Process Mining: A Research Agenda

W.M.P. van der Aalst and A.J.M.M. Weijters

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

Abstract. Enterprise information systems support and control operational business processes ranging from simple internal back-office processes to complex interorganizational processes. Technologies such as Workflow Management (WFM), Enterprise Application Integration (EAI), Enterprise Resource Planning (ERP), and Web Services (WS) typically focus on the realization of IT support rather than monitoring the operational business processes. Process mining aims at extracting information from event logs to capture the business process as it is being executed. In this paper, we try to put the topic of process mining into context, discuss the main issues around process mining, and finally we introduce the papers in this special issue.

Key words: Process mining, Workflow mining, Workflow management, Data mining, Petri nets.

1 Introduction

During the last decade explicit process concepts (e.g., workflow models) [2, 4, 17, 30, 32, 33, 36, 57] have been applied in many enterprise information systems. These concepts are also playing a major role in cross-organizational processes, cf. the work on web services composition languages such as BPEL4WS and BPML [13, 9]. Workflow Management (WFM) systems such as Staffware, IBM MQSeries, COSA, etc. offer generic modeling and enactment capabilities for structured business processes. By making graphical process definitions, i.e., models describing the life-cycle of a typical case (process instance) in isolation, one can configure these systems to support business processes. Besides pure WFM systems many other software systems use explicit process models. Consider for example ERP (Enterprise Resource Planning) systems such as SAP, PeopleSoft, Baan and Oracle, CRM (Customer Relationship Management) software, etc. Although enterprise information systems are increasingly "process aware" and there are generic languages and tools (e.g., WFM languages and systems), little attention is devoted to process monitoring and improvement. Note that typically WFM systems do not provide functionality to diagnose the running workflow.

The strong focus on process automation and little attention to issues like flexibility and diagnosis, resulted in many failures. For example, many workflow projects failed. As a result, e.g., workflow vendors are broadening their scope.

Currently, many workflow vendors are positioning their systems as BPM (Business Process Management) systems. Gartner expects the BPM market to grow and also identifies *Business Process Analysis* (BPA) as an important aspect [18]. It is expected that the BPA market will continue to grow. Note that BPA covers aspects neglected by traditional workflow products (e.g., diagnosis, simulation, etc.). *Business Activity Monitoring* (BAM) is one of the emerging areas in BPA. The goal of BAM tools is to use data logged by the information system to diagnose the operational processes. An example is the ARIS Process Performance Manager (PPM) of IDS Scheer [29]. ARIS PPM extracts information from audit trails (i.e., information logged during the execution of cases) and displays this information in a graphical way (e.g., flow times, bottlenecks, utilization, etc.). The trend to focus more on BPA and BAM is not limited to WFM systems. For example, ERP systems are offering so-called Business Intelligence (BI) tools. For example, SAP states that "Business Intelligence integrates all your corporate information so you can turn information into insight, insight into action, and action into improved business operations.".

Buzzwords like BPA, BAM, and BI illustrate the desire to offer tools to monitor operational business processes. Process mining can be seen as a technology to contribute to this. The goal of process mining is to extract an explicit process model from event logs, i.e., the challenge to create a process model given a log with events such that the model is consistent with the observed dynamic behavior. Note that this is not limited to performance data, i.e., a process is more than its average flow time. Process mining also focuses on causal relations between activities.

Process mining generates a number of scientific and practical challenges (e.g., which processes can be discovered and how much data is needed to provide useful information). In this paper, we discuss some of these challenges. First, we define process mining using an example. Then, we review existing literature in this domain. Section 4 highlights the main challenges. Section 5 discusses the spectrum of solution approaches. Finally, Section 6 briefly discusses the papers in this special issue, and Section 7 concludes the paper.

2 Process mining

Instead of starting with a process design, process mining starts by gathering information about the processes as they take place. We assume that it is possible to record events such that we have information about the order in which the events of a case are executed. Any information system using transactional systems such as ERP, CRM, B2B, SCM and WFM systems will offer this information in some form. Note that we do not assume the presence of a WFM system. The only assumption we make, is that it is possible to collect a process log with data about the order the events take place. This process log is used to construct a process specification, which adequately models the behavior registered. We use the term *process mining* for the method of distilling a structured process description from a set of real executions. In recent years we have developed techniques for process

mining and have evaluated these techniques in different domains.

case identifier	task identifier
case 1	task A
case 2	task A
case 3	task A
case 3	task B
case 1	task B
case 1	task C
case 2	task C
case 4	task A
case 2	task B
case 2	task D
case 5	task E
case 4	task C
case 1	task D
case 3	task C
case 3	task D
case 4	task B
case 5	task F
case 4	task D

Table 1. A process log.

To illustrate the principle of process mining, we consider the process log shown in Table 1. This log contains information about five cases (i.e., process instances). The log shows that for four cases (1,2,3, and 4) the tasks A, B, C, and D have been executed. For the fifth case only two different tasks are executed: tasks E and F. If task B is executed, then also task C is executed. However, for some cases task C is executed before task 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 for example the process model shown in Figure 1. The model is represented in terms of a Petri net [46]. The Petri net can start with task A and finish with task D. These tasks are represented by transitions. After executing A tasks B and C are in parallel. Note that for this example we assume that two tasks are in parallel if they appear in any other. By distinguishing between start events and end events for tasks it is possible to explicitly detect parallelism. Instead of starting with A the process can also start with E. Task E is always followed by task F.

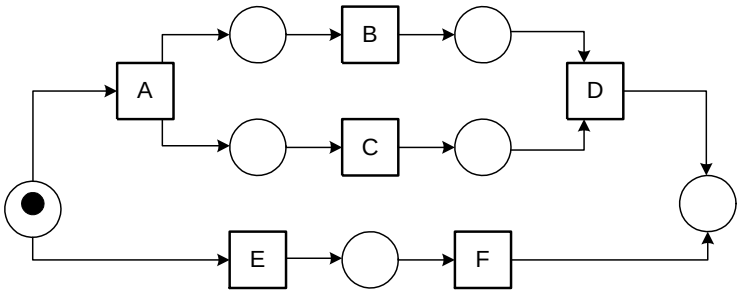

Fig. 1. A process model corresponding to the process log.

Table 1 contains the minimal information we assume to be present. In many applications, the process log contains a timestamp for each event and this information can be used to extract additional causality information. Moreover, we

also investigate the relation between attributes of the case and the actual route taken by a particular case. For example, when handing traffic violations: Is the make of a car relevant for the routing of the corresponding traffic violations? (E.g., People driving a Ferrari always pay their fines in time.)

For this simple example, it is quite simple to construct a process model that is able to regenerate the process log. For larger process models this is much more difficult. For example, if the model exhibits alternative and parallel routing, then the process log will typically not contain all possible combinations. Consider 10 tasks which can be executed in parallel. The total number of interleavings is $10! = 3628800$. It is not realistic that each interleaving is present in the log. Moreover, certain paths through the process model may have a low probability and therefore remain undetected. Noisy data (i.e., errors in the log) can further complicate matters. These are just some of the problems we that we need to face in this project.

3 Literature on process mining

The idea of process mining is not new [7, 10–12, 23–28, 37–39, 48–52, 54–56]. Cook and Wolf have investigated similar issues in the context of software engineering processes. In [10] they describe three methods for process discovery: one using neural networks, one using a purely algorithmic approach, and one Markovian approach. The authors consider the latter two the most promising approaches. The purely algorithmic approach builds a finite state machine where states are fused if their futures (in terms of possible behavior in the next k steps) are identical. The Markovian approach uses a mixture of algorithmic and statistical methods and is able to deal with noise. Note that the results presented in [10] are limited to sequential behavior. Related, but in a different domain, is the work presented in [34, 35] also using a Markovian approach restricted to sequential processes. Cook and Wolf extend their work to concurrent processes in [11]. They propose specific metrics (entropy, event type counts, periodicity, and causality) and use these metrics to discover models out of event streams. However, they do not provide an approach to generate explicit process models. In [12] Cook and Wolf provide a measure to quantify discrepancies between a process model and the actual behavior as registered using event-based data. The idea of applying process mining in the context of workflow management was first introduced in [7]. This work is based on workflow graphs, which are inspired by workflow products such as IBM MQSeries workflow (formerly known as Flowmark) and InConcert. In this paper, two problems are defined. The first problem is to find a workflow graph generating events appearing in a given workflow log. The second problem is to find the definitions of edge conditions. A concrete algorithm is given for tackling the first problem. The approach is quite different from other approaches: Because the nature of workflow graphs there is no need to identify the nature (AND or OR) of joins and splits. As shown in [31], workflow graphs use true and false tokens which do not allow for cyclic graphs.

Nevertheless, [7] partially deals with iteration by enumerating all occurrences of a given task and then folding the graph. However, the resulting conformal graph is not a complete model. In [39], a tool based on these algorithms is presented. Schimm [48, 49, 52] has developed a mining tool suitable for discovering hierarchically structured workflow processes. This requires all splits and joins to be balanced. Herbst and Karagiannis also address the issue of process mining in the context of workflow management [25, 23, 24, 27, 28, 26] using an inductive approach. The work presented in [26, 28] is limited to sequential models. The approach described in [25, 23, 24, 27] also allows for concurrency. It uses stochastic task graphs as an intermediate representation and it generates a workflow model described in the ADONIS modeling language. In the induction step task nodes are merged and split in order to discover the underlying process. A notable difference with other approaches is that the same task can appear multiple times in the workflow model, i.e., the approach allows for duplicate tasks. The graph generation technique is similar to the approach of [7, 39]. The nature of splits and joins (i.e., AND or OR) is discovered in the transformation step, where the stochastic task graph is transformed into an ADONIS workflow model with block-structured splits and joins. In contrast to the previous papers, the following papers are characterized by the focus on workflow processes with concurrent behavior (rather than adding ad-hoc mechanisms to capture parallelism). In [54–56] a heuristic approach using rather simple metrics is used to construct so-called "dependency/frequency tables" and "dependency/frequency graphs". In [37] another variant of this technique is presented using examples from the health-care domain. The preliminary results presented in [37, 54–56] only provide heuristics and focus on issues such as noise. The approach described in [6] differs from these approaches in the sense that for the α algorithm it is proven that for certain subclasses it is possible to find the right workflow model. In [3] the EMiT tool is presented which uses an extended version of α algorithm to incorporate timing information.

Process mining can be seen as a tool in the context of Business (Process) Intelligence (BPI). In [22, 47] 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 order to do workflow mining it is convenient to have a so-called "process data warehouse" to store audit trails. Such as data warehouse simplifies and speeds up the queries needed to derive causal relations. In [15, 41–43] the design of such warehouse and related issues are discussed in the context of workflow logs. Moreover, [43] describes the PISA tool which can be used to extract performance metrics from workflow logs. Similar diagnostics are provided by the ARIS Process Performance Manager (PPM) [29]. The later tool is commercially available and a customized version of PPM is the Staffware Process Monitor (SPM) [53] which is tailored towards mining Staffware logs. Note that none of the latter tools is extracting the process model. The main

focus is on clustering and performance analysis rather than causal relations as in [7, 10–12, 23–28, 37–39, 48–52, 54–56].

More from a theoretical point of view, the rediscovery problem discussed in this paper is related to the work discussed in [8, 20, 21, 45]. In these papers the limits of inductive inference are explored. For example, in [21] it is shown that the computational problem of finding a minimum finite-state acceptor compatible with given data is NP-hard. Several of the more generic concepts discussed in these papers could be translated to the domain of process mining. It is possible to interpret the problem described in this paper as an inductive inference problem specified in terms of rules, a hypothesis space, examples, and criteria for successful inference. The comparison with literature in this domain raises interesting questions for process mining, e.g., how to deal with negative examples (i.e., suppose that besides log W there is a log V of traces that are not possible, e.g., added by a domain expert). However, despite the many relations with the work described in [8, 20, 21, 45] there are also many differences, e.g., we are mining at the net level rather than sequential or lower level representations (e.g., Markov chains, finite state machines, or regular expressions).

There is a long tradition of theoretical work dealing with the problem of inferring grammars out of examples: given a number of sentences (traces) out of a language, find the simplest model that can generate these sentences. There is a strong analogy with the process-mining problem: given a number of process traces, can we find the simplest process model that can generate these traces. Many issues important in the language-learning domain are also relevant for process mining (i.e. learning from only positive examples, how to deal with noise, measuring the quality of a model, etc.). However, an important difference between the grammar inference domain and the process-mining domain is the problem of concurrency in the traces: concurrency seems not relevant in the grammar inference domain. In spite of this important difference, it seems usefully to investigate which theoretical results, measurements, and mining techniques can be used or updated so that they become useful in process mining. A good overview of prominent computational approaches for learning different classes of formal languages is given in [44] and a special issue of the machine learning journal about this subject [**?**].

Additional related work is the seminal work on regions [16]. This work investigates which transition systems can be represented by (compact) Petri nets (i.e., the so-called synthesis problem). Although the setting is different and our notion of completeness is much weaker than knowing the transition system, there are related problems such as duplicate transitions, etc.

For more information on existing research, we also refer to [5] for a survey.

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 require further research. Therefore, we review the most challenging problems.

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, that in Table 1 the events referring to task A are removed. Although the log does not reveal task A 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 if we remove all events referring to task D from the log. Suppose that both A and D are removed from Table 1. In this case it is still possible to automatically construct a process model similar Figure 1 (see Figure 2). 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 [19].

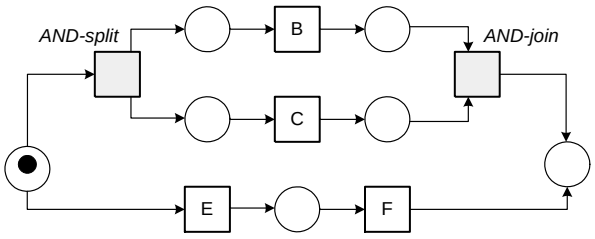

Fig. 2. A process model with two hidden tasks.

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 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

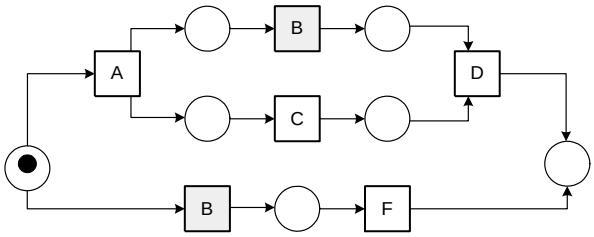

Fig. 3. A process model with duplicate tasks.

input place of the other [14]. 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.

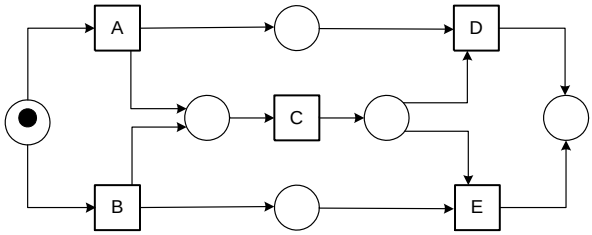

Fig. 4. A process model with a non-free-choice construct.

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.

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 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.

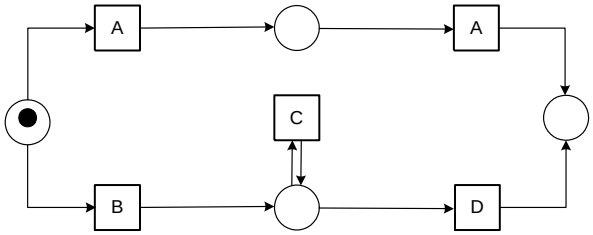

Fig. 5. A process model with a loop.

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 the minimal information needed to conduct some form of process mining. Each line corresponds to an event (i.e., the execution of a task for a specific case). In many cases, the log also contains time information, i.e., each event has a *timestamp*. 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 [3]. 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 control-flow perspective can be extended to include timing information (i.e., events have timestamps). However, in addition to the control-flow perspective one could also consider: the organization perspective, the information perspective, and the application perspective. In the organization perspective, the organizational structure and the population are specified. 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. The information perspective 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. The application perspective deals with the applications being used to execute tasks (e.g., the use of a text editor).

In an event log one can find traces of these other perspectives. An event in the log may give information about the resource (e.g., worker) that executed the corresponding task. This information can be used to derive knowledge about the organization perspective (e.g., roles and groups, collaboration structures, efficiency of cooperation, etc.). For example, process mining could be used to find that cases that require the cooperation of two specific workers have significantly longer processing times. The log may also record the "dataflow", e.g., which process variables are updated in a given event and what are their values. Such logs could be used to derive knowledge about the information perspective. It is particularly interesting to link the information perspective to the control-flow perspective. For example, is there a correlation between the flow time and certain process variables (e.g., large orders take more time) or is there a relation between the routing of a case and its process variables (e.g., cases of customers located in a specific region typically require several additional checks)? Similarly, the log could be augmented with information about the applications being used to derive knowledge about the application perspective. Thus far, most research efforts have focused on the control-flow perspective. Therefore, it is an interesting challenge to include the organization perspective, the information 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 [37, 54–56] 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 tasks E and F are 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. 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

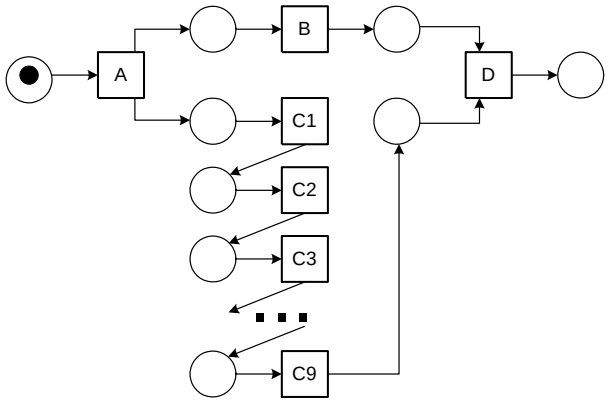

Fig. 6. A process model where it is difficult to pinpoint the synchronization.

of logs relevant for process mining. One approach is to use a data warehouse which extract the information from these logs [15]. In [3] a tool independent XML format is proposed to serve as input format for several tools 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 [29] 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, 12]. 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 Differences in mining algorithms

In the previous section (Section 4), we reviewed a number of (partly) solved and unsolved problems related to process mining. In this section we focus on differences in mining *algorithms*. There is of course a strong relation between the mining algorithm and the type of problems that can be successfully handled by that algorithm. Therefore, if we try to characterize a mining algorithm, we can start with a enumeration of the types of problems that can be successful handled by the algorithm at hand (e.g. dealing with noise, incomplete logs, using time, duplicate tasks, non-free-choice constructs, loops, mining different perspectives, visualizing the mining results, etc.).

So far it appears almost impossible to use existing data mining techniques directly (i.e. without some modifications) for process mining (for a an excellent overview of the key mining algorithms we refer to [40]). That means that most of the process mining techniques have some very specific properties. In spite of this, process mining can be seen as a sub-domain of data mining in general. Many of the characteristics relevant for data mining algorithms appear also relevant for process mining (e.g. the inductive bias, the local-global dimension, computational complexity, memory requirement, etc.). Below we will discuss them in the context of process mining.

Workflow logs can also contain information about the attributes of cases and the actual route taken by a particular case. Given a process model, traditional data mining techniques can be used for the mining of decision rules that predict the routing of a specific case. For this reason the focus during process mining is on mining the process model, not on the induction of the rules for predicting the routing of a individual case.

Most workflow logs only contain positive examples and, in case of noise, perhaps even negative examples without a mark that they are negative. Only if domain specialists are involved, it is sometimes possible to generate *negative examples* (this event-pattern can not appear). It seems that if a process mining algorithm needs negative examples, the practical application of that algorithm becomes questionable.

5.1 The inductive bias during process mining algorithm

In data mining, but also in process mining, the mining process can be seen as searching through a large space of possible models implicitly defined by the language we use to represent discovered process models. The goal of this search is to find the process model that best fits with the data in the workflow log. It is very important to realize that the choice of the process representation language

strongly influences the mining process. Examples of process model representation languages are Petri nets, block-oriented process models, and event dependency models. Some modeling languages are stronger than other modeling languages. For instance each block-oriented process model can easily be translated in a Petri-net, the other way around is not always possible. For that reason a Petri-net is a stronger representation language than block-oriented process models. If we know nothing about the process we try to mine, it seems attractive to use the strongest process model representation language available. After all, the choice for a to weak language will make it impossible to find an appropriate model. But the choice for the most general representation language has the negative effect that the size of the search space grows. An enlarged search space (i) makes the mining technique more sensitive for noise, (ii) needs more data for successful mining, and (iii) has a negative effect on the *computational complexity* and *memory requirement*. The situation is more or less comparable with the situations were we are searching for the right regression model: if we know that we are looking for a linear model and we are using linear regression as our modeling technique (i) a few data examples are appropriate, (ii) the approach is less sensitive for noise, and (iii) the computing time is shorter then for the non linear case. If we know in advance witch type of process model we are looking for and we use this information during the selection of our process model representation language we have a strong *inductive bias*. In practice, many process mining algorithms have a strong inductive bias.

5.2 The local-global dimension

Given the process representation languages of a process mining technique we can look to the mining process as a search for the most appropriate process out of the search space of candidate process models. Mining algorithms can use different strategies to find the most appropriate model. Two extreme strategies can be distinguished (i) *local strategies* primarily based on a step by step building of the optimal process model based on very local information, and (ii) *global strategies* primarily based on a one strike search for the optimal model.

Example of a very local strategies are the α-algorithm as presented in [6] and mining techniques based an Markovian approach. Only very local information about binary relations between events is used.

An example of a very global search strategy is a genetic search for the optimal process model. A genetic search starts with a population of complete process models. Because the quality or fitness of a candidate model is calculated by comparing the process model with all traces in the workflow log the search process is very global.

Both approaches have their advantages and disadvantages. General speaking, local strategies are less complex from a *computational* point of view and the *memory requirement* is lower than for global strategies. However, for local strategy there is no guarantee that the outcome of the locally optimal steps (at the level of binary event relations) will result in a globally optimal process model. Hence, the performance of such local mining techniques can be hampered

seriously when the necessary information is not local available. For instance the α-algorithm mentioned above can not handle the non-free choice constructs as mentioned in Section 4.3 because the choice between tasks is not determined inside some node in the process model but may depend on choices made in other parts of the process model. For a more global technique there is the chance that non-free-choice construct will be discovered.

In practical situations logs are rarely complete and/or noise free. Then it becomes important *how sensitive an algorithm is for noise*: can one erroneous example completely mess up the derivation of a right model or is the algorithm robust for noise. Mostly, global strategies are more robust for noise.

In some approaches local and global strategies are combined. First a local search approach is used. Afterwards a global check is performed on the whole model and all data in the workflow log. In case of some deficiencies the model is automatically updated or suggestions are given on how to repair the model.

In this section, we discussed important issues that can be used for the characterization of different process mining techniques. First, we can characterize algorithms on the basis of the type of problems that can be successfully handled by it. Other dimensions are the inductive bias, the global-local dimensions and strongly related dimensions as the sensitivity for noise, the amount of data needed, computational complexity, and memory requirement.

6 In this special issue

In this section we briefly introduce the six papers selected for this special issue on process mining. The first three papers describe mining systems that result more or less in complete process models. The fourth paper focuses on the problem of the detection of concurrent behavior in processes. The result of the mining of the systems introduced in the last two papers is not a complete process model, but information about some global properties of the process at hand.

6.1 Workflow Mining with InWoLvE

In the first paper of this special issue on process mining, Joachim Herbst and Dimitris Karagiannis give an overview of the algorithms that were implemented within the InWoLvE workflow mining system. InWoLvE solves the workflow mining problem in two steps. In the first step it creates a stochastic activity graph from the example set and in the second step it transforms this stochastic graph into a well-defined workflow model. The presented experiments shows that InWoLvE is applicable for a wide range of workflow models (i.e. that the inductive bias is low).

6.2 Mining Exact Models of Concurrent Workflows

The paper of Guido Schimm presents an approach to mine exact workflow models from workflow logs. The process model representation language is block-oriented.

A process model consists of a arbitrary number of nested building blocks (i.e. *Sequence*, *Parallel*, *Alternative*, and *Loop*). The advantage of the use of the block-oriented representation language is the relation with process algebra (i.e., well defined semantics, modularity, and extensibility) and the property that resulting workflow models are always exact (e.g. complete, specific and minimal). The disadvantage seems the inductive bias of the mining technique.

6.3 Discovering Workflow Models From Activities' Lifespans

The paper of Shlomit Pinter and Mati Golani is more or less a extension of the work of Agrawal, Gunopulos, and Leymann [7] with time information. Two new algorithms for synthesizing (mining) process models out of workflow logs (audit logs) are presented. The model graph generated by each of the algorithms captures all the executions and dependencies that are present in the log, and preserves existing parallelism. The algorithms presented in this paper are compared with the algorithm in [7] by running them on simulated data. The authors claim that the new algorithm outperforms the original one in the sense that the number of excess and absent edges in the resulting graphs is consistently smaller.

6.4 Discovering Models of Behavior for Concurrent Workflows

The focus of the paper of Jonathan Cook, Zhidian Du, Chongbing Liu, and Alexander Wolf is on concurrent behavior of processes. The paper presents techniques to discover patterns of concurrent behavior from traces of workflow events. The techniques are based on a probabilistic analysis of the event traces. Using metrics for the number, frequency, and regularity of event occurrences, a determination is made of the likely concurrent behavior manifested by the system.

The focus of the last two papers in this special issue is not on the discovery of the complete underlying process model out of a workflow log. The goal is to discover more global properties of the process. For this reason, the inductive bias for both techniques is low.

6.5 Business Process Intelligence

The paper of Fabio Casati, Malu Castellanos, Umeshwar Dayal, Mehmet Sayal, and Ming-Chien Shan presents a set of integrated tools (BPI) that supports business and IT users in managing process execution quality by providing several features, such as analysis, prediction, monitoring, control, and optimization. The tool is based on the use of more general data mining techniques to business processes. Experimental results of the use of the BPI-tool are presented in the paper.

6.6 Discovery of Temporal Patterns from Process Instances

The paper of San-Yih Hwang, Chih-Ping Wei, and Wan-Shiou Yang focuses on the discovering frequently occurring temporal patterns and does not assume the

existence of a single process model to which all process instances comply. Discovery of temporal patterns seems sensible in domains with very weak structured process models. Health care is an example of a domain in which many of the process models are weak structured and for this reason very difficult to mine for techniques with a strong inductive bias. In this paper the temporal pattern discovery problem is formally defined and three different temporal pattern discovery algorithms are evaluated, namely TP-Graph, TP-Itemset and TP-Sequence.

7 Conclusion

This paper introduced the topic of process mining. Using a number of simple examples, we illustrated the potential of process mining but also the many scientific challenges that need to be addressed. Problems like hidden tasks, duplicate tasks, non-free-choice constructs, loops, time, noise and lack of completeness limit the practical application of process mining. This special issue provides insight into the state-of-the-art on process mining at this point in time. We hope that this will trigger new research efforts to solve some of the open problems.

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, J. Desel, and A. Oberweis, editors. *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2000.
3. 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.
4. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
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*, pages ??–??, 2003.
6. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Which Processes can be Rediscovered? BETA Working Paper Series, WP 74, Eindhoven University of Technology, Eindhoven, 2002.
7. 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.
8. D. Angluin and C.H. Smith. Inductive Inference: Theory and Methods. *Computing Surveys*, 15(3):237–269, 1983.
9. A. Arkin et al. Business Process Modeling Language (BPML), Version 1.0, 2002.

10. 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.
11. J.E. Cook and A.L. Wolf. Event-Based Detection of Concurrency. In *Proceedings of the Sixth International Symposium on the Foundations of Software Engineering (FSE-6)*, pages 35–45, 1998.
12. 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.
13. 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 propsal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2002.
14. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
15. 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.
16. A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures - Part 1 and Part 2. *Acta Informatica*, 27(4):315–368, 1989.
17. L. Fischer, editor. *Workflow Handbook 2001, Workflow Management Coalition*. Future Strategies, Lighthouse Point, Florida, 2001.
18. Gartner. Gartner's Application Development and Maintenance Research Note M-16-8153, The BPA Market Cathes another Major Updraft. http://www.gartner.com, 2002.
19. R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.
20. E.M. Gold. Language Identfication in the Limit. *Information and Control*, 10(5):447–474, 1967.
21. E.M. Gold. Complexity of Automaton Identification from Given Data. *Information and Control*, 37(3):302–320, 1978.
22. 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.
23. 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.
24. J. Herbst. Dealing with Concurrency in Workflow Induction. In U. Baake, R. Zobel, and M. Al-Akaidi, editors, *European Concurrent Engineering Conference*. SCS Europe, 2000.
25. J. Herbst. *Ein induktiver Ansatz zur Akquisition und Adaption von Workflow-Modellen*. PhD thesis, Universität Ulm, November 2001.
26. J. Herbst and D. Karagiannis. Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. In *Proceedings of the Ninth International Workshop on Database and Expert Systems Applications*, pages 745–752. IEEE, 1998.

27. J. Herbst and D. Karagiannis. An Inductive Approach to the Acquisition and Adaptation of Workflow Models. In M. Ibrahim and B. Drabble, editors, *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, pages 52–57, Stockholm, Sweden, August 1999.
28. J. Herbst and D. Karagiannis. Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 9:67–92, 2000.
29. IDS Scheer. ARIS Process Performance Manager (ARIS PPM). http://www.ids-scheer.com, 2002.
30. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
31. B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows (submitted)*. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2002. Available via http://www.tm.tue.nl/it/research/patterns.
32. P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.
33. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
34. H. Mannila and D. Rusakov. Decomposing Event Sequences into Independent Components. In V. Kumar and R. Grossman, editors, *Proceedings of the First SIAM Conference on Data Mining*, pages 1–17. SIAM, 2001.
35. H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
36. D.C. Marinescu. *Internet-Based Workflow Management: Towads a Semantic Web*, volume 40 of *Wiley Series on Parallel and Distributed Computing*. Wiley-Interscience, New York, 2002.
37. L. Maruster, W.M.P. van der Aalst, A.J.M.M. Weijters, A. van den Bosch, and W. Daelemans. Automated Discovery of Workflow Models from Hospital Data. In B. Kröse, M. de Rijke, G. Schreiber, and M. van Someren, editors, *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pages 183–190, 2001.
38. L. Maruster, A.J.M.M. Weijters, W.M.P. van der Aalst, and A. van den Bosch. Process Mining: Discovering Direct Successors in Process Logs. In *Proceedings of the 5th International Conference on Discovery Science (Discovery Science 2002)*, volume 2534 of *Lecture Notes in Artificial Intelligence*, pages 364–373. Springer-Verlag, Berlin, 2002.
39. M.K. Maxeiner, K. Küspert, and F. Leymann. Data Mining von Workflow-Protokollen zur teilautomatisierten Konstruktion von Prozemodellen. In *Proceedings of Datenbanksysteme in Büro, Technik und Wissenschaft*, pages 75–84. Informatik Aktuell Springer, Berlin, Germany, 2001.
40. T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
41. M. zur Mühlen. Process-driven Management Information Systems Combining Data Warehouses and Workflow Technology. In B. Gavish, editor, *Proceedings of the International Conference on Electronic Commerce Research (ICECR-4)*, pages 550–566. IEEE Computer Society Press, Los Alamitos, California, 2001.

42. M. zur Mühlen. Workflow-based Process Controlling-Or: What You Can Measure You Can Control. In L. Fischer, editor, *Workflow Handbook 2001, Workflow Management Coalition*, pages 61–77. Future Strategies, Lighthouse Point, Florida, 2001.
43. 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.
44. R. Parekh and V. Honavar. Automata Induction, Grammar Inference, and Language Acquisition. In Dale, Moisl, and Somers, editors, *Handbook of Natural Language Processing*. New York: Marcel Dekker, 2000.
45. L. Pitt. Inductive Inference, DFAs, and Computational Complexity. In K.P. Jantke, editor, *Proceedings of International Workshop on Analogical and Inductive Inference (AII)*, volume 397 of *Lecture Notes in Computer Science*, pages 18–44. Springer-Verlag, Berlin, 1889.
46. 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.
47. 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.
48. G. Schimm. Process Mining. http://www.processmining.de/.
49. G. Schimm. Generic Linear Business Process Modeling. In S.W. Liddle, H.C. Mayr, and B. Thalheim, editors, *Proceedings of the ER 2000 Workshop on Conceptual Approaches for E-Business and The World Wide Web and Conceptual Modeling*, volume 1921 of *Lecture Notes in Computer Science*, pages 31–39. Springer-Verlag, Berlin, 2000.
50. G. Schimm. Process Mining elektronischer Geschäftsprozesse. In *Proceedings Elektronische Geschäftsprozesse*, 2001.
51. G. Schimm. Process Mining linearer Prozessmodelle - Ein Ansatz zur automatisierten Akquisition von Prozesswissen. In *Proceedings 1. Konferenz Professionelles Wissensmanagement*, 2001.
52. G. Schimm. Process Miner - A Tool for Mining Process Schemes from Event-based Data. In S. Flesca and G. Ianni, editors, *Proceedings of the 8th European Conference on Artificial Intelligence (JELIA)*, volume 2424 of *Lecture Notes in Computer Science*, pages 525–528. Springer-Verlag, Berlin, 2002.
53. Staffware. Staffware Process Monitor (SPM). http://www.staffware.com, 2002.
54. A.J.M.M. Weijters and W.M.P. van der Aalst. Process Mining: Discovering Workflow Models from Event-Based Data. In B. Kröse, M. de Rijke, G. Schreiber, and M. van Someren, editors, *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pages 283–290, 2001.
55. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data. In V. Hoste and G. de Pauw, editors, *Proceedings of the 11th Dutch-Belgian Conference on Machine Learning (Benelearn 2001)*, pages 93–100, 2001.
56. A.J.M.M. Weijters and W.M.P. van der Aalst. Workflow Mining: Discovering Workflow Models from Event-Based Data. In C. Dousson, F. Höppner, and R. Quiniou, editors, *Proceedings of the ECAI Workshop on Knowledge Discovery and Spatial Data*, pages 78–84, 2002.
57. WFMC. Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface – XML Process Definition Language (XPDL) (WFMC-

TC-1025). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2002.

About the authors

Wil van der Aalst is a full professor of Information Systems and head of the section of Information and Technology of the Department of Technology Management at Eindhoven University of Technology. He is also a part-time full professor at the Computing Science faculty at the department of Mathematics and Computer Science at the same university and an adjoint professor at the Centre for Technology Innovation (CITI) of Queensland University of Technology (QUT). His research interests include information systems, simulation, Petri nets, process models, workflow management systems, verification techniques, enterprise resource planning systems, computer supported cooperative work, and interorganizational business processes.

Ton Weijters is associate professor at the Department of Technology Management of the Eindhoven University of Technology (TUE), and member of the BETA research group. Currently he is working on (i) the application of Knowledge Engineering and Machine Learning techniques for planning, scheduling, and process mining (ii) fundamental research in the domain of Machine Learning and Knowledge Discovering. He is the author of many scientific publications in the mentioned research field.