The ProM framework: A new era in process mining tool support

B.F. van Dongen, A.K.A. de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst

Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
b.f.v.dongen@tue.nl

Abstract. Under the umbrella of buzzwords such as "Business Activity Monitoring" (BAM) and "Business Process Intelligence" (BPI) both academic (e.g., EMiT, Little Thumb, InWoLvE, Process Miner, and MinSoN) and commercial tools (e.g., ARIS PPM, HP BPI, and ILOG JViews) have been developed. The goal of these tools is to extract knowledge from event logs (e.g., transaction logs in an ERP system or audit trails in a WFM system), i.e., to do *process mining*. Unfortunately, tools use different formats for reading/storing log files and present their results in different ways. This makes it difficult to use different tools on the same data sets and to compare the mining results. Furthermore, some of these tools implement concepts that can be very useful in the other tools but it is often difficult to combine tools. As a result, researchers working on new process mining techniques are forced to build a mining infrastructure from scratch or test their techniques in an isolated way, disconnected from any practical applications. To overcome these kind of problems, we have developed the ProM framework, i.e., an "plugable" environment for process mining. The framework is flexible with respect to the input and output format, and is also open enough to allow for the easy reuse of code during the implementation of new process mining ideas. This paper introduces the ProM framework and gives an overview of the plug-ins that have been developed.

1 Introduction

The research domain *process mining* is relatively new. A complete overview of recent process mining research is beyond the scope of this paper. Therefore, we limit ourselves to a brief introduction to this topic and refer to [3, 4] and the http://www.processmining.org web page for a more complete overview.

The goal of process mining is to extract information about processes from transaction logs. It assumes that it is possible to record events such that (i) each event refers to an *activity* (i.e., a well-defined step in the process), (ii) each event refers to a *case* (i.e., a process instance), (iii) each event can have a *performer* also referred to as *originator* (the actor executing or initiating the activity), and (iv) events have a *timestamp* and are totally ordered. Table 1 shows an example of a log involving 19 events, 5 activities, and 6 originators. In addition to the information shown in this table, some event logs contain more information on the

case itself, i.e., data elements referring to properties of the case. For example, the case handling system FLOWer logs every modification of some data element.

case id	activity id	originator	case id	activity id	originator
case 1	activity A	John	case 5	activity A	Sue
case 2	activity A	John	case 4	activity C	Carol
case 3	activity A	Sue	case 1	activity D	Pete
case 3	activity B	Carol	case 3	activity C	Sue
case 1	activity B	Mike	case 3	activity D	Pete
case 1	activity C	John	case 4	activity B	Sue
case 2	activity C	Mike	case 5	activity E	Clare
case 4	activity A	Sue	case 5	activity D	Clare
case 2	activity B	John	case 4	activity D	Pete
case 2	activity D	Pete			

Table 1. An event log (audit trail).

Event logs such as the one shown in Table 1 are used as the starting point for mining. We distinguish three different perspectives: (1) the process perspective, (2) the organizational perspective and (3) the case perspective. The *process perspective* focuses on the control-flow, i.e., the ordering of activities. The goal of mining this perspective is to find a good characterization of all possible paths, e.g., expressed in terms of a Petri net [14] or Event-driven Process Chain (EPC) [11, 12]. The *organizational perspective* focuses on the originator field, i.e., which performers are involved and how are they related. The goal is to either structure the organization by classifying people in terms of roles and organizational units or to show relation between individual performers (i.e., build a social network [2] and references there). The *case perspective* focuses on properties of cases. Cases can be characterized by their path in the process or by the originators working on a case. However, cases can also be characterized by the values of the corresponding data elements. For example, if a case represents a replenishment order, it is interesting to know the supplier or the number of products ordered.

Orthogonal to the three perspectives (process, organization, and case), the result of a mining effort may refer to *logical* issues and/or *performance* issues. For example, process mining can focus on the logical structure of the process model (e.g., the Petri net shown in Figure 1(a)) or on performance issues such as flow time. For mining the organizational perspectives, the emphasis can be on the roles or the social network (cf. Figure 1(b) and (c)) or on the utilization of performers or execution frequencies.

After developing *ad hoc* tools for the mining of the process perspective (e.g., EMiT [1] and Little Thumb [16]) and other *ad hoc* tools (e.g., MinSoN [2]) for the other mining perspectives we started the design of a flexible framework in which different algorithms for each of the perspectives can be plugged in.

2 Architecture

As indicated in the introduction, the basis for all process mining techniques is a *process log*. Such a log is a file generated by some information system, with

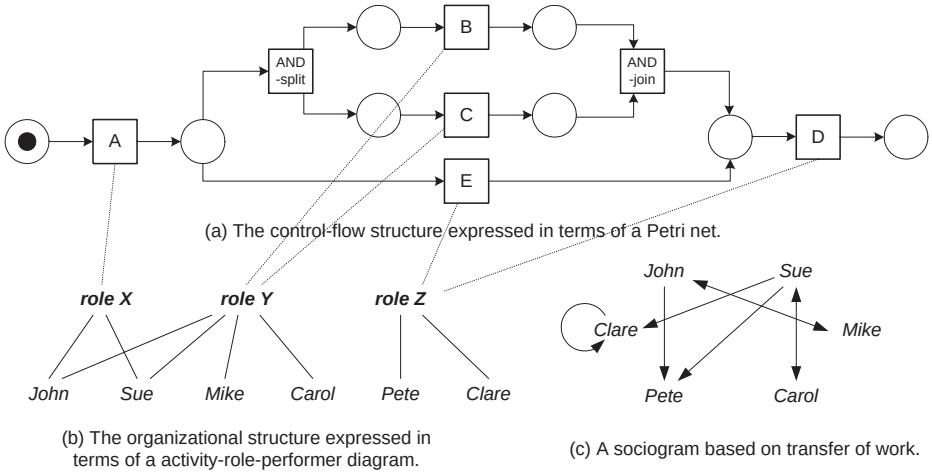

Fig. 1. Some mining results for the process perspective (a) and organizational (b and c) perspective based on the event log shown in Table 1.

information about the execution of a process. Since each information system has its own format for storing log files, we have developed a generic XML format for the ProM framework to store a log in. This format was based on a thorough comparison of the input needs of various existing (ad-hoc) process mining tools and the information typically contained in an audit trail or transaction log of some complex information system (e.g., an ERP or a WFM system).

Another important feature of the ProM framework is that it allows for interaction between a large number of so-called plug-ins. A plug-in is basically the implementation of an algorithm that is of some use in the process mining area, where the implementation agrees with the framework. Such plug-ins can be added to the entire framework with relative ease: Once the plug-in is ready it can be added to the framework by adding its name to some *ini*-file. Note that there is no need to modify the ProM framework (e.g., recompiling the code) when adding new plug-ins, i.e., it is a truly "plugable" environment.

In Figure 2, we show an overview of the framework that we developed. It explains the relations between the framework, the process log format, and the plug-ins. As the figure shows, the ProM framework can read files in the XML format through the *Log filter* component. This component is able to deal with large data-set and filter them before the actual mining starts. Through the *Import plug-ins* a wide variety of models can be loaded ranging from a Petri net to LTL formulas. The *Mining plug-ins* do the actual mining and the result is stored as a *Frame*. These frames can be used for visualization, e.g., displaying a Petri net [14], an EPC [12] or a Social network [2], or further analysis or conversion. The *Analysis plug-ins* take a mining result an analyze it, e.g., calculating a place invariant for a resulting Petri net. The *Conversion plug-ins* take a mining result and transform it into another format, e.g., transforming an EPC into a Petri net or vice versa. In the remainder of this section, we describe both the process log format and the plug-ins.

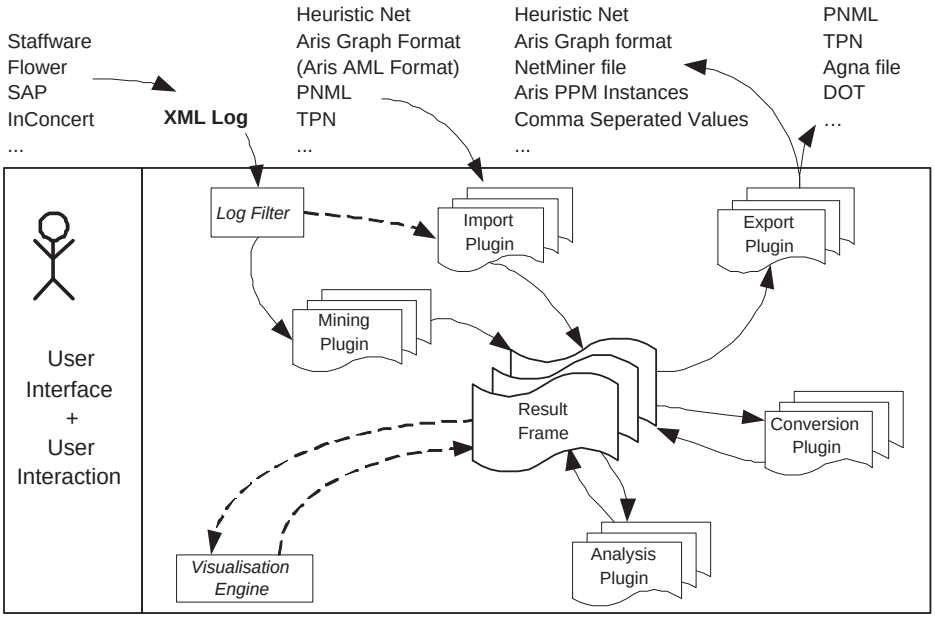

Fig. 2. Overview of the ProM framework

2.1 Process Log Format

Figure 3 visualizes the XML schema that specifies the process log format. The root element is a *WorkflowLog* element. (The name "workflow log" is chosen for backwards compatibility and we prefer to talk about process log.) The *WorkflowLog* element contains (in the given order) an optional *Data* element, an optional *Source* element, and a number of *Process* elements. A *Data* element allows for storing arbitrary textual data, and contains a list of *Attribute* elements. A *Source* element can be used to store information about the information system this log originated from. A *Process* element refers to a specific process in an information system. Since most information systems typically control several processes, multiple *Process* elements may exist in a log file. A *ProcessInstance* is an instance of the process, i.e., a case. An *AuditTrailEntry* may refer to an activity (*WorkflowModelElement*), an eventtype (*Eventtype*), a timestamp (*Timestamp*), and a person that executed the activity (*Originator*).

As will be clear from what was mentioned earlier, a log file typically contains information about events that took place in a system. Such events typically refer to a case and a specific activity within that case. Examples of such events are:

- The activity *send message* is now ready to be executed.
- The activity *wait for incoming transmission* has not been started for three weeks.
- The case with ID 203453 was aborted.

In order to be able to talk about these events in a standard way, we developed a transactional model that shows the events that we assume can appear in a

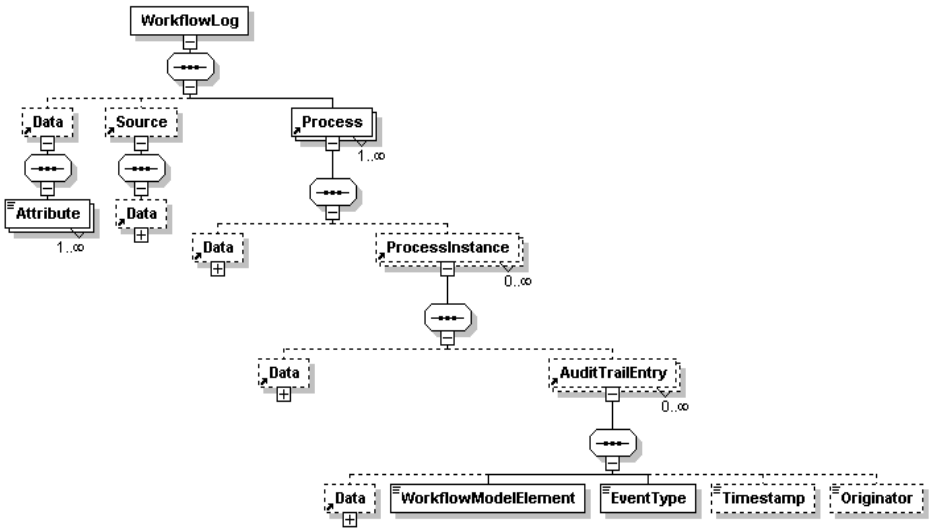

Fig. 3. Format of a process log

log. Again this model is based on analyzing the different types of logs in real-life systems (e.g., Staffware, SAP, FLOWer, etc.) Figure 4 shows the transactional model.

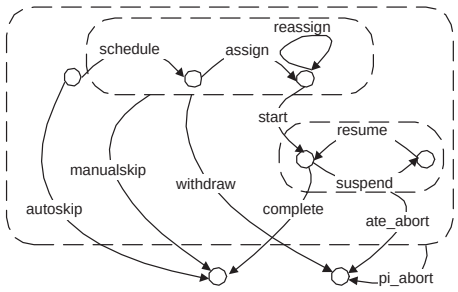

Fig. 4. Transactional model for activities

When an activity is created, it is either *schedule*d or skipped automatically (*autoskip*). Scheduling an activity means that the control over that activity is put into the information system. The information system can now *assign* this activity to a certain person or group of persons. It is possible to *reassign* an assigned activity to another person or group of persons. This can be done by the system, or by a user. A user can *start* working on an activity that was assigned to him, or some user can decide to *withdraw* the activity or skip it manually (*manualskip*), which can even happen before the activity was assigned. The main difference between a withdrawal and a manual skip is the fact that after the manual skip the activity has been executed correctly, while after a withdrawal it is not. The user that started an activity can *suspend* and *resume* the activity several times,

but in the end s/he either has to *complete* or abort (*ate_abort*) it. Note the activity can get aborted (*pi_abort*) during its entire life cycle.

We do not claim that we have captured all possible behavior of all systems. However, we have verified our transactional model against several commercial systems and they all seem to fit nicely. Nonetheless, in the XML format, we allow for other event types to be defined on the fly.

2.2 Plug-ins

In this section, we provide an overview of the plug-ins as currently implemented in the context of the ProM framework. As shown in Figure 2 there are five kinds of plug-ins:

Mining plug-ins which implement some mining algorithm, e.g., mining algorithms that construct a Petri net based on some event log.

Export plug-ins which implement some "save as" functionality for some objects (such as graphs). For example, there are plug-ins to save EPCs, Petri nets (e.g., in PNML format [7]), spreadsheets, etc.

Import plug-ins which implement an "open" functionality for exported objects, e.g., load instance-EPCs from ARIS PPM.

Analysis plug-ins which typically implement some property analysis on some mining result. For example, for Petri nets there is a plug-in which constructs place invariants, transition invariants, and a coverability graph. However, there are also analysis plug-ins to compare a log and a model (i.e., conformance testing) or a log and an LTL formula.

Conversion plug-ins which implement conversions between different data formats, e.g., from EPCs to Petri nets.

The current version of the framework contains a large set of plug-ins. A detailed description of these plug-ins is beyond the scope of this paper. Currently, there are nine export plug-ins, four import plug-ins, seven analysis plug-ins, and three conversion plug-ins. Therefore, we only mention some of the available *mining plug-ins*. For each of the three perspectives which were mentioned in the introduction, there are different mining plug-ins.

For the process perspective, four plug-ins are available:

α-algorithm which implements the α-algorithm [5] and its extensions as developed by the authors. The α-algorithm constructs a Petri net which models the process recorded in the log.

Tshinghua-α algorithm which uses timestamps in the log files to construct a Petri net. It is related to the α algorithm, but uses a different approach. It is interesting to note that this mining plug-in was the first plug-in developed by researchers outside of our research group. Researchers from Tshinghua University in China (Jianmin Wang and Wen Lijie) were able to develop and integrate this plug-in without any help or changes to the framework.

Genetic algorithm which uses genetic algorithms to tackle possible noise in the log file. Its output format is a heuristics net (which can be converted into an EPC or a Petri net).

Multi-phase mining which implements a series of process mining algorithms that use instance graphs (comparable to runs) as an intermediate format. The two-phase approach resembles the aggregation process in Aris PPM.

For the organizational perspective, one plug-in is available:

Social network miner which uses the log file to determine a social network of people [2]. It requires the log file to contain the *Originator* element.

Finally, for the case perspective, also one plug-in is available:

Case data extraction which can be used for interfacing with a number of standard *knowledge discovering tools*, e.g., Viscovery and SPSS AnswerTree.

Sometimes a collection of plug-ins is needed to achieve the desired functionality. An example is the *LTL-checker* which checks whether logs satisfies some Linear Temporal Logic (LTL) formula. For example, the LTL-checker can be used to check the "four eyes" principle, i.e., two activities within the same case should not be executed by the same person to avoid possible fraud. The LTL-checker combines a mining plug-in (to get the log), an import plug-in (to load the file with predefined LTL formulas), and an analysis plug-in (to do the actual checking).

3 User Interface

Since the ProM framework contains a large number of plug-ins, it is impossible to discuss them all in detail. Therefore, we only present some screenshots of a few plug-ins that we applied to the example of Table 1. In Figure 5, we show the result of applying the α-mining plug-in to the example. The default settings of the plug-in were used, and the result is a Petri net that is behaviorally equivalent to the one presented in Figure 1. In Figure 6, we show the result of the social network mining plug-in. We used the *handover of work* setting, considering only direct succession, to generate this figure. Comparing it to Figure 1(c) shows that the result is an isomorphic graph (i.e. the result is the same).

Petri nets are not the only modelling language supported by the framework. Instead, we also have built-in support for EPCs (Event-driven Process Chains). In Figure 7, we show the result of the multi-phase mining plug-in. The result is an aggregated EPC describing the behavior of all cases. Note that it allows for more behavior than the Petri net, since the connectors are of the type *logical or*. In Figure 8 we show the user interface of the analysis plug-in that can be used for the verification of EPCs.

In this section, we showed some screenshots to provide an overview of the framework. We would like to stress that we only showed a few plug-ins of the many that are available. We would also like to point out that most plug-ins allow for user interaction. The latter it important because process mining is often an interactive process where human interpretation is important and additional knowledge can be used to improve the mining result.

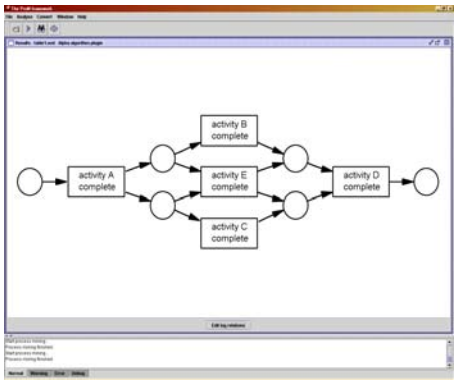

Fig. 5. The α-mining plug-in

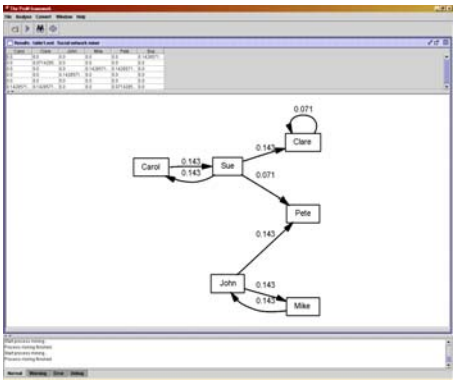

Fig. 6. The social network mining plug-in

4 Related work

Process mining can be seen as a tool in the context of Business Activity Monitoring (BAM) and Business (Process) Intelligence (BPI). In [9] a BPI toolset on top of HP's Process Manager is described. The BPI tools set includes a so-called "BPI Process Mining Engine". However, this engine does not provide any techniques as discussed before. Instead it uses generic mining tools such as SAS Enterprise Miner for the generation of decision trees relating attributes of cases to information about execution paths (e.g., duration). In [13] the PISA tool is described which can be used to extract performance metrics from workflow logs. Similar diagnostics are provided by the ARIS Process Performance Manager (PPM) [11]. The later tool is commercially available and a customized version of PPM is the Staffware Process Monitor (SPM) [15] which is tailored towards mining Staffware logs.[1]

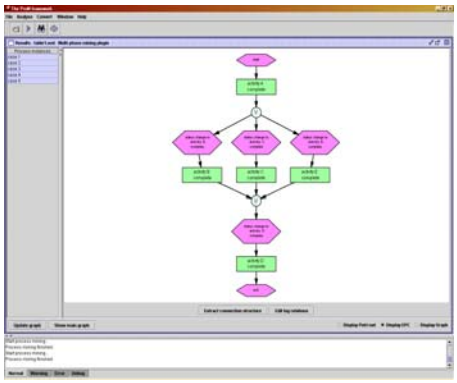

Fig. 7. The discovered EPC

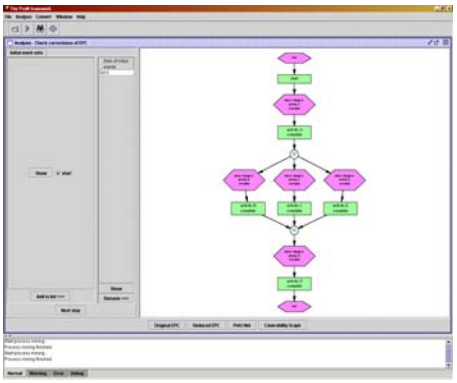

Fig. 8. Analyzing the EPC for correctness

[1] Note that the ProM Framework interfaces with Staffware, SPM, ARIS Toolset, and ARIS PPM.

Given the many papers on mining the process perspective it is not possible to give a complete overview. Instead we refer to [3, 5]. Historically, Cook et al. [8] and Agrawal et al. [6] started to work on the problem addressed in this paper. Herbst et al. [10] took an alternative approach which allows for dealing with duplicate activities. The authors of this paper have been involved in different variants of the so-called α-algorithm [1, 5, 16]. Each of the approaches has its pros and its cons. Most approaches that are able to discover concurrency have problems dealing with issues such as duplicate activities, hidden activities, non-free-choice constructs, noise, and incompleteness.

The ProM framework subsumes process mining tools like EMiT [1], Little Thumb [16] and MinSon [2]. Most of these tools had their own format to store log files in, and had their own limitations. The tool EMiT for example was unable to deal with log files of more than 1000 cases. To be able to use all these tools together in an interactive way, we developed the ProM framework, which can be seen as a successor of all these tools. The framework allows researchers to seamlessly combine their own algorithms with algorithms from other people. Furthermore, using the framework allows you to interface with many existing tools, both commercial and public. These tools include: the Aris Toolset, Aris PPM, Woflan, The Petri net kernel, Netminer, Agna, Dot, Viscovery, etc.

5 Conclusion

The ProM framework integrates the functionality of several existing process mining tools and provides many additional process mining plug-ins. The ProM framework supports multiple formats and multiple languages, e.g., Petri nets, EPCs, Social Networks, etc. The plug-ins can be used in several ways and combined to be applied in real-life situations. We encourage developers and researchers to use the ProM framework for implementing new ideas. It is easy to add a new plug-in. For adding new plug-ins it suffices to add a few lines to the configuration files and no changes to the code are necessary, i.e., new mining plug-ins can be added without re-compiling the source code. Experiences with adding the *Thingua alpha-plugin* and the *Social network miner* show that this is indeed rather straightforward.

6 Acknowledgements

The authors would like to thank all people that have been involved in the development and implementation of the ProM framework. In particular we would like to thank Minseok Song, Jianmin Wang and Wen Lijie for their contributions. Furthermore, we would like to thank IDS Scheer for providing us with Aris PPM and the Aris toolset. Last, but certainly not least, we would like to thank Peter van den Brand for doing the major part of the implementation work for us.

References

1. W.M.P. van der Aalst and B.F. van Dongen. Discovering Workflow Performance Models from Timed Logs. In Y. Han, S. Tai, and D. Wikarski, editors, *International*

Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002), volume 2480 of *Lecture Notes in Computer Science*, pages 45–63. Springer-Verlag, Berlin, 2002.

2. W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering interaction patterns in business processes. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 244–260. Springer-Verlag, Berlin, 2004.
3. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
4. W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
5. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
6. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
7. J. Billington and et. al. The Petri Net Markup Language: Concepts, Technology, and Tools. In W.M.P. van der Aalst and E. Best, editors, *Application and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 483–506. Springer-Verlag, Berlin, 2003.
8. 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.
9. D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. In P. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. Snodgrass, editors, *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 159–168. Morgan Kaufmann, 2001.
10. J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.
11. IDS Scheer. ARIS Process Performance Manager (ARIS PPM): Measure, Analyze and Optimize Your Business Process Performance (whitepaper). IDS Scheer, Saarbruecken, Gemany, http://www.ids-scheer.com, 2002.
12. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
13. M. zur Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
14. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
15. Staffware. Staffware Process Monitor (SPM). http://www.staffware.com, 2002.
16. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.