, B., de Rijke, M., Schreiber, G. and van Someren, M. (eds.): *Proceedings 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'01)*, 25-26 October 2001, Amsterdam, The Netherlands, pp. 283 - 290.

[14] M.J.F. Wouters, G.J. Sharman, and J.C. Wortmann. Reconstructing the Sales and Fulfillment Cycle to Create Supply Chain Differentiation. *International Journal of Logistics Management,* **10** (1999), no.2, p. 83 - 98

[15] M.K. Maxeiner, K. Kuspert, and F. Leymann. Data Mining von Workflow-Protokollen zur teilautomatisierten Konstruktion von Prozemodellen. In Proceedings of Datenbanksysteme in Buro, Technik und Wissenschaft, pages 75 - 84. Informatik Aktuell Springer, Berlin, Germany, 2001.