A Meta Model for Process Mining Data

B.F. van Dongen 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,w.m.p.v.d.aalst}@tue.nl

Abstract. Modern process-aware information systems store detailed information about processes as they are being executed. This kind of information can be used for very different purposes. The term *process mining* refers to the techniques and tools to extract knowledge (e.g., in the form of models) from this. Several key players in this area have developed sophisticated process mining tools, such as Aris PPM and the HP Business Cockpit, that are capable of using the information available to generate meaningful insights.
What most of these commercial process mining tools have in common is that installation and maintenance of the systems requires enormous effort, and deep knowledge of the underlying information system. Moreover, information systems log events in different ways. Therefore, the interface between process-aware information systems and process mining tools is far from trivial. It is vital to correctly map and interpret event logs recorded by the underlying information systems. Therefore, we propose a meta model for event logs. We give the requirements for the data that should be available, both informally and formally. Furthermore, we back our meta model up with an XML format called MXML and a tooling framework that is capable of reading MXML files. Although, the approach presented in this paper is very pragmatic, it can be seen as a first step towards and ontological analysis of process mining data.

1 Introduction

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., event logs in an ERP system or audit trails in a WFM system), i.e., to do *process mining*.

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 [2, 3] 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 event logs. It assumes that it is possible to record events such that (i) each

[⋆] This work is supported by the INTEROP Network of Excellence - Contract no.: IST-508 011 (http://www.interop-noe.org) 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), and (iii) events are totally ordered. Furthermore, all kinds of system-specific data elements can be present in these event logs. This immediately shows one of the biggest challenges faced in the process mining research. Each information system has its own internal data structure, and its own language to describe the internal structure. When trying to use event logs from different system to do process mining, we need to be able to present the logs in a standardized way, i.e., there is a need for a good description of such a log. Furthermore, for each information system, a mapping has to be provided onto that description. In other words, we need a meta model for process mining. In this paper, we take a first step towards such a process mining meta model.

Few of the meta models described in literature (e.g., [12]) focus on process mining. The work of Zur Muehlen [9] is closest to the results reported in this paper. However, our work is more pragmatic and driven by concrete tools and systems. The application of ProM, our process mining platform, provides us with insights that are valuable for people using BAM, BPI, and other process mining tools.

In practice, the mapping of event logs from one system to the standard format is a non-trivial task. It requires the mapping of one meta model onto another. It can be seen as a form of ontological analysis in the spirit of [4, 6, 11]. Instead of the Bunge-Wand-Weber ontology, we use a meta model that can be seen as a starting point for an ontology for process mining. Similar to the work in [4, 6, 11], "ontological goodness" and "ontological weaknesses" of process-aware information systems with respect to event logs can be analyzed by comparing the different meta models.

The remainder of this paper is organized as follows. In Section 2, we introduce process-aware information systems and we provide a high-level classification thereof. Then, in Section 3, we introduce an XML format (MXML) for storing event logs. In Section 4, we introduce the process mining framework that can work with MXML files. In Section 5, we show an example of a mapping between the the meta model of a widely-used information system (Staffware) to our meta model, and we show how this can be translated to a mapping of the log to MXML. In this section, we also provide an ontological analysis of the logging facilities of Staffware. Section 6, we touch some of the issues related to these mappings for other information systems. Finally, we discuss related work and conclude the paper.

2 Process-aware information systems

Process-aware information systems are widely used in practice (cf. ERP, WFM, CRM, PDM systems). At the basis of most of these systems lays a process model of some kind. However, the way systems enforce the handling of cases is different for all systems. On the one hand there are systems that enforce a given process description to all users, while some other systems only provide an easy way of handling access to files. As a result of this, information systems are used in very diverse organizations and with all kinds of expectations. Even though each system has its individual advantages and disadvantages, these systems can be divided in several groups. In Figure 1, we give four types of information systems, and position them with respect to the structure of the process that is dealt with and whether they are data or process driven. In Figure 2, we give the trade-offs that are made for each of these four types of systems with respect to flexibility, support, performance and design effort.

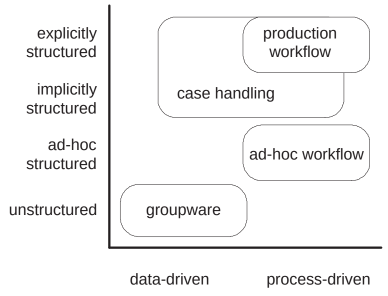

Fig. 1. PAIS spectrum

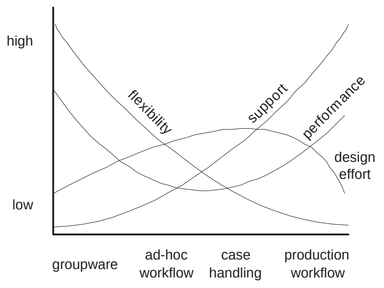

Fig. 2. PAIS tradeoffs

Production workflow systems such as for example Staffware are typically used in organizations where processes are highly standardized, and volumes are big (i.e. a lot of cases are to be dealt with in parallel). These systems not only handle data, but enforce a certain process definition to be followed by the letter. Case handling systems such as Flower on the other hand, are typically used in environments where people have a good understanding of the complete process. This allows these so-called "knowledge workers" to handle cases with more flexibility. In the end however, the case handling system keeps structure in both the data involved and the steps required. Ad-hoc workflow systems such as InConcert allow for the users to deviate completely from given processes. Processes definitions are still provided, but not enforced on an execution level. They merely serve as reference models. The final category of systems, i.e. groupware is the most flexible one. Systems such as Lotus Notes provide a structured way to store and retrieve data, but no processes are defined at all.

Due to the fact that each information system serves a different purpose, and that they are used in very different organizations, it is obvious that there is a difference in the internal data warehousing of those systems. In this paper, we are interested in the event logs that can be generated by process-aware information systems. Since the information in an event log highly depends on the internal data representation of each individual system, it is safe to assume that each system provides information in its own way. Therefore, we need to provide a standard for the information we need and mappings from each system to this standard. For this, we introduce MXML.

3 XML mining format MXML

As we stated in the introduction, there is a minimal amount of information that needs to be present in order to do process mining. In this section, we first give some requirements with respect to this information. From these requirements, we derive a meta model in terms of a UML class diagram. Then, we introduce a formal XML definition for event logs, called *MXML*, to support this meta model. We conclude the section with an example of an MXML file.

3.1 Requirements

All process-aware information systems have one thing in common, namely the process specification. For groupware systems, such a specification is nothing more than a unstructured set of possible activities, while for production workflows this specification is extremely detailed. For process mining, log files of such systems are needed as a starting point. First we give the requirements for the information needed.

To make the distinction between events that took place, and logged events, we will refer to the latter by *audit trail entries* from here on. When events are logged in some information system, we need them to meet the following requirements in order to be useful in the context of process mining:

1. Each audit trail entry should be an event that happened at a given point in time. It should not refer to a period of time. For example, starting to work on some work-item in a workflow system would be an event, as well as finishing the work-item. The process of working on the work-item itself is not.
2. Each audit trail entry should refer to one activity only, and activities should be uniquely identifiable.
3. Each audit trail entry should contain a description of the event that happened with respect to the activity. For example, the activity was started or completed.
4. Each audit trail entry should refer to a specific process instance (case). We need to know, for example, for which invoice the payment activity was started.
5. Each process instance should belong to a specific process.

Using the requirements given above, we are able to make a meta model of the information that should be provided for process mining.

3.2 Mining meta model

From the requirements given in the previous section, we derive the UML class diagram of Figure 3. Note that we use the term “Workflow Model Element” instead of activity, and “Workflow log” instead of event log. This is done for historic reasons.

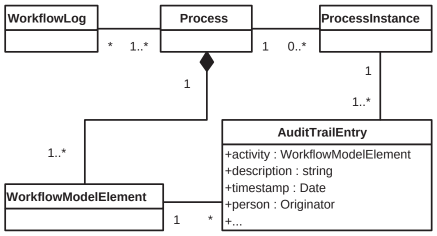

Fig. 3. Mining data meta model.

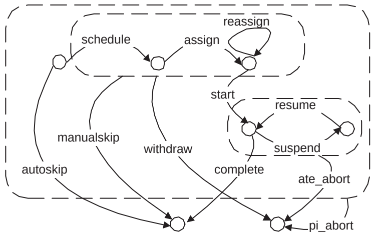

Fig. 4. Transactional model.

As we stated in the requirements, each audit trail entry contains a description of the event that generated it. In order to be able to talk about these events in a standardized way, we developed a transactional model that shows the events that we assume can appear in a log. This model is based on analyzing the different types of logs in real-life systems (e.g., Staffware, SAP, FLOWer, etc.) Figure 4 shows this transactional model.

When an activity (or Workflow Model Element) is created, it is either "scheduled" 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 he or she either has to "complete" or abort ("ate_abort") it. Note the activity can get aborted ("pi_abort") during its entire life cycle. Since we cannot claim that we have captured all possible behavior of all systems, we will have to allow for user defined events in the MXML format.

3.3 XML Structure

Using the meta model of Figure 3, we can easily derive an XML format for storing event logs. In Figure 5 a schema definition is given for the format that is used by our process mining framework ProM.

Most of the elements in the XML schema can be found in the meta model and they speak for themselves. There are however two exceptions. First, there is the "Data" element. This element allows for storing arbitrary textual data, and contains a list of "Attribute" elements. On every level, it can be used to store information about the environment in which the log was created. Second, there is the "Source" element. This element can be used to store information about the information system this log originated from. It can in itself contain a data element, to store information about the information system. It can for example be used to store configuration settings.

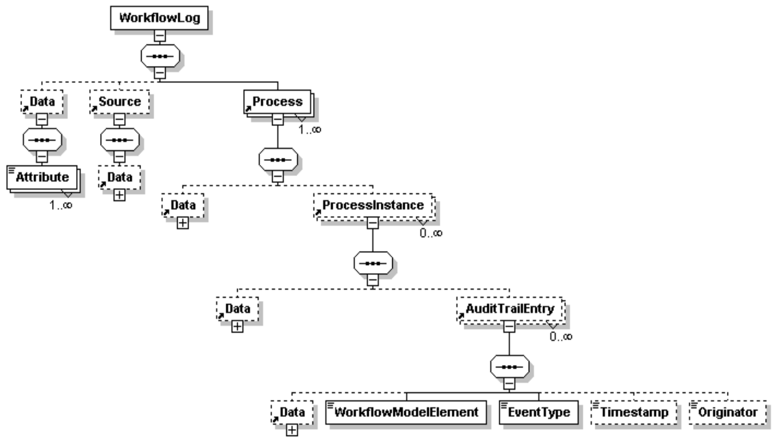

Fig. 5. XML mining format.

3.4 Example

We conclude the section about the MXML format with an example of an XML log file. This example log is the first part of a translation of a Staffware log to the MXML format (without the standard headers). In Section 5, we introduce this translation in more detail. It shows two audit trail entries in a complaints handling process.

```
<Source program="staffware">
    <Data>
        <Attribute name="version">7.0</Attribute>
    </Data>
</Source>
<Process id="main_process">
    <Data>
        <Attribute name="description">complaints handling</Attribute>
    </Data>
    <ProcessInstance id="Case 1">
        <AuditTrailEntry>
            <WorkflowModelElement>Case start</WorkflowModelElement>
            <EventType unknowntype="case_event">unknown</EventType>
            <Timestamp>2002-04-16T11:06:00.000+01:00</Timestamp>
        </AuditTrailEntry>
        <AuditTrailEntry>
            <WorkflowModelElement>Register complaint</WorkflowModelElement>
            <EventType>schedule</EventType>
            <Timestamp>2002-04-16T11:16:00.000+01:00</Timestamp>
            <originator>jvluin@staffw</originator>
        </AuditTrailEntry>
```

Table 1. A part of an MXML file.

4 ProM

Defining an XML format such as MXML would not make sense unless it is backed up by a good tool. For this, the ProM framework [5] has been developed. The ProM framework is a “pluggable” environment for process mining. 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. When dealing with log files, the framework requires them to be in the MXML format.[1]

[1] For more information about ProM, we refer to http://www.processmining.org.

The ProM framework can read log files in the MXML format. 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 [10], an EPC [8] or a Social network [1], or further analysis or conversion. The *Analysis plug-ins* take a mining result and 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.

Using the ProM framework, we have seen promising results with respect to the applicability of process mining in real business environments. In the next section, we present a small case study thereof.

5 Case: Staffware logs to MXML

In this section, we show that it is possible to actually extract MXML log files from commercial systems, since this greatly improves the practical applicability of the ProM framework. As a case study, we show that this is possible for a commercial workflow system called *TIBCO Staffware Process Suite* (in short Staffware). Through this example, we show why we call our model from Figure 3 a *meta*-model. We consider the data model used by Staffware as an instantiation of the meta-model. Then, the Staffware log file should be seen as an instantiation of the Staffware data model. By studying the mapping between the Staffware data model and our meta-model in Section 5.2 we give a translation from the Staffware log file to the MXML file.

5.1 A Staffware log file

In Table 2, we show an example of a Staffware log file. It consists of one complete case of a complaint handling process and a second, incomplete case of the same process.

```
Case 1
Diractive Description  Event         User           yyyy/mm/dd hh:mm
--------------------------------------------------------------------
                       Start         jvluin@staffw  2002/04/16 11:06
Register complaint     Processed To  jvluin@staffw  2002/04/16 11:16
Register complaint     Released By   jvluin@staffw  2002/04/16 11:26
Evaluate complaint     Processed To  jvluin@staffw  2002/04/16 11:36
Evaluate complaint     Released By   jvluin@staffw  2002/04/16 11:46
                       Terminated                   2002/04/16 11:56
Case 2
Diractive Description  Event         User           yyyy/mm/dd hh:mm
--------------------------------------------------------------------
                       Start         jvluin@staffw  2002/04/16 12:36
Register complaint     Processed To  jvluin@staffw  2002/04/16 12:46
Register complaint     Expired       jvluin@staffw  2002/04/17 13:07
Register complaint     Withdrawn     jvluin@staffw  2002/04/17 13:07
```

Table 2. A Staffware log file.

5.2 Mapping Staffware to MXML

In order to map Staffware logs to MXML, we fist need to map the internal data model of Staffware onto our model from Figure 3. In Figure 6, we show the Staffware data model, and we show how the mapping should be done.

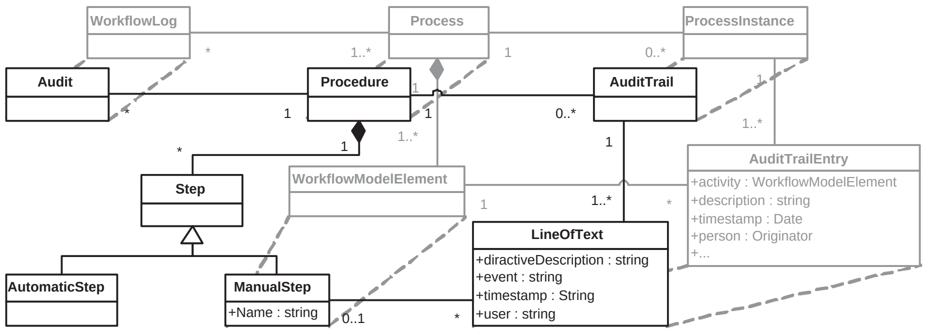

Fig. 6. Mapping meta models.

When this mapping is available, it is almost trivial to map log files to MXML. Each audit in Staffware, contains the logs of only one procedure (or process) of which each audit trail is described separately. These audit trails can easily be mapped onto "process instances". Each line of text in the audit file of Table 2 contains four columns. The first column can be mapped onto the "workflow model element", since they refer to manual steps of which the process is composed. The third and fourth column are obviously mapped onto the "originator" and "timestamp" respectively. In Table 3, we show the mapping for the events that can appear in the second column of Table 2 and what to do with the beginning of a case and the end of a case, i.e. the "Start" and "Terminated" events respectively. The beginning and end of a case are special situations, since they are lines of text *without* an associated manual step. Therefore, we have to create virtual manual steps.

Staffware event	ProM workflowmodelelement	ProM eventtype
Start	"case start"	unknown: case event
Processed To	as in 2nd column	schedule
Released By	as in 2nd column	complete
Expired	as in 2nd column	unknown: expire
Withdrawn	as in 2nd column	withdrawn
Terminated	"case end"	unknown: case event

Table 3. The Staffware mapping.

5.3 Ontological analysis

In the spirit of [4, 6, 11] we can conduct an ontological analysis of the Staffware logging capabilities. First, we check for *Ontological Incompleteness*, also named *Construct Deficit*, which exists "unless there is at least one grammatical construct for each ontological construct" [4, 6, 11]. Staffware is quite complete. The most important construct deficit is the absence of an event type comparable to "start" (e.g., a worker that picks up a work-item). As a result it is impossible to distinguish "waiting" and "service" times. Second, we check for *Ontological Clarity.* This is determined by the extent to which the grammar does not exhibit one or more of the following deficiencies: (1) *Construct Overload* exists in a grammar if one grammatical construct represents more than one ontological construct, (2) *Construct Redundancy* exists if more than one grammatical construct represents the same ontological construct, (3) *Construct Excess* exists in a grammar when a grammatical construct is present that does not map to any ontological construct [4, 6, 11]. The fact that Staffware uses a start (end) event to record the creation (completion) of a case can be seen as construct overload. There is no construct redundancy nor construct excess. So overall, the ontological analysis is quite positive. For Staffware such an analysis may seem trivial. However, for systems that support a completely different way of logging, the ontological analysis is less straightforward.

5.4 Meeting the requirements

In Section 3.1, we gave five requirements for log files in order for them to be useful for process mining. In this section, we discuss whether these requirements are met for Staffware, taking the mapping we established and the ontological analysis into account.

1. Each audit trail entry refers to a specific point in time. In the Staffware log, each line is an audit trail entry and since each line has a timestamp, this requirement is met.
2. Each audit trail entry refers to one activity. In the Staffware log, the activities are described by the first column. However, there are two problems. First of all, the start and termination of a case does not have an activity-name. Second, we cannot conclude that the activity names are unique. In fact it is possible to have multiple activities in the definition of a Staffware process with the same label. To still meet this requirement, we assume the start and end of a case to belong to a fictive manual step. Furthermore, we just *assume* that activities are uniquely identified by their labels.
3. The second column contains a reference to the event that actually happened, so this requirement is met.
4. The audit trail entries are sorted per case, so this requirement is met.
5. The cases all belong to the same process, since a Staffware log file always contains at most one procedure. Therefore, the last requirement is met as well.

In this section we have shown that it is possible to make mappings from log files of commercial systems to MXML. Due to space limitations, we cannot show the complete result of the translation. However, Table 1 shows a part of the converted log file.

6 Known issues

In Section 5, we have shown that it is possible to map Staffware logs onto MXML. It may be clear that our goal is to come up with such mappings for as many information systems as possible. So far, we have discovered that for most production workflow system, making these mappings is almost trivial. However, for systems that are more data driven than process driven, these mappings become extremely difficult. For example, information systems like SAP R/3 and Peoplesoft are capable of logging almost anything on a database level. However, it has proven to be impossible to discover the case (process instance) to which an event belongs from the event log.

Obviously, the internal data structures of complex information systems like SAP and Peoplesoft have to be able to link database transactions to cases. However, usually, the way these are linked is implementation and release specific, which makes it almost impossible to derive generic results. On an implementation-specific level however, we have seen promising results in both SAP and Peoplesoft.

Another issue that needs to be addressed is the situation where not one system is providing the event logs, but logs are taken from multiple legacy systems. This of course requires even more bookkeeping and makes it even harder to restore relations between events, cases, etc. However, data warehousing techniques may ease the burden.

7 Related work

The MXML format presented in this paper is not the only attempt to give a formalization of data models for event logging. Several papers focus on the use of meta models in the context of process-aware information systems [9, 12]. Most of these meta models however focus on the functionality of these systems rather than their ability to record event logs. The ontological analysis of different languages has been the topic of many papers. For example, in [4, 6, 11] the Bunge-Wand-Weber ontology is used to compare different languages. Again such an analysis focuses on the core functionality rather than logging facilities.

We would like to discuss two related approaches in more detail. The first is an attempt by the Workflow Management Coalition (WFMC) to standardize the communication between workflow engines and administration and monitoring tools. The second is the tool Aris PPM (Process Performance Monitor), developed by IDS Scheer.

7.1 Interface 5 of the WFMC reference model

In the area of workflow management, the Workflow Management Coalition has developed a reference model for communication between the core of a workflow system, i.e. the Workflow Engine, and several supporting tools. For this, five interfaces have been developed, of which *Interface 5* is of most interest to us.

It is defined as the interface for communication between the workflow engine and administration and monitoring tools. Unfortunately, a good standard for this interface has never been developed. A meta model for this interface was proposed recently in Section 4, page 175 of [9]. This model, however, shows how information in a log file relates to objects created at runtime and objects created at build time, but it is too high level to be used as a starting point for process mining.

7.2 PPM

A well known tool in the area of process performance monitoring, is *ARIS PPM* (Process Performance Monitor) [7] developed by IDS Scheer. ARIS PPM allows for the visualization, aggregation, and analysis of process instances expressed in terms of *instance EPCs* (i-EPCs). An instance EPC describes the control-flow of a case and it provides a graphical representation describing the causal relations between events within the case. In case of parallelism, there may be different traces having the same instance EPC. Note that in the presence of parallelism, two subsequent events do not have to be causally related. ARIS PPM exploits the advantages of having instance EPCs rather than traces to provide additional management information, i.e., instances can be visualized and aggregated in various ways.

Typically, Aris PPM communicates with systems like Staffware and SAP R/3 using a number of custom-made adapters. These adapters, unfortunately, can only create instance EPCs if the actual process is known. As a result, it is very time consuming to build adapters. Moreover, the approaches used only work in environments where there are explicit process models available.

8 Conclusion

In this paper, we introduced a standard for storing event logs generated by process-aware information systems. For this, we provide requirements, a data model and an XML format called MXML. Furthermore, we have shown an example of a mapping from an event log of a commercial workflow system to MXML. In Section 4, we introduced the process mining framework ProM. This framework accepts event logs in the MXML format and it enables researchers to implement new process mining techniques and benefit from each others ideas, without having to care about the information system the event logs were generated by. Furthermore, by mapping event logs from commercial information systems to MXML, the applicability of process mining in business environments greatly improves. However, to establish mappings from the log formats of different information systems to the MXML format, an in-depth evaluation of a large enough number of these systems is needed.

Acknowledgements and relation to INTEROP

We thank INTEROP for supporting this work that has been conducted in the context of the INTEROP work package "Domain Ontologies for Interoperability". More specifically, the paper touches the issue of how to deal with multiple data-models (and accompanying ontologies) of logs of information systems, which fits into Subtask 2 of the work package. Furthermore, especially the mapping presented deals with issues stated in Subtask 3 of the work package, i.e. semantic mapping of ontologies and data models. Since the work is inspired by a practical problem, it can be used as a starting point for further work in Subtask 4, i.e., the investigation of the effectiveness of the use of ontologies for interoperability. As indicated, an evaluation of a more systems (FLOWer, FileNet, IBM Websphere, etc.) is planned in the context of this subtask.

References

1. 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.
2. 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.
3. 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.
4. I. Davies, P. Green, S. Milton, and M. Rosemann. Analyzing and Comparing Ontologies with Meta-Models. In J. Krogs, T. Halpin, and K. Siau, editors, *Information Modeling Methods and Methodologies*, pages 1–16. Idea Group, 2005.
5. B.F. van Dongen, A.J.M.M. Weijters A.K.A. de Medeiros, H.M.W. Verbeek, and W.M.P. van der Aalst. The PRoM framework: A new era in process mining tool support. In *accepted tool presentation at ATPN 2005*, 2005.
6. P. Green and M. Rosemann. Integrated Process Modeling: An Ontological Evaluation. *Information Systems*, 25(3):73–87, 2000.
7. 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.
8. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
9. M. Zur Muehlen. *Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems*. Logos, Berlin, 2004.
10. 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.
11. M. Rosemann and P. Green. Developing a meta model for the Bunge-Wand-Weber ontological constructs. *Information Systems*, 27(2):75–91, 2002.
12. M. Rosemann and M. Zur Muehlen. Evaluation of Workflow Management Systems - a Meta Model Approach. *Australian Journal of Information Systems*, 6(1):103–116, 1998.