Discovering Coordination Patterns using Process Mining

W.M.P. van der Aalst

Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands. w.m.p.v.d.aalst@tm.tue.nl

Abstract. Recently, process mining has become a vivid research area [5, 6]. The basic idea of process mining is to diagnose business processes by mining event logs for knowledge. Process mining techniques and tools provide the means for discovering process, control, data, organizational, and social structures from event logs. In this paper we provide an overview of process mining techniques/tools and their challenges. Using the coordination pattern between a service client and a service provider, we illustrate the application of process mining techniques to uncover coordination patterns.

Key words: Process mining, coordination, business process management, workflow management, delta analysis, Petri nets.

1 Introduction

Today's information systems have become complex concurrent and distributed artifacts consisting of many interacting software components. These software components may reside in one organization but may also span multiple organizations. It is interesting to note that similar interaction patterns occur at the level of software components, business processes, and organizations. Therefore, there is an increasing interest in coordination languages and models. The interest in Component Based Software Engineering (CBSE) and Web Services Composition (WSC), often referred to as Web Services Orchestration (WSO), illustrates this. An nice example is the recent attention for BPEL4WS [11] and formal methods like Pi-calculus [22].

Most of the work on coordination focuses on languages and models for modeling, specifying, and implementing coordination mechanisms. In this paper, we focus on techniques for *monitoring* coordination mechanisms present in enterprise information systems. Today, many enterprise information systems store relevant events in some structured form. For example, workflow management systems typically register the start and completion of activities [3]. ERP systems like SAP log all transactions, e.g., users filling out forms, changing documents, etc. Business-to-business (B2B) systems log the exchange of messages with other parties. Call center packages but also general-purpose CRM systems log interactions with customers. These examples show that many systems have some kind of *event log* often referred to as "history", "audit trail", "transaction log", etc.

[5, 8, 16, 25]. The event log typically contains information about events referring to an *activity* and a *case*. The case (also named process instance) is the "thing" which is being handled, e.g., a customer order, a job application, an insurance claim, a building permit, etc. The activity (also named task, operation, action, or work-item) is some operation on the case. Typically, events have a *timestamp* indicating the time of occurrence. Moreover, when people are involved, event logs will typically contain information on the person executing or initiating the event, i.e., the *originator*. Based on this information several tools and techniques for process mining have been developed [2, 4, 5, 7–9, 17, 18, 23, 25, 27].

Process mining is useful for at least two reasons. First of all, it could be used as a tool to find out how people and/or procedures really work. Consider for example processes supported by an ERP system like SAP (e.g., a procurement process). Such a system logs all transactions but in many cases does not enforce a specific way of working. In such an environment, process mining could be used to gain insight in the actual process. Another example would be the flow of patients in a hospital. Note that in such an environment all activities are logged but information about the underlying process is typically missing. In this context it is important to stress that management information systems provide information about key performance indicators like resource utilization, flow times, and service levels but *not* about the underlying business processes (e.g., causal relations, ordering of activities, etc.). Second, process mining could be used for *Delta analysis*, i.e., comparing the actual process with some predefined process. Note that in many situations there is a descriptive or prescriptive process model. Such a model specifies how people and organizations are assumed/expected to work. By comparing the descriptive or prescriptive process model with the discovered model, discrepancies between both can be detected and used to improve the process. Consider for example the so-called reference models in the context of SAP. These models describe how the system should be used. Using process mining it is possible to verify whether this is the case. In fact, process mining could also be used to compare different departments/organizations using the same ERP system.

Process mining can be used to monitor coordination in and between enterprise information systems. Some of the coordination is done by humans while other coordination tasks are done by software. As indicated, similar interaction patterns occur are the level of software components, business processes, and organizations. Therefore, process mining can be done at many levels. In this paper we focus on coordination mechanisms at the level of business processes. However, the applicability is not limited to this level.

The remainder of this paper is organized as follows. Section 2 briefly discusses related work. Section 3 introduces the concept of business process mining. Section 4 highlights the main challenges. Section 5 presents an example where mining can be used to discover the coordination pattern between a service client and a service provider. Finally, Section 6 concludes the paper.

2 Related work

The idea of process mining is not new [2, 5, 7–9, 17, 18, 20, 23, 25, 27]. Most process mining techniques aim at the control-flow perspective. However, as indicated in the previous section, process mining is not limited to the control-flow perspective. For example, in [4] we use process mining techniques to construct a social network. For more information on process mining we refer to a special issue of Computers in Industry on process mining [6] and a survey paper [5]. In this paper, unfortunately, it is impossible to do justice to the work done in this area.

We have been using different variants of the α-algorithm. For more information on the basic algorithm, we refer to [2, 7, 20, 27]. In [21] one of the problems raised in [20] is tackled ("short loops") and should be considered as an extension of [7].

This paper is based on earlier work of the author [4–6]. Its goals is to introduce the topic of process mining and discuss its relevance in the context of coordination.

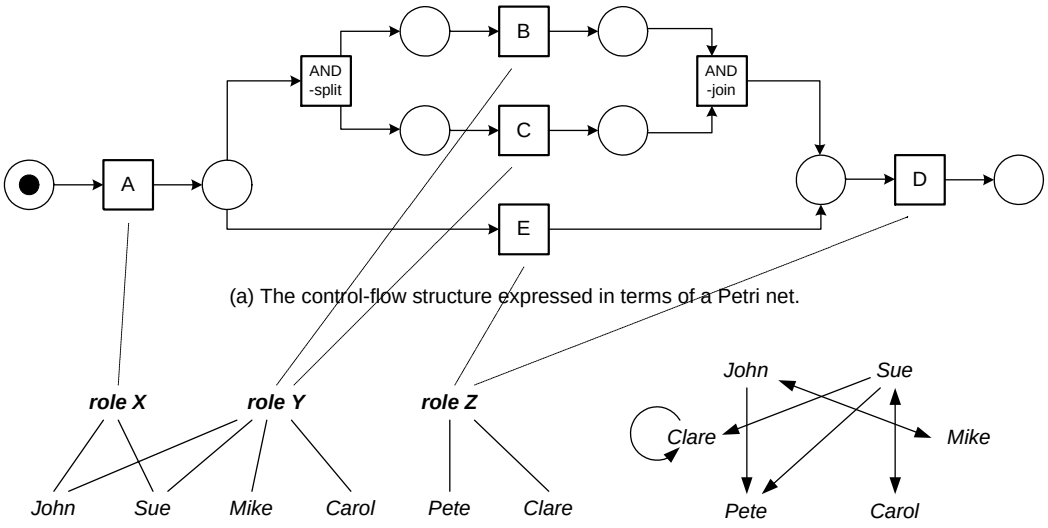
3 Business Process Mining: An overview

The goal of process mining is to extract information about processes from transaction logs [5]. We assume 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 person 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 systems FLOWer logs every modification of some data element.

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 [24] or Event-driven Process Chain (EPC) [19, 18]. 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 [26]). 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.

case id	activity id	originator	timestamp
case 1	activity A	John	9-3-2004:15.01
case 2	activity A	John	9-3-2004:15.12
case 3	activity A	Sue	9-3-2004:16.03
case 3	activity B	Carol	9-3-2004:16.07
case 1	activity B	Mike	9-3-2004:18.25
case 1	activity C	John	10-3-2004:9.23
case 2	activity C	Mike	10-3-2004:10.34
case 4	activity A	Sue	10-3-2004:10.35
case 2	activity B	John	10-3-2004:12.34
case 2	activity D	Pete	10-3-2004:12.50
case 5	activity A	Sue	10-3-2004:13.05
case 4	activity C	Carol	11-3-2004:10.12
case 1	activity D	Pete	11-3-2004:10.14
case 3	activity C	Sue	11-3-2004:10.44
case 3	activity D	Pete	11-3-2004:11.03
case 4	activity B	Sue	11-3-2004:11.18
case 5	activity E	Clare	11-3-2004:12.22
case 5	activity D	Clare	11-3-2004:14.34
case 4	activity D	Pete	11-3-2004:15.56

Table 1. An event log.



(a) The control-flow structure expressed in terms of a Petri net.

(b) The organizational structure expressed in terms of a activity-role-performer diagram.

(c) A sociogram based on transfer of work.

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.

The process perspective is concerned with the "How?" question, the organizational perspective is concerned with the "Who?" question, and the case perspective is concerned with the "What?" question. To illustrate the first two consider Figure 1. The log shown in Table 1 contains information about five cases (i.e., process instances). The log shows that for four cases (1, 2, 3, and 4) the activities A, B, C, and D have been executed. For the fifth case only three activities are executed: activities A, E, and D. Each case starts with the execution of A and ends with the execution of D. If activity B is executed, then also activity C is executed. However, for some cases activity C is executed before activity B. Based on the information shown in Table 1 and by making some assumptions about the completeness of the log (i.e., assuming that the cases are representative and a sufficient large subset of possible behaviors is observed), we can deduce the process model shown in Figure 1(a). The model is represented in terms of a Petri net [24]. The Petri net starts with activity A and finishes with activity D. These activities are represented by transitions. After executing A there is a choice between either executing B and C in parallel or just executing activity E. To execute B and C in parallel two non-observable activities (AND-split and AND-join) have been added. These activities have been added for routing purposes only and are not present in the event log. Note that for this example we assume that two activities are in parallel if they appear in any order. By distinguishing between start events and complete events for activities it is possible to explicitly detect parallelism.

Figure 1(a) does not show any information about the organization, i.e., it does not use any information on the people executing activities. However, Table 1 shows information about the performers. For example, we can deduce that activity A is executed by either John or Sue, activity B is executed by John, Sue, Mike or Carol, C is executed by John, Sue, Mike or Carol, D is executed by Pete or Clare, and E is executed by Clare. We could indicate this information in Figure 1(a). The information could also be used to "guess" or "discover" organizational structures. For example, a guess could be that there are three roles: X, Y, and Z. For the execution of A role X is required and John and Sue have this role. For the execution of B and C role Y is required and John, Sue, Mike and Carol have this role. For the execution of D and E role Z is required and Pete and Clare have this role. For five cases these choices may seem arbitrary but for larger data sets such inferences capture the dominant roles in an organization. The resulting "activity-role-performer diagram" is shown in Figure 1(b). The three "discovered" roles link activities to performers. Figure 1(c) shows another view on the organization based on the transfer of work from one individual to another, i.e., not focus on the relation between the process and individuals but on relations among individuals (or groups of individuals). Consider for example Table 1. Although Carol and Mike can execute the same activities (B and C), Mike is always working with John (cases 1 and 2) and Carol is always working with Sue (cases 3 and 4). Probably Carol and Mike have the same role but based on the small sample shown in Table 1 it seems that John is not working with

Carol and Sue is not working with Carol.[1] These examples show that the event log can be used to derive relations between performers of activities, thus resulting in a sociogram. For example, it is possible to generate a sociogram based on the transfers of work from one individual to another as is shown in Figure 1(c). Each node represents one of the six performers and each arc represents that there has been a transfer of work from one individual to another. The definition of "transfer of work from A to B" is based on whether for the same case an activity executed by A is directly followed by an activity executed by B. For example, both in case 1 and 2 there is a transfer from John to Mike. Figure 1(c) does not show frequencies. However, for analysis proposes these frequencies can be added. The arc from John to Mike would then have weight 2. Typically, we do not use absolute frequencies but weighted frequencies to get relative values between 0 and 1. Figure 1(c) shows that work is transferred to Pete but not vice versa. Mike only interacts with John and Carol only interacts with Sue. Clare is the only person transferring work to herself.

Besides the "How?" and "Who?" question (i.e., the process and organization perspectives), there is the case perspective that is concerned with the "What?" question. Figure 1 does not address this. In fact, focusing on the case perspective is most interesting when also data elements are logged but these are not listed in Table 1. The case perspective looks at the case as a whole and tries to establish relations between the various properties of a case. Note that some of the properties may refer to the activities being executed, the performers working on the case, and the values of various data elements linked to the case. Using clustering algorithms it would for example be possible two show a positive correlation between the the size of an order or its handling time and the involvement of specific people.

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.

To address the three perspectives and the logical and performance issues we have developed a set of tools including EMiT [2], Thumb [27], and MinSoN [4]. These tools share a common XML format. For more details we refer to http://www.processmining.org.

4 Challenging problems

Process mining raises a number of interesting scientific questions. As indicated in the previous section, some of these questions have been answered while others

[1] Clearly the number of events in Table 1 is too small to establish these assumptions accurately. However, for the sake of argument we assume that the things that did not happen will never happen.

require further research. Therefore, we review the most challenging problems. For concrete examples of these problems we use the α algorithm [7, 27] as a starting point.

4.1 Mining hidden tasks

One of the basic assumptions of process mining is that each event (i.e., the occurrence of a task for a specific case) is registered in the log. Clearly, it is not possible to find information about tasks that are not recorded. However, given a specific language it is possible to register that there is a so-called "hidden task".

Consider, for example, Table 1 where A, B, and C are visible but the AND-split in-between A, and B and C is not. Although the log does not reveal the AND-split it is clear that there has to be an AND-split if we assume tasks B and C to be in parallel. Similarly, we can detect that there has to be an AND-join. Suppose that both A and D are removed from Table 1. In this case the α algorithm [7] would not be able to detect the two hidden transitions and the resulting model is shown in Figure 2. Note that Figure 2 can be considered to be equivalent to Figure 1 For this example it is still possible to automatically construct a process model similar to Figure 1. However, for more complicated processes it is more difficult to add these "hidden tasks", and thus posing a interesting problem also related to issues such as observable behavior and (branching) bisimulation [15].

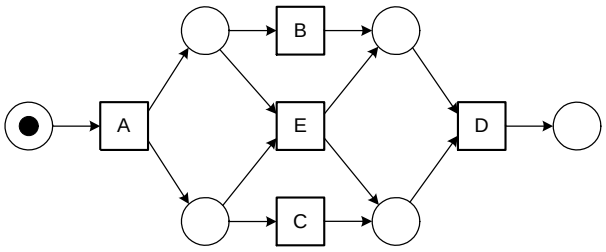


Fig. 2. Applying the α algorithm to Table 1 results in a correct model without the AND-split and AND-join.

4.2 Mining duplicate tasks

The problem of duplicate tasks refers to the situation that one can have a process model (e.g., a Petri net) with two nodes referring to the same task. Suppose that in Table 1 and Figure 1 task E is renamed to B (see Figure 3). Clearly, the modified log could be the result of the modified process model. However, it becomes very difficult to automatically construct a process model from Table 1 with E

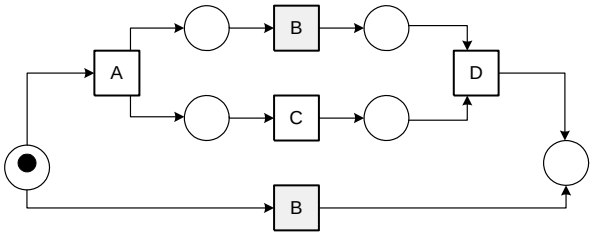


Fig. 3. A process model with duplicate tasks.

renamed to B because it is not possible to distinguish the"B" in case 5 from the "B's" in the other cases. Note that the presence of duplicate tasks is related to hidden tasks. Many processes with hidden tasks but with no duplicate tasks can be modified into equivalent processes with duplicate tasks but with no hidden tasks.

4.3 Mining non-free-choice constructs

Free-choice Petri nets are Petri nets where there are no two transitions consuming from the same input place but where one has an input place which is not an input place of the other [12]. This excludes the possibility to merge choice and synchronization into one construct. Free-choice Petri nets are a well-known and widely used subclass of Petri nets. However, many processes cannot be expressed in terms of a free-choice net. Unfortunately, most of the mining techniques (also those that are not using Petri nets) assume process models corresponding to the class of free-choice nets. Non-free-choice constructs are difficult to model since they represent "controlled choices", i.e., the choice between two tasks is not determined inside some node in the process model but may depend on choices made in other parts of the process model. Clearly, such non-local behavior is difficult to mine and may require many observations.

Figure 1 is free-choice since synchronization (task D) is separated from the choice between A and E. Figure 4 shows a non-free-choice construct. After executing task C there is a choice between task D and task E. However, the choice between D and E is "controlled" by the earlier choice between A and B. Note that tasks D and E are involved in a choice but also synchronize two flows. Clearly such constructs are difficult to mine since the choice is non-local and the mining algorithm has to "remember" earlier events.

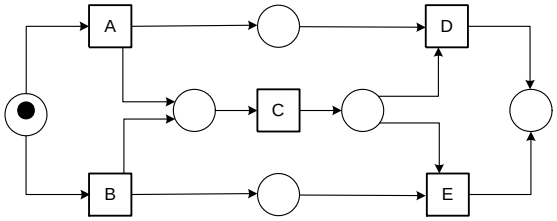


Fig. 4. A non-free-choice process.

4.4 Mining loops

In a process it may be possible to execute the same task multiple times. If this happens, this typically refers to a loop in the corresponding model. Figure 5 shows an example with a loop. After executing task B, task C can be executed arbitrarily many times, i.e., possible event sequences are BD, BCD, BCCD, BCCCD, etc. Loops like the one involving task C are easy to discover. However, loops can also be

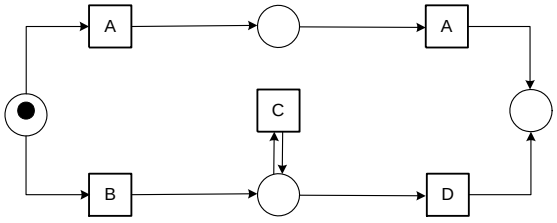


Fig. 5. A process model with a loop.

used to jump back to any place in the process. For more complex processes, mining loops is far from trivial since there are multiple occurrences of the same task in a given case. Some techniques number each occurrence, e.g., B1 C1 C2 C3 D1 denotes BCCCD. These occurrences are then mapped onto a single task. As illustrated by Figure 5 there is a relation between loops and duplicate tasks. In Figure 5 task A is executed multiple times (i.e., twice) but is not in a loop. Many mining techniques make some assumptions about loops which restricts the class of processes that can be mined correctly.

4.5 Using time

Table 1 shows some time information, i.e., each event has a *timestamp*. This log considers activities to be atomic. (Another interpretation would be to think of events as the completion of some activity.) To model the duration of the execution of a task one can log start events and end events. By comparing the difference between the timestamp of a start event and the timestamp of the corresponding end event it is possible the determine the processing time. The timing information can be used for two purposes: (1) adding time information to the process model and (2) improve the quality of the discovered process model.

It is relatively easy to augment a process model with time information. An approach is to first mine the process model while ignoring the timestamps and then "replay" the log in the process model. By replaying the log, it is easy to calculate (average, variance, minimum, and maximum) flow times, waiting times, and processing times [2]. One complication may be that for some cases, the discovered process model may not fit. This information may be used to modify the process model (e.g., modify the resulting model directly, clean the log, or add knowledge and rerun the mining algorithm).

Using timing information to improve the quality of the log is more involved. For example, if two events occur within a short time interval, it is likely that there is some causal relation. A notion of "time distance" could be used in the mining algorithms. However, the added value of this is not clear yet. In fact, as far as we know, no work has been done on this.

4.6 Mining different perspectives

The dominant perspective of process mining is the so-called control-flow perspective. The essence of this perspective is the ordering of tasks. As indicated, the process (i.e., control-flow) perspective can be extended to include timing information (i.e., events have timestamps). However, in addition to the process perspective one could also consider other perspectives. We already discussed the organizational and case perspectives. The organizational structure describes relations between roles (resource classes based on functional aspects) and groups (resource classes based on organizational aspects), and other artifacts clarifying organizational issues (e.g., responsibility, availability). Resources, ranging from humans to devices, form the organizational population and are allocated to roles and groups. Another approach is to build the social network and use

SNA techniques to analyze this. The case perspective typically also includes information/data aspects. Note that the case perspective typically deals with control and production data. Control data are data introduced solely for process management purposes, e.g., variables introduced for routing purposes. Production data are information objects (e.g., documents, forms, and tables) whose existence does not depend on process management. There may be even more perspectives that are interesting for mining, e.g., application perspective deals with the applications being used to execute tasks (e.g., the use of a text editor).

Thus far, most research efforts have focused on the process (control-flow) perspective. Therefore, it is an interesting challenge to include the organization perspective, the case perspective, and/or the application perspective.

4.7 Dealing with noise

Most mining algorithms assume the information to be correct. Although this is a valid assumption in most situations, the log may contain "noise", i.e., incorrectly logged information. For example, it could be that sometimes an event is not recorded or recorded some time after it actually took place. The mining algorithm needs to be robust with respect to noise, i.e., causal relations should not be based on a single observation. In fact, one could argue that the mining algorithm needs to distinguish exceptions from the "normal flow". When considering noise, one often has to determine a threshold value to cut-off exceptional or incorrectly logged behavior. See [27] for some heuristics to deal with noise.

4.8 Dealing with incompleteness

Related to the issue of noise is the notion of incompleteness. A log is incomplete if it does not contain sufficient information to derive the process. Consider Table 1 and the derived process model shown in Figure 1. Suppose that Figure 1 is a correct representation of the actual process but that that the route represented by case 5 is very rare. When mining only a few cases it could be that only cases similar to cases 1, 2, 3, and 4 are recorded. As a result, the discovered process model is not correct because task E is missing. This example may seem trivial, however, for real-life processes there are easily up to a million possible paths when allowing for parallel, conditional and iterative routing. Consider for example Figure 6.

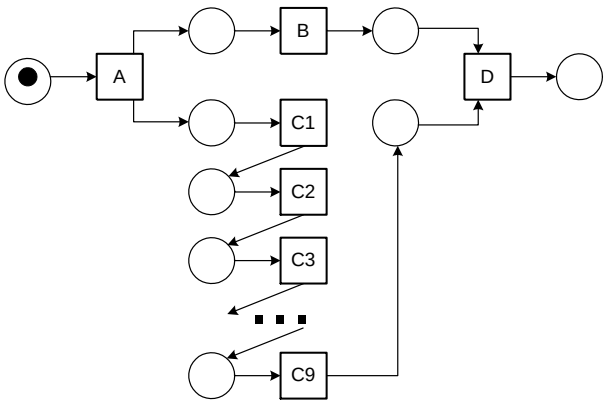


Fig. 6. A process model where it is difficult to pinpoint the synchronization.

Note that in this process there are no choices, i.e., all tasks are executed only once. However, task B and the sequence of 9 tasks C1, C2, ..., C9 are executed in parallel. As a result there are ten possible routes, i.e., even though there are no choices at least 10 cases are needed to derive the process model shown in Figure 6. In fact, observations where B is executed after the sequence of 9 tasks C1, C2, ..., C9 may be highly unlikely and perhaps thousands of logged cases are needed to discover the correct model. If we change the process in Figure 6 such that tasks C1, C2, ..., C9 are executed in parallel, then there are $10! = 3628800$ possible routes. In this case, the log is likely to be incomplete and heuristics are needed to tackle this problem. These heuristics are typically based on Occam's Razor, i.e., the principle that states "When you have two competing theories which make exactly the same predictions, the one that is simpler is the better.".

4.9 Gathering data from heterogeneous sources

Today's enterprise information systems are incredibly complex and typically composed of a large number of applications/components. Applications typically support fragments of a process and as a result the information required for process mining is scattered over the enterprise information system. Therefore, the step to collect the event log used as input for process mining is far from trivial. Even within a single product, events may be logged at several levels of parts of the system. Consider for example an ERP system like SAP: there are dozens of logs relevant for process mining. One approach is to use a data warehouse which extract the information from these logs [14]. In [2] a tool independent XML format is proposed to serve as the standard input format for process mining.

4.10 Visualizing results

Another challenge is to present the results of process mining in such as way that people actually gain insight in the process. Non-trivial management information should visualized in such a way that it is easy to understand. A typical term used in this context is "management cockpit" to emphasize the relevance of presenting the results of process mining. Existing commercial products such as ARIS PPM [18] focus mainly on performance indicators such as flow time, work in progress, etc. Visualizing the complete control-flow perspective or the other perspectives is more difficult and requires further research.

4.11 Delta analysis

Process mining always results in a process model including the control-flow perspective and, perhaps, some of the other perspectives. However, there may already be descriptive or normative models. For example, business consultants may have modeled the process by hand using a simple diagramming tool or even a simulation package. Moreover, the configuration of a WFM system requires an explicit process model and ERP systems are configured on basis of so-called reference models. Given the fact that there may be descriptive or normative models

made by people, it is interesting to compare these models with the models resulting from process mining. Delta analysis is used to compare the two models and explain the differences. Few techniques are know to detect differences and commonalities of process models [1, 10]. Both from a practical point of view and a scientific point of view, Delta analysis is interesting and deserves more attention.

In this section, we identified a number of domains comprising challenging problems that remain unsolved (satisfactorily). By tackling these problems, it is possible to improve the applicability and relevance of process mining.

5 Example: Discovering coordination patterns between a service client and service provider

After providing an introduction to process mining and discussing some of the many challenges, we now focus on a specific coordination problem: establishing an agreement between a service client and a service provider. However, before doing so, let us make some general comments on applying process mining to coordination.

Coordination can be defined as "the harmonious functioning of parts for effective results". Coordination is required at many levels ranging from the hardware in a single computer to the orchestration of interorganizational business processes. There are basically two ways of coordination: (1) coordination between a control system and a controlled system and (2) coordination between two or more autonomous systems. When applying process mining to coordination there are several ways to monitor coordination activities between systems. Each system may monitor its interactions with the environment (e.g., receiving and sending of messages). In case of a binary coordination setting (i.e., just two systems), both systems have a complete view on all interactions. In case of more than two systems, such a complete view may be lacking. In such situations, it may be useful to establish some monitoring agent that sits in-between all systems participating in the coordination.

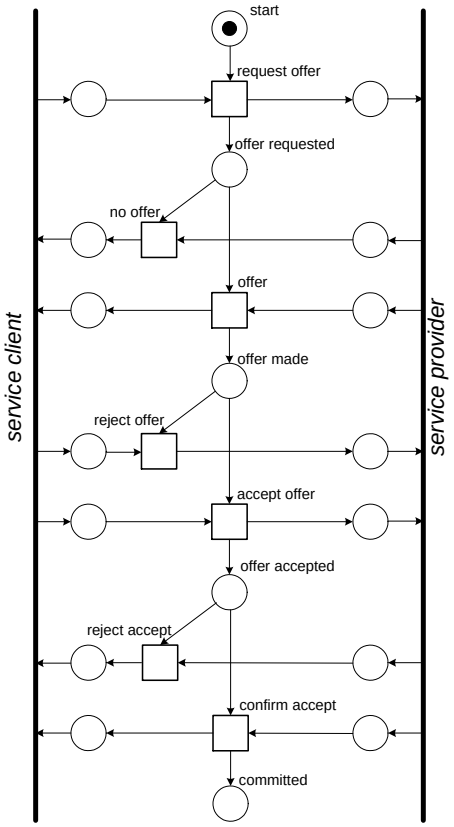


Fig. 7. Coordination pattern: Single non-binding offer [13].

Let us now focus on a concrete example. Figure 7 shows the so-called "single non-binding offer" negotiation pattern. This pattern is taken from [13] where 8 negotiation patterns are described. These patterns correspond to generic building blocks for coordinating a service client and a service provider. In the coordination pattern depicted in Figure 7, the client requests an offer. The provider either sends an offer or not. If no offer is returned, the coordination ends. The service provider can send additional offers until some offer is accepted or rejected. If the offer is rejected, the coordination ends. If it is accepted, the service provider can reject or confirm the acceptance. Only in the latter case, the negotiation ends successfully. The 7 variants of this negotiation pattern discussed in [13] consider "binding offers" (i.e., after acceptance of the client the offer cannot be "withdrawn"), "counter offers", etc.

Note that in Figure 7 each of the two parties involved can monitor all coordination activities (i.e., the transitions corresponding to message exchange are visible for both). About 10 negotiations are needed to discover the Petri net shown in Figure 7, i.e., process mining can be used to discover coordination patterns. As indicated in the introduction, this may be useful for two reasons. First of all, it is a way to find out how coordination really works. By including additional information such as data and time it is also possible to discover relations between performance (e.g., time and success rate) and properties of the negotiation and the two parties involved. Second, process mining can also be used for Delta analysis, i.e., it is possible to compare the actual behavior and the assumed/expected behavior specified in some descriptive or prescriptive model (e.g., the frame contract).

6 Conclusion

This paper presented an overview of process mining and highlighted some of the more technical challenges. The paper provides an overview on earlier work presented in [4–6] and it's goal is to trigger discussion at the ATPN 2004 Workshop on Petri Nets and Coordination (PNC04). For more information on process mining we refer to http://www.processmining.org.

As indicated in the introduction, similar interaction patterns occur at the level of software components, business processes, and organizations. In this paper, we focused on coordination mechanisms at the level of business processes. Clearly, the event logs of many enterprise information systems can be used to monitor and analyze at least part of the coordination processes taking place. It is interesting to think of mining of coordination patterns at different levels. One obvious application is in the area of web services and web services composition/orchestration in particular. For example, it would be interesting to extend languages like BPEL4WS [11] with standard logging facilities.

Acknowledgements
The author would like to thank Ton Weijters, Boudewijn van Dongen, Ana Karla Alves de Medeiros, Minseok Song, Laura Maruster, Eric Verbeek, Monique Jansen-Vullers, Hajo Reijers, Michael Rosemann, and Peter van den Brand for

their on-going work on process mining techniques and tools at Eindhoven University of Technology. Parts of this survey/discussion paper have been based on earlier papers with these researchers.

References

1. W.M.P. van der Aalst and T. Basten. Identifying Commonalities and Differences in Object Life Cycles using Behavioral Inheritance. In J.M. Colom and M. Koutny, editors, *Application and Theory of Petri Nets 2001*, volume 2075 of *Lecture Notes in Computer Science*, pages 32–52. Springer-Verlag, Berlin, 2001.
2. 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.
3. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
4. W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering interaction patterns in business processes. In M. Weske, B. Pernici, and J. Desel, editors, *International Conference on Business Process Management (BPM 2004)*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2004.
5. 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.
6. 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.
7. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. QUT Technical report, FIT-TR-2003-03, Queensland University of Technology, Brisbane, 2003. (Accepted for publication in IEEE Transactions on Knowledge and Data Engineering.).
8. 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.
9. 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.
10. 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.
11. F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.0. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2002.
12. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
13. A. van Dijk. Contracting Workflows and Protocol Patterns. In W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, editors, *International Conference on*

Business Process Management (BPM 2003), volume 2678 of *Lecture Notes in Computer Science*, pages 152–167. Springer-Verlag, Berlin, 2003.
14. J. Eder, G.E. Olivotto, and Wolfgang Gruber. A Data Warehouse for Workflow 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 1–15. Springer-Verlag, Berlin, 2002.
15. R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.
16. 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.
17. 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.
18. IDS Scheer. ARIS Process Performance Manager (ARIS PPM). http://www.ids-scheer.com, 2002.
19. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
20. A.K.A. de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters. Workflow Mining: Current Status and Future Directions. In R. Meersman, Z. Tari, and D.C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 389–406. Springer-Verlag, Berlin, 2003.
21. A.K.A. de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters. Process Mining: Extending the α-algorithm to Mine Short Loops. BETA Working Paper Series, WP 113, Eindhoven University of Technology, Eindhoven, 2004.
22. R. Milner. *Communicating and Mobile Systems: The Pi-Calculus*. Cambridge University Press, Cambridge, UK, 1999.
23. 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.
24. 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.
25. M. Sayal, F. Casati, and M.C. Shan U. Dayal. Business Process Cockpit. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 880–883. Morgan Kaufmann, 2002.
26. J. Scott. *Social Network Analysis*. Sage, Newbury Park CA, 1992.
27. 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.