Wil M. P. van der Aalst

Process Mining

Discovery, Conformance and Enhancement of Business Processes

Springer

Process Mining

Wil M.P. van der Aalst

Process Mining

Discovery, Conformance and Enhancement of Business Processes

Springer

Wil M.P. van der Aalst
Department Mathematics & Computer Science
Eindhoven University of Technology
Den Dolech 2
5612 AZ Eindhoven
The Netherlands
w.m.p.v.d.aalst@tue.nl

ISBN 978-3-642-19344-6 e-ISBN 978-3-642-19345-3
DOI 10.1007/978-3-642-19345-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011926240

ACM Computing Classification (1998): H.4.1, H.2.8, I.2.6, F.3.2, D.2.2, J.1

Cover design: deblik

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Thanks to Karin for understanding that science is more rewarding than running errands

Thanks to all people that contributed to ProM; the fruits of their efforts demonstrate that sharing a common goal is more meaningful than "cashing in the next publon"[1]

In remembrance of Gerry Straatman-Beelen (1932–2010)

[1] publon = smallest publishable unit

Preface

Process mining provides a new means to improve processes in a variety of application domains. There are two main drivers for this new technology. On the one hand, more and more events are being recorded thus providing detailed information about the history of processes. Despite the omnipresence of event data, most organizations diagnose problems based on fiction rather than facts. On the other hand, vendors of Business Process Management (BPM) and Business Intelligence (BI) software have been promising miracles. Although BPM and BI technologies received lots of attention, they did not live up to the expectations raised by academics, consultants, and software vendors.

Process mining is an emerging discipline providing comprehensive sets of tools to provide fact-based insights and to support process improvements. This new discipline builds on process model-driven approaches and data mining. However, process mining is much more than an amalgamation of existing approaches. For example, existing data mining techniques are too data-centric to provide a comprehensive understanding of the end-to-end processes in an organization. BI tools focus on simple dashboards and reporting rather than clear-cut business process insights. BPM suites heavily rely on experts modeling idealized to-be processes and do not help the stakeholders to understand the as-is processes.

This book presents a range of process mining techniques that help organizations to uncover their actual business processes. Process mining is not limited to process discovery. By tightly coupling event data and process models, it is possible to check conformance, detect deviations, predict delays, support decision making, and recommend process redesigns. Process mining breathes life into otherwise static process models and puts today's massive data volumes in a process context. Hence, managements trends related to process improvement (e.g., Six Sigma, TQM, CPI, and CPM) and compliance (SOX, BAM, etc.) can benefit from process mining.

Process mining, as described in this book, emerged in the last decade [102, 106]. However, the roots date back about half a century. For example, Anil Nerode presented an approach to synthesize finite-state machines from example traces in 1958 [71], Carl Adam Petri introduced the first modeling language adequately capturing concurrency in 1962 [73], and Mark Gold was the first to systematically explore different notions of learnability in 1967 [45]. When data mining started to flourish in the nineties, little attention was given to processes. Moreover, only recently event logs have become omnipresent thus enabling end-to-end process discovery. Since the first survey on process mining in 2003 [102], progress has been spectacular. Process mining techniques have become mature and supported by various tools. Moreover, whereas initially the primary focus was on process discovery, the process mining spectrum has broadened markedly. For instance, conformance checking, multi-perspective process mining, and operational support have become integral parts of ProM, one of the leading process mining tools.

This is the first book on process mining. Therefore, the intended audience is quite broad. The book provides a comprehensive overview of the state-of-the-art in process mining. It is intended as an introduction to the topic for practitioners, students, and academics. On the one hand, the book is accessible for people that are new to the topic. On the other hand, the book does not avoid explaining important concepts on a rigorous manner. The book aims to be self-contained while covering the entire process mining spectrum from process discovery to operational support. Therefore, it also serves as a reference handbook for people dealing with BPM or BI on a day-to-day basis.

The reader can immediately put process mining into practice due to the applicability of the techniques, the availability of (open-source) process mining software, and the abundance of event data in today's information systems. I sincerely hope that you enjoy reading this book and start using some of the amazing process mining techniques available today.

Schleiden, Germany
December 2010

Wil M.P. van der Aalst

Acknowledgements

Many individuals and organizations contributed to the techniques and tools described in this book. Therefore, it is important to acknowledge their support, efforts, and contributions.

All of this started in 1999 with a research project named "Process Design by Discovery: Harvesting Workflow Knowledge from Ad-hoc Executions" initiated by Ton Weijters and myself. At that time, I was still working as a visiting professor at the University of Colorado in Boulder. However, the research school BETA had encouraged me to start collaborating with existing staff in my new research group at TU/e (Eindhoven University of Technology). After talking to Ton it was clear that we could benefit from combining his knowledge of machine learning with my knowledge of workflow management and Petri nets. Process mining (at that time we called it workflow mining) was the obvious topic for which we could combine our expertise. This was the start of a very successful collaboration. Thanks Ton!

Since then many PhD students have been working on the topic: Laura Maruster, Ana Karla Alves de Medeiros, Boudewijn van Dongen, Minseok Song, Christian Günther, Anne Rozinat, Carmen Bratosin, R.P. Jagadeesh Chandra (JC) Bose, Ronny Mans, Maja Pesic, Joyce Nakatumba, Helen Schonenberg, Arya Adriansyah, and Joos Buijs. I'm extremely grateful for their efforts.

Ana Karla Alves de Medeiros was the first PhD student to work on the topic under my supervision (genetic process mining). She did a wonderful job; her thesis on genetic process mining was awarded with the prestigious ASML 2007 Promotion Prize and was selected as the best thesis by the KNAW research school BETA. Also Boudewijn van Dongen has been involved in the development of ProM right from the start. As a Master student he already developed the process mining tool EMiT, i.e., the predecessor of ProM. He turned out to be a brilliant PhD student and developed a variety of process mining techniques. Eric Verbeek did a PhD on workflow verification, but over time he got more and more involved in process mining research and the development of ProM. Many people underestimate the importance of a scientific programmer like Eric. Tool development and continuity are essential for scientific progress! Boudewijn and Eric have been the driving force behind

ProM and their contributions have been crucial for process mining research at TU/e. Moreover, they are always willing to help others. Thanks guys!

Christian Günther and Anne Rozinat joined the team in 2005. Their contributions have been of crucial importance for extending the scope of process mining and lifting the ambition level. Christian managed to make ProM look beautiful while significantly improving its performance. Moreover, his Fuzzy miner facilitated dealing with Spaghetti processes. Anne managed to widen the process mining spectrum by adding conformance checking and multi-perspective process mining to ProM. It is great that they succeeded in founding a process mining company (Fluxicon). Another person crucial for the development of ProM is Peter van den Brand. He set up the initial framework and played an important role in the development of the architecture of ProM 6. Based on his experiences with ProM, he set up a process mining company (Futura Process Intelligence). It is great to work with people like Peter, Christian, and Anne; they are essential for turning research results into commercial products. I sincerely hope that Fluxicon and Futura Process Intelligence continue to be successful (not only because of prospective sports cars . . .).

Academics of various universities contributed to ProM and supported our process mining research. We are grateful to the Technical University of Lisbon, Katholieke Universiteit Leuven, Universitat Politècnica de Catalunya, Universität Paderborn, University of Rostock, Humboldt-Universität zu Berlin, University of Calabria, Queensland University of Technology, Tsinghua University, Universität Innsbruck, Ulsan National Institute of Science and Technology, Università di Bologna, Zhejiang University, Vienna University of Technology, Universität Ulm, Open University, Jilin University, University of Padua, and University of Nancy for their help. I would also like to thank the members of the IEEE Task Force on Process Mining for promoting the topic. We are grateful to all other organizations that supported process mining research at TU/e: NWO, STW, EU, IOP, LOIS, BETA, SIKS, Stichting EIT Informatica Onderwijs, Pallas Athena, IBM, LaQuSo, Philips Healthcare, ESI, Jacquard, Nuffic, BPM Usergroup, and WWTF. Special thanks go to Pallas Athena for promoting the topic of process mining and their collaboration in a variety of projects. More than 100 organizations provided event logs that helped us to improve our process mining techniques. Here, I would like to explicitly mention the AMC hospital, Philips Healthcare, ASML, Ricoh, Vestia, Catharina hospital, Thales, Océ, Rijkswaterstaat, Heusden, Harderwijk, Deloitte, and all organizations involved in the SUPER, ACSI, PoSecCo, and CoSeLoG projects. We are grateful for allowing us to use their data and for providing feedback.

It is impossible to name all of the individuals that contributed to ProM or helped to advance process mining. Nevertheless, I would like to make a modest attempt. Besides the people mentioned earlier, I would like to thank Piet Bakker, Huub de Beer, Tobias Blickle, Andrea Burattin, Riet van Buul, Toon Calders, Jorge Cardoso, Josep Carmona, Alina Chipaila, Francisco Curbera, Marlon Dumas, Schahram Dustdar, Paul Eertink, Dyon Egberts, Dirk Fahland, Diogo Ferreira, Walid Gaaloul, Stijn Goedertier, Adela Grando, Gianluigi Greco, Dolf Grünbauer, Antonella Guzzo, Kees van Hee, Joachim Herbst, Arthur ter Hofstede, John Hoogland, Ivo de Jong, Ivan Khodyrev, Thom Langerwerf, Massimiliano de Leoni, Jiafei Li, Ine van der

Ligt, Zheng Liu, Niels Lohmann, Peter Hornix, Fabrizio Maggi, Jan Mendling, Frits Minderhoud, Arnold Moleman, Marco Montali, Michael zur Muehlen, Jorge Munoz-Gama, Mariska Netjes, Andriy Nikolov, Mykola Pechenizkiy, Carlos Pedrinaci, Viara Popova, Silvana Quaglini, Manfred Reichert, Hajo Reijers, Remmert Remmerts de Vries, Stefanie Rinderle-Ma, Marcello La Rosa, Michael Rosemann, Vladimir Rubin, Stefania Rusu, Eduardo Portela Santos, Natalia Sidorova, Alessandro Sperduti, Christian Stahl, Keith Swenson, Nikola Trcka, Kenny van Uden, Irene Vanderfeesten, George Varvaressos, Marc Verdonk, Sicco Verwer, Jan Vogelaar, Hans Vrins, Jianmin Wang, Teun Wagemakers, Barbara Weber, Lijie Wen, Jan Martijn van der Werf, Mathias Weske, Michael Westergaard, Moe Wynn, Bart Ydo, and Marco Zapletal for their support. Thanks to all that read earlier drafts of this book (special thanks go to Christian, Eric, and Ton for their detailed comments).

Thanks to Springer-Verlag for publishing this book. Ralf Gerstner encouraged me to write this book and handled things in a truly excellent manner. Thanks Ralf!

More than 95% of book was written in beautiful Schleiden. Despite my sabbatical, there were many other tasks competing for attention. Thanks to my weekly visits to Schleiden (without Internet access!), it was possible to write this book in a three month period. The excellent coffee of Serafin helped when proofreading the individual chapters, the scenery did the rest.

As always, acknowledgements end with thanking the people most precious. Lion's share of credits should go to Karin, Anne, Willem, Sjaak, and Loes. They often had to manage without me under difficult circumstances. Without their continuing support, this book would have taken ages.

Schleiden, Germany Wil M.P. van der Aalst
December 2010

Contents

Chapter 1
Introduction

Information systems are becoming more and more intertwined with the operational processes they support. As a result, multitudes of events are recorded by today's information systems. Nevertheless, organizations have problems extracting value from these data. The goal of *process mining* is to use event data to extract process-related information, e.g., to automatically *discover* a process model by observing events recorded by some enterprise system. To show the importance of process mining, this chapter discusses the spectacular growth of event data and links this to the limitations of classical approaches to business process management. To explain the basic concepts, a small example is used. Finally, it is shown that process mining can play an important role in realizing the promises made by contemporary management trends such as SOX and Six Sigma.

1.1 Data Explosion

The expanding capabilities of information systems and other systems that depend on computing, are well characterized by Moore's law. Gordon Moore, the cofounder of Intel, predicted in 1965 that the number of components in integrated circuits would double every year. During the last fifty years, the growth has indeed been exponential, albeit at a slightly slower pace. For example, the number of transistors on integrated circuits has been doubling every two years. Disk capacity, performance of computers per unit cost, the number of pixels per dollar, etc. have been growing at a similar pace. Besides these incredible technological advances, people and organizations depend more and more on computerized devices and information sources on the Internet. The IDC Digital Universe Study of May 2010 illustrates the spectacular growth of data [56]. This study estimates that the amount of digital information (cf. personal computers, digital cameras, servers, sensors) stored exceeds 1 Zettabyte and predicts that the "digital universe" will to grow to 35 Zettabytes in 2010. The IDC study characterizes 35 Zettabytes as a "stack of DVDs reaching halfway to Mars". This is what we refer to as the *data explosion*.

From Bits to Zettabytes

A "bit" is the smallest unit of information possible. One bit has two possible values: 1 (on) and 0 (off). A "byte" is composed of 8 bits and can represent $2^8 = 256$ values. To talk about larger amounts of data, multiples of 1000 are used: 1 Kilobyte (KB) equals 1000 bytes, 1 Megabyte (MB) equals 1000 KB, 1 Gigabyte (GB) equals 1000 MB, 1 Terabyte (TB) equals 1000 GB, 1 Petabyte (PB) equals 1000 TB, 1 Exabyte (EB) equals 1000 PB, and 1 Zettabyte (ZB) equals 1000 EB. Hence, 1 Zettabyte is $10^{21} = 1{,}000{,}000{,}000{,}000{,}000{,}000{,}000$ bytes. Note that here we used the International System of Units (SI) set of unit prefixes, also known as SI prefixes, rather than binary prefixes. If we assume binary prefixes, then 1 Kilobyte is $2^{10} = 1024$ bytes, 1 Megabyte is $2^{20} = 1{,}048{,}576$ bytes, and 1 Zettabyte is $2^{70} \approx 1.18 \times 10^{21}$ bytes.

Most of the data stored in the digital universe is unstructured and organizations have problems dealing with such large quantities of data. One of the main challenges of today's organizations is to extract information and value from data stored in their information systems.

The importance of information systems is not only reflected by the spectacular growth of data, but also by the role that these systems play in today's business processes as the digital universe and the physical universe are becoming more and more aligned. For example, the "state of a bank" is mainly determined by the data stored in the bank's information system. Money has become a predominantly digital entity. When booking a flight over the Internet, the customer is interacting with many organizations (airline, travel agency, bank, and various brokers), often without actually realizing it. If the booking is successful, the customer receives an e-ticket. Note that an e-ticket is basically a number, thus illustrating the alignment between the digital and physical universe. When the SAP system of a large manufacturer indicates that a particular product is out of stock, it is impossible to sell or ship the product even when it is available in physical form. Technologies such as RFID (Radio Frequency Identification), GPS (Global Positioning System), and sensor networks will stimulate a further alignment of the digital universe and the physical universe. RFID tags make it possible to track and trace individual items. Also note that more and more devices are being monitored. For example, Philips Healthcare is monitoring its medical equipment (e.g., X-ray machines and CT scanners) all over the world. This helps Philips to understand the needs of customers, test their systems under realistic circumstances, anticipate problems, service systems remotely, and learn from recurring problems. The success of the "App Store" of Apple illustrates that location-awareness combined with a continuous Internet connection enables new ways to pervasively intertwine the digital universe and the physical universe.

The growth of a digital universe that is well-aligned with processes in organizations makes it possible to record and analyze *events*. Events may range from the withdrawal of cash from an ATM, a doctor setting the dosage of an X-ray machine, a citizen applying for a driver license, the submission of a tax declaration, and the receipt of an e-ticket number by a traveler. The challenge is to *exploit event data in a meaningful way*, for example, to provide insights, identify bottlenecks, anticipate problems, record policy violations, recommend countermeasures, and streamline processes. This is what process mining is all about!

1.2 Limitations of Modeling

Process mining, i.e., extracting valuable, process-related information from event logs, complements existing approaches to *Business Process Management* (BPM). BPM is the discipline that *combines knowledge from information technology and knowledge from management sciences and applies this to operational business processes* [93, 127]. It has received considerable attention in recent years due to its potential for significantly increasing productivity and saving cost. BPM can be seen as an extension of *Workflow Management* (WFM). WFM primarily focuses on the automation of business processes [57, 61, 98], whereas BPM has a broader scope: from process automation and process analysis to process management and the organization of work. On the one hand, BPM aims to improve operational business processes, possibly without the use of new technologies. For example, by modeling a business process and analyzing it using simulation, management may get ideas on how to reduce costs while improving service levels. On the other hand, BPM is often associated with software to manage, control, and support operational processes. This was the initial focus of WFM. Traditional WFM technology aims at the automation of business processes in a rather mechanistic manner without much attention for human factors and management support.

Process-Aware Information Systems (PAISs) include the traditional WFM systems, but also include systems that provide more flexibility or support specific tasks [37]. For example, larger ERP (Enterprise Resource Planning) systems (SAP, Oracle), CRM (Customer Relationship Management) systems, rule-based systems, call center software, high-end middleware (WebSphere), etc. can be seen as process-aware, although they do not necessarily control processes through some generic workflow engine. Instead, these systems have in common that there is an explicit process notion and that the information system is aware of the processes it supports. Also a database system or e-mail program may be used to execute steps in some business process. However, such software tools are not "aware" of the processes they are used in. Therefore, they are not actively involved in the management and orchestration of the processes they are used for. Some authors use the term BPMS (BPM system), or simply PMS (Process Management System), to refer systems that are "aware" of the processes they support. We use the term PAIS to stress that the scope is much broader than conventional workflow technology.

BPM and PAIS have in common that they heavily rely on *process models*. A plethora of notations exists to model operational business processes (e.g., Petri nets, BPMN, UML, and EPCs), some of which will be discussed in the next chapter. These notations have in common that processes are described in terms of activities

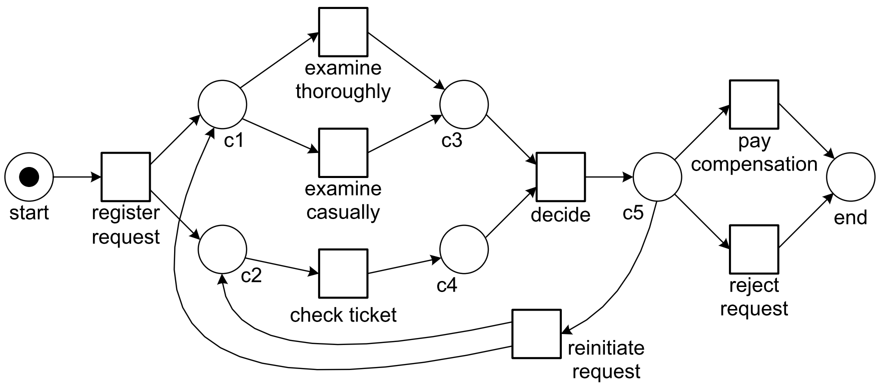

Fig. 1.1 A Petri net modeling the handling of compensation requests

(and possibly subprocesses). The ordering of these activities is modeled by describing casual dependencies. Moreover, the process model may also describe temporal properties, specify the creation and use of data, e.g., to model decisions, and stipulate the way that resources interact with the process (e.g., roles, allocation rules, and priorities).

Figure 1.1 shows a process model expressed in terms of a *Petri net* [35]. The model describes the handling of a request for compensation within an airline. Customers may request compensation for various reasons, e.g., a delayed or canceled flight. As Fig. 1.1 shows, the process starts by registering the request. This activity is modeled by transition *register request*. Each *transition* is represented by a square. Transitions are connected through *places* that model possible states of the process. Each place is represented by a circle. In a Petri net a transition is *enabled*, i.e., the corresponding activity can occur, if all input places hold a *token*. Transition *register request* has only one input place (*start*) and this place initially contains a token to represent the request for compensation. Hence, the corresponding activity is enabled and can occur. This is also referred to as *firing*. When firing, the transition consumes one token from each of its input places and produces one token for each of its output places. Hence, the firing of transition *register request* results in the removal of the token from input place *start* and the production of two tokens: one for output place $c1$ and one for output place $c2$. Tokens are shown as black dots. The configuration of tokens over places—in this case the state of the request—is referred to as *marking*. Figure 1.1 shows the initial marking consisting of one token in place *start*. The marking after firing transition *register request* has two tokens: one in place $c1$ and one in place $c2$. After firing transition *register request*, three transitions are enabled. The token in place $c2$ enables transition *check ticket*. This transition models an administrative check to see whether the customer is eligible to issue a request. For example, while executing *check ticket* it is verified whether the customer indeed has a ticket issued by the airline. In parallel, the token in $c1$ enables both *examine thoroughly* and *examine casually*. Firing *examine thoroughly* will remove the token from $c1$, thus disabling *examine casually*. Similarly, the oc-

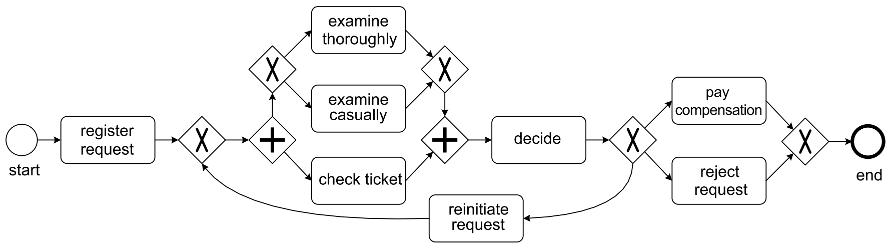

Fig. 1.2 The same process modeled in terms of BPMN

currence of *examine casually* will disable *examine thoroughly*. In other words, there is a choice between these two activities. Transition *examine thoroughly* is executed for requests that are suspicious or complex. Straightforward requests only need a casual examination. Firing *check ticket* does not disable any other transition, i.e., it can occur concurrently with *examine thoroughly* or *examine casually*. Transition *decide* is only enabled if both input places contain a token. The ticket needs to be checked (token in place $c4$) and the casual or thorough examination of the request has been conducted (token in place $c3$). Hence, the process synchronizes before making a decision. Transition *decide* consumes two tokens and produces one token for $c5$. Three transitions share $c5$ as an input place, thus modeling the three possible outcomes of the decision. The requested compensation is paid (transition *pay compensation* fires), the request is declined (transition *reject request* fires), or further processing is needed (transition *reinitiate request* fires). In the latter case, the process returns to the state marking places $c1$ and $c2$: transition *reinitiate request* consumes a token from $c5$ and produces a token for each of its output places. This was the marking directly following the occurrence of *register request*. In principle, several iterations are possible. The process ends after paying the compensation or rejecting the request.

Figure 1.1 models the process as a Petri net. There exist many different notations for process models. Figure 1.2 models the same process in terms of a so-called BPMN diagram [72, 127]. The *Business Process Modeling Notation* (BPMN) uses explicit *gateways* rather than places to model the control-flow logic. The diamonds with a "×" sign denote XOR split/join gateways, whereas diamonds with a "+" sign denote AND split/join gateways. The diamond directly following activity *register request* is an XOR-join gateway. This gateway is used to be able to "jump back" after making the decision to reinitiate the request. After this XOR-join gateway, there is an AND-split gateway to model that the checking of the ticket can be done in parallel with the selected examination type (thorough or casual). The remainder of the BPMN diagram is self explanatory as the behavior is identical to the Petri net described before.

Figures 1.1 and 1.2 show only the *control-flow*, i.e., the ordering of activities for the process described earlier. This is a rather limited view on business processes. Therefore, most modeling languages offer notations for modeling other perspectives such as the organizational or resource perspective ("The decision needs to be made by a manager"), the data perspective ("After four iteration always a decision is made unless more than 1 million Euro is claimed"), and the time perspective ("After two weeks the problem is escalated"). Although there are important differences between the various process modeling languages, we do not elaborate one these in this book. Instead, we refer to the systematic comparisons in the context of the *Workflow Patterns Initiative* [101, 130]. This allows us to focus on the role that process models play in BPM.

What Are Process Models Used for?

- *Insight*: while making a model, the modeler is triggered to view the process from various angles.
- *Discussion*: the stakeholders use models to structure discussions.
- *Documentation*: processes are documented for instructing people or certification purposes (cf. ISO 9000 quality management).
- *Verification*: process models are analyzed to find errors in systems or procedures (e.g., potential deadlocks).
- *Performance analysis*: techniques like simulation can be used to understand the factors influencing response times, service levels, etc.
- *Animation*: models enable end users to "play out" different scenarios and thus provide feedback to the designer.
- *Specification*: models can be used to describe a PAIS before it is implemented and can hence serve as a "contract" between the developer and the end user/management.
- *Configuration*: models can be used to configure a system.

Clearly, process models play an important role in larger organizations. When redesigning processes and introducing new information systems, process models are used for a variety of reasons. Typically, two types of models are used: (a) *informal models* and (b) *formal models* (also referred to as "executable" models). Informal models are used for discussion and documentation whereas formal models are used for analysis or enactment (i.e., the actual execution of process). On the one end of the spectrum there are "PowerPoint diagrams" showing high-level processes whereas on the other end of the spectrum there are process models captured in executable code. Whereas informal models are typically ambiguous and vague, formal models tend to have a rather narrow focus or are too detailed to be understandable by the stakeholders. The lack of alignment between both types of models has been discussed extensively in BPM literature [37, 53, 90, 93, 100, 127, 131]. Here, we would like to provide another view on the matter. Independent of the kind of model—informal or formal—one can reflect on the alignment between model and reality. A process model used to configure a workflow management system is probably well-aligned with reality as the model is used to force people to work in a particular way. Unfortunately, most hand-made models are disconnected from reality and provide only an idealized view on the processes at hand. Moreover, also formal models that allow for rigorous analysis techniques may have little to do with the actual process.

The value of models is limited if too little attention is paid to the alignment of model and reality. Process models become "paper tigers" when the people involved cannot trust them. For example, it makes no sense to conduct simulation experiments while using a model that assumes an idealized version of the real process. It is likely that—based on such an idealized model—incorrect redesign decisions are made. It is also precarious to start a new implementation project guided by process models that hide reality. A system implemented on the basis of idealized models is likely to be disruptive and unacceptable for end users. A nice illustration is the limited quality of most *reference models*. Reference models are used in the context of large enterprise systems such as SAP [25] but also to document processes for particular branches, cf. the NVVB (Nederlandse Vereniging Voor Burgerzaken) models describing the core processes in Dutch municipalities. The idea is that "best practices" are shared among different organizations. Unfortunately, the quality of such models leaves much to be desired. For example, the SAP reference model has very little to do with the processes actually supported by SAP. In fact, more than 20 percent of the SAP models contain serious flaws (deadlocks, livelocks, etc.) [66]. Such models are not aligned with reality and, thus, have little value for end users.

Given (a) the interest in process models, (b) the abundance of event data, and (c) the limited quality of hand-made models, it seems worthwhile to relate event data to process models. This way the actual processes can be discovered and existing process models can be evaluated and enhanced. This is precisely what process mining aims to achieve.

1.3 Process Mining

To position process mining, we first describe the so-called *BPM life-cycle* using Fig. 1.3. The life-cycle describes the different phases of managing a particular business process. In the *design* phase, a process is designed. This model is transformed into a running system in the *configuration/implementation* phase. If the model is already in executable form and a WFM or BPM system is already running, this phase may be very short. However, if the model is informal and needs to be hard-coded in conventional software, this phase may take substantial time. After the system supports the designed processes, the *enactment/monitoring* phase starts. In this phase, the processes are running while being monitored by management to see if any changes are needed. Some of these changes are handled in the *adjustment* phase shown in Fig. 1.3. In this phase, the process is not redesigned and no new software is created; only predefined controls are used to adapt or reconfigure the process. The *diagnosis/requirements* phase evaluates the process and monitors emerging requirements due to changes in the environment of the process (e.g., changing policies, laws, competition). Poor performance (e.g., inability to meet service levels) or new demands imposed by the environment may trigger a new iteration of the BPM life-cycle starting with the *redesign* phase.

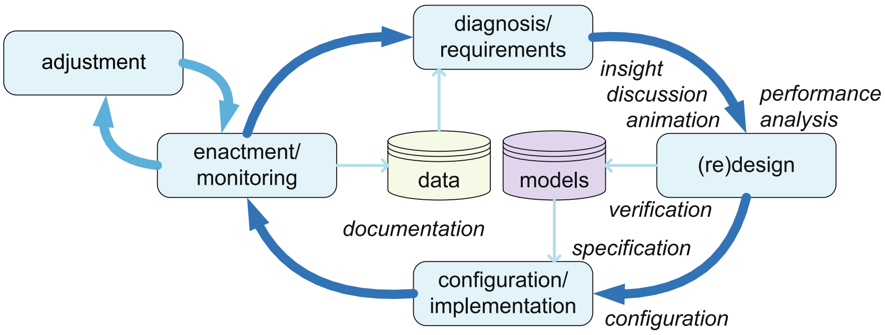

Fig. 1.3 The BPM life-cycle showing the different uses of process models

As Fig. 1.3 shows, process models play a dominant role in the (re)design and configuration/implementation phases, whereas data plays a dominant role in the enactment/monitoring and diagnosis/requirements phases. The figure also lists the different ways in which process models are used (as identified in Sect. 1.2). Until recently, there were few connections between the data produced while executing the process and the actual process design. In fact, in most organizations the diagnosis/requirements phase is not supported in a systematic and continuous manner. Only severe problems or major external changes will trigger another iteration of the life-cycle, and factual information about the current process is not actively used in redesign decisions. Process mining offers the possibility to truly "close" the BPM life-cycle. Data recorded by information systems can be used to provide a better view on the actual processes, i.e., deviations can be analyzed and the quality of models can be improved.

Process mining is a relative young research discipline that sits between machine learning and data mining on the one hand and process modeling and analysis on the other hand. The idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's systems.

Figure 1.4 shows that process mining establishes links between the actual processes and their data on the one hand and process models on the other hand. As explained in Sect. 1.1, the digital universe and the physical universe become more and more aligned. Today's information systems log enormous amounts of events. Classical WFM systems (e.g., Staffware and COSA), BPM systems (e.g., BPM|one by Pallas Athena, SmartBPM by Pegasystems, FileNet, Global 360, and Teamwork by Lombardi Software), ERP systems (e.g., SAP Business Suite, Oracle E-Business Suite, and Microsoft Dynamics NAV), PDM systems (e.g., Windchill), CRM systems (e.g., Microsoft Dynamics CRM and SalesForce), middleware (e.g., IBM's WebSphere and Cordys Business Operations Platform), and hospital information systems (e.g., Chipsoft and Siemens Soarian) provide detailed information about the activities that have been executed. Figure 1.4 refers to such data as *event logs*. All of the PAISs just mentioned directly provide such event logs.

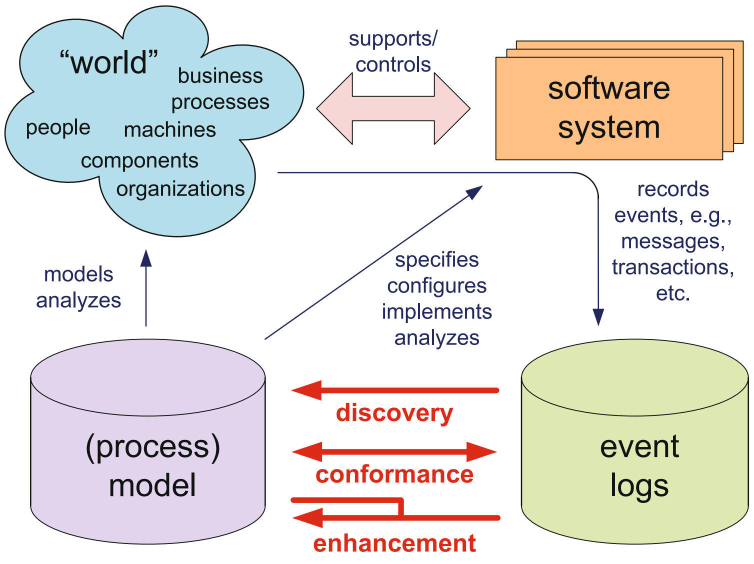

Fig. 1.4 Positioning of the three main types of process mining: *discovery*, *conformance*, and *enhancement*

However, most information systems store such information in unstructured form, e.g., event data is scattered over many tables or needs to be tapped off from subsystems exchanging messages. In such cases, event data exist but some efforts are needed to extract them. Data extraction is an integral part of any process mining effort.

Let us assume that it is possible to *sequentially record events* such that each event refers to an *activity* (i.e., a well-defined step in the process) and is related to a particular *case* (i.e., a process instance). Consider, for example, the handling of requests for compensation modeled in Fig. 1.1. The cases are individual requests and per case a *trace* of events can be recorded. An example of a possible trace is ⟨*register request*, *examine casually*, *check ticket*, *decide*, *reinitiate request*, *check ticket*, *examine thoroughly*, *decide*, *pay compensation*⟩. Here activity names are used to identify events. However, there are two *decide* events that occurred at different times (the fourth and eighth event of the trace), produced different results, and may have been conducted by different people. Obviously, it is important to distinguish these two decisions. Therefore, most event logs store additional information about events. In fact, whenever possible, process mining techniques use extra information such as the *resource* (i.e., person or device) executing or initiating the activity, the *timestamp* of the event, or *data elements* recorded with the event (e.g., the size of an order).

Event logs can be used to conduct three types of process mining as shown in Fig. 1.4.

The first type of process mining is *discovery*. A discovery technique takes an event log and produces a model without using any a-priori information. An example is the α-algorithm [103] that will be described in Chap. 5. This algorithm takes an event log and produces a Petri net explaining the behavior recorded in the log. For example, given sufficient example executions of the process shown in Fig. 1.1, the α-algorithm is able to automatically construct the Petri net without using any additional knowledge. If the event log contains information about resources, one can also discover resource-related models, e.g., a social network showing how people work together in an organization.

The second type of process mining is *conformance*. Here, an existing process model is compared with an event log of the same process. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa. For instance, there may be a process model indicating that purchase orders of more than one million Euro require two checks. Analysis of the event log will show whether this rule is followed or not. Another example is the checking of the so-called "four-eyes" principle stating that particular activities should not be executed by one and the same person. By scanning the event log using a model specifying these requirements, one can discover potential cases of fraud. Hence, conformance checking may be used to detect, locate and explain deviations, and to measure the severity of these deviations. An example is the conformance checking algorithm described in [80]. Given the model shown in Fig. 1.1 and a corresponding event log, this algorithm can quantify and diagnose deviations.

The third type of process mining is *enhancement*. Here, the idea is to extend or improve an existing process model using information about the actual process recorded in some event log. Whereas conformance checking measures the alignment between model and reality, this third type of process mining aims at changing or extending the a-priori model. One type of enhancement is *repair*, i.e., modifying the model to better reflect reality. For example, if two activities are modeled sequentially but in reality can happen in any order, then the model may be corrected to reflect this. Another type of enhancement is *extension*, i.e., adding a new perspective to the process model by cross-correlating it with the log. An example is the extension of a process model with performance data. For instance, by using timestamps in the event log of the "request for compensation" process, one can extend Fig. 1.1 to show bottlenecks, service levels, throughput times, and frequencies. Similarly, Fig. 1.1 can be extended with information about resources, decision rules, quality metrics, etc.

As indicated earlier, process models such as depicted in Figs. 1.1 and 1.2 show only the control-flow. However, when extending process models, additional perspectives are added. Moreover, discovery and conformance techniques are not limited to control-flow. For example, one can discover a social network and check the validity of some organizational model using an event log. Hence, orthogonal to the three types of mining (discovery, conformance, and enhancement), different perspectives can be identified.

In the remainder, we consider the following *perspectives*.

- The *control-flow 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 or some other notation (e.g., EPCs, BPMN, and UML ADs).
- The *organizational perspective* focuses on information about resources hidden in the log, i.e., which actors (e.g., people, systems, roles, and departments) 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 the social network.
- The *case perspective* focuses on properties of cases. Obviously, a case can be characterized by its path in the process or by the originators working on it. However, cases can also be characterized by the values of the corresponding data elements. For example, if a case represents a replenishment order, it may be interesting to know the supplier or the number of products ordered.
- The *time perspective* is concerned with the timing and frequency of events. When events bear timestamps it is possible to discover bottlenecks, measure service levels, monitor the utilization of resources, and predict the remaining processing time of running cases.

Note that the different perspectives are partially overlapping and non-exhaustive. Nevertheless, they provide a good characterization of the aspects that process mining aims to analyze.

In most examples given thus far it is assumed that process mining is done *off-line*, i.e., processes are analyzed afterward to see how they can be improved or better understood. However, more and more process mining techniques can also be used in an *online* setting. We refer to this as *operational support*. An example is the detection of nonconformance at the moment the deviation actually takes place. Another example is time prediction for running cases, i.e., given a partially executed case the remaining processing time is estimated based on historic information of similar cases. This illustrates that the "process mining spectrum" is broad and not limited to process discovery. In fact, today's process mining techniques are indeed able to support the whole BPM life-cycle shown in Fig. 1.3. Process mining is not only relevant for the design and diagnosis/requirements phases, but also for the enactment/monitoring and adjustment phases.

1.4 Analyzing an Example Log

After providing an overview of process mining and positioning it in the broader BPM discipline, we use the event log shown in Table 1.1 to clarify some of the foundational concepts. The table shows just a fragment of a possible log corresponding to the handling of requests for compensation. Each line presents one event. Note that events are already grouped per case. Case 1 has five associated events. The first event of Case 1 is the execution of activity *register request* by Pete on December 30th, 2010. Table 1.1 also shows a unique id for this event: 35654423. This is merely used for the identification of the event, e.g., to distinguish it from event 35654483 that also corresponds to the execution of activity *register request* (first event of second case). Table 1.1 shows a date and a timestamp for each event. In some event logs, this information is more coarse-grained and only a date or partial ordering of events is given. In other logs, there may be more elaborate timing information also showing when the activity was started, when it was completed, and sometimes even when it was offered to the resource. The times shown in Table 1.1 should be interpreted as completion times. In this particular event log, activities are considered to be atomic and the table does not reveal the duration of activities. In the table, each event is associated to a resource. In some event logs, this information will be missing. In other logs, more detailed information about resources may be stored, e.g., the role a resource has or elaborate authorization data. The table also shows the costs associated to events. This is an example of a data attribute. There may be many other data attributes. For example, in this particular example it would be interesting to record the outcome of the different types of examinations and checks. Another data element that could be useful for analysis is the amount of compensation requested. This could be an attribute of the whole case or stored as an attribute of the *register request* event.

Table 1.1 illustrates the typical information present in an event log. Depending on the process mining technique used and the questions at hand, only part of this information is used. The minimal requirements for process mining are that any event can be related to both a case and an activity and that events within a case are ordered. Hence, the "case id" and "activity" columns in Table 1.1 represent the bare minimum for process mining. By projecting the information in these two columns, we obtain the more compact representation shown in Table 1.2. In this table, each case is represented by a sequence of activities also referred to as *trace*. For clarity, the activity names have been transformed into single-letter labels, e.g., a denotes activity *register request*.

Process mining algorithms for process discovery can transform the information shown in Table 1.2 into process models. For instance, the basic α-algorithm [103] discovers the Petri net described earlier when providing it with the input data in Table 1.2. Figure 1.5 shows the resulting model with the compact labels just introduced. It is easy to check that all six traces in Table 1.2 are possible in the model. Let us replay the trace of the first case—$\langle a, b, d, e, h\rangle$—to show that the trace "fits" (i.e., conforms to) the model. In the initial marking shown in Fig. 1.5, a is indeed enabled because of the token in *start*. After firing a places $c1$ and $c2$ are marked, i.e., both places contain a token. b is enabled at this marking and its execution results in the marking with tokens in $c2$ and $c3$. Now we have executed $\langle a, b\rangle$ and the sequence $\langle d, e, h\rangle$ remains. The next event d is indeed enabled and its execution results in the marking enabling e (tokens in places $c3$ and $c4$). Firing e results in the marking with one token in $c5$. This marking enables the final event h in the trace.

Table 1.1 A fragment of some event log: each line corresponds to an event

Case id	Event id	Properties				
		Timestamp	Activity	Resource	Cost	...
1	35654423	30-12-2010:11.02	Register request	Pete	50	...
	35654424	31-12-2010:10.06	Examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	Check ticket	Mike	100	...
	35654426	06-01-2011:11.18	Decide	Sara	200	...
	35654427	07-01-2011:14.24	Reject request	Pete	200	...
2	35654483	30-12-2010:11.32	Register request	Mike	50	...
	35654485	30-12-2010:12.12	Check ticket	Mike	100	...
	35654487	30-12-2010:14.16	Examine casually	Pete	400	...
	35654488	05-01-2011:11.22	Decide	Sara	200	...
	35654489	08-01-2011:12.05	Pay compensation	Ellen	200	...
3	35654521	30-12-2010:14.32	Register request	Pete	50	...
	35654522	30-12-2010:15.06	Examine casually	Mike	400	...
	35654524	30-12-2010:16.34	Check ticket	Ellen	100	...
	35654525	06-01-2011:09.18	Decide	Sara	200	...
	35654526	06-01-2011:12.18	Reinitiate request	Sara	200	...
	35654527	06-01-2011:13.06	Examine thoroughly	Sean	400	...
	35654530	08-01-2011:11.43	Check ticket	Pete	100	...
	35654531	09-01-2011:09.55	Decide	Sara	200	...
	35654533	15-01-2011:10.45	Pay compensation	Ellen	200	...
4	35654641	06-01-2011:15.02	Register request	Pete	50	...
	35654643	07-01-2011:12.06	Check ticket	Mike	100	...
	35654644	08-01-2011:14.43	Examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	Decide	Sara	200	...
	35654647	12-01-2011:15.44	Reject request	Ellen	200	...
5	35654711	06-01-2011:09.02	Register request	Ellen	50	...
	35654712	07-01-2011:10.16	Examine casually	Mike	400	...
	35654714	08-01-2011:11.22	Check ticket	Pete	100	...
	35654715	10-01-2011:13.28	Decide	Sara	200	...
	35654716	11-01-2011:16.18	Reinitiate request	Sara	200	...
	35654718	14-01-2011:14.33	Check ticket	Ellen	100	...
	35654719	16-01-2011:15.50	Examine casually	Mike	400	...
	35654720	19-01-2011:11.18	Decide	Sara	200	...
	35654721	20-01-2011:12.48	Reinitiate request	Sara	200	...
	35654722	21-01-2011:09.06	Examine casually	Sue	400	...
	35654724	21-01-2011:11.34	Check ticket	Pete	100	...
	35654725	23-01-2011:13.12	Decide	Sara	200	...
	35654726	24-01-2011:14.56	Reject request	Mike	200	...

Table 1.1 (Continued)

Case id	Event id	Properties				
		Timestamp	Activity	Resource	Cost	...
6	35654871	06-01-2011:15.02	Register request	Mike	50	...
	35654873	06-01-2011:16.06	Examine casually	Ellen	400	...
	35654874	07-01-2011:16.22	Check ticket	Mike	100	...
	35654875	07-01-2011:16.52	Decide	Sara	200	...
	35654877	16-01-2011:11.47	Pay compensation	Mike	200	...
...	...	...	...	...	...	...

Table 1.2 A more compact representation of log shown in Table 1.1: a = *register request*, b = *examine thoroughly*, c = *examine casually*, d = *check ticket*, e = *decide*, f = *reinitiate request*, g = *pay compensation*, and h = *reject request*

Case id	Trace
1	$\langle a, b, d, e, h\rangle$
2	$\langle a, d, c, e, g\rangle$
3	$\langle a, c, d, e, f, b, d, e, g\rangle$
4	$\langle a, d, b, e, h\rangle$
5	$\langle a, c, d, e, f, d, c, e, f, c, d, e, h\rangle$
6	$\langle a, c, d, e, g\rangle$
...	...

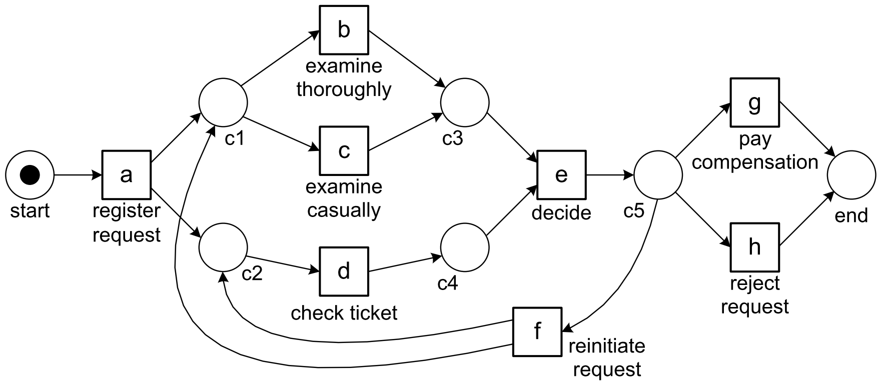

Fig. 1.5 The process model discovered by the α-algorithm [103] based on the set of traces $\{\langle a, b, d, e, h\rangle, \langle a, d, c, e, g\rangle, \langle a, c, d, e, f, b, d, e, g\rangle, \langle a, d, b, e, h\rangle, \langle a, c, d, e, f, d, c, e, f, c, d, e, h\rangle, \langle a, c, d, e, g\rangle\}$

After executing h, the case ends in the desired final marking with just a token in place *end*. Similarly, it can be checked that the other five traces shown in Table 1.2 are also possible in the model and that all of these traces result in the marking with just a token in place *end*.

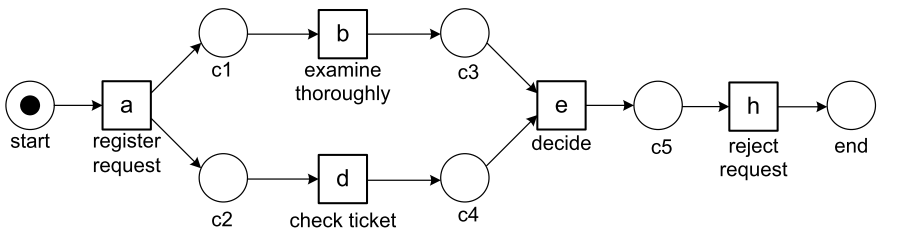

Fig. 1.6 The process model discovered by the α-algorithm based on Cases 1 and 4, i.e., the set of traces $\{\langle a, b, d, e, h\rangle, \langle a, d, b, e, h\rangle\}$

The Petri net shown in Fig. 1.5 also allows for traces not present in Table 1.2. For example, the traces $\langle a, d, c, e, f, b, d, e, g\rangle$ and $\langle a, c, d, e, f, c, d, e, f, c, d, e, f, c, d, e, f, b, d, e, g\rangle$ are also possible. This is a desired phenomenon as the goal is *not* to represent just the *particular set of example traces* in the event log. Process mining algorithms need to generalize the behavior contained in the log to show the most likely underlying model that is not invalidated by the next set of observations. One of the challenges of process mining is to balance between "overfitting" (the model is too specific and only allows for the "accidental behavior" observed) and "underfitting" (the model is too general and allows for behavior unrelated to the behavior observed).

When comparing the event log and the model, there seems to be a good balance between "overfitting" and "underfitting". All cases start with a and end with either g or h. Every e is preceded by d and one of the examination activities (b or c). Moreover, e is followed by f, g, or h. The repeated execution of b or c, d, and e suggests the presence of a loop. These characteristics are adequately captured by the net of Fig. 1.5.

Let us now consider an event log consisting of only two traces $\langle a, b, d, e, h\rangle$ and $\langle a, d, b, e, h\rangle$, i.e., Cases 1 and 4 of the original log. For this log, the α-algorithm constructs the Petri net shown in Fig. 1.6. This model only allows for two traces and these are exactly the ones in the small event log. b and d are modeled as being concurrent because they can be executed in any order. For larger and more complex models, it is important to discover concurrency. Not modeling concurrency typically results in large "Spaghetti-like" models in which the same activity needs to be duplicated.[1]

The α-algorithm is just one of many possible process discovery algorithms. For real-life logs, more advanced algorithms are needed to better balance between "overfitting" and "underfitting" and to deal with "incompleteness" (i.e., logs containing only a small fraction of the possible behavior due to the large number of alternatives) and "noise" (i.e., logs containing exceptional/infrequent behavior that should not automatically be incorporated in the model). This book will describe several of such algorithms and guide the reader in selecting one. In this section, we used Petri nets

[1]See, for example, Figs.12.1 and 12.10 to understand why we use the term "Spaghetti" to refer to models that are difficult to comprehend.

Table 1.3 Another event log: Cases 7, 8, and 10 are not possible according to Fig. 1.5

Case id	Trace
1	$\langle a, b, d, e, h\rangle$
2	$\langle a, d, c, e, g\rangle$
3	$\langle a, c, d, e, f, b, d, e, g\rangle$
4	$\langle a, d, b, e, h\rangle$
5	$\langle a, c, d, e, f, d, c, e, f, c, d, e, h\rangle$
6	$\langle a, c, d, e, g\rangle$
7	$\langle \mathbf{a}, \mathbf{b}, \mathbf{e}, \mathbf{g}\rangle$
8	$\langle \mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{e}\rangle$
9	$\langle a, d, c, e, f, d, c, e, f, b, d, e, h\rangle$
10	$\langle \mathbf{a}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{b}, \mathbf{d}, \mathbf{g}\rangle$

to represent the discovered process models, because Petri nets are a succinct way of representing processes and have unambiguous and simple semantics. However, most mining techniques are independent of the desired representation. For instance, the discovered Petri net model shown in Fig. 1.5 can be (automatically) transformed into the BPMN model shown in Fig. 1.2.

As explained in Sect. 1.3, process mining is not limited to process discovery. Event logs can be used to check conformance and enhance existing models. Moreover, different perspectives may be taken into account. To illustrate this, let us first consider the event log shown in Table 1.3. The first six cases are as before. It is easy to see that Case 7 with trace $\langle a, b, e, g\rangle$ is not possible according to the model in Fig. 1.5. The model requires the execution of d before e, but d did not occur. This means that the ticket was not checked at all before making a decision and paying compensation. Conformance checking techniques aim at discovering such discrepancies [80]. When checking the conformance of the remainder of the event log, it can also be noted that Cases 8 and 10 do not conform either. Case 9 conforms although it is not identical to one of the earlier traces. Trace $\langle a, b, d, e\rangle$ (i.e., Case 8) has the problem that no concluding action was taken (rejection or payment). Trace $\langle a, c, d, e, f, b, d, g\rangle$ (Case 10) has the problem that the airline paid compensation without making a final decision. Note that conformance can be viewed from two angles: (a) the model does not capture the real behavior ("the model is wrong") and (b) reality deviates from the desired model ("the event log is wrong"). The first viewpoint is taken when the model is supposed to be *descriptive*, i.e., capture or predict reality. The second viewpoint is taken when the model is *normative*, i.e., used to influence or control reality.

The original event log shown in Table 1.1 also contains information about resources, timestamps and costs. Such information can be used to discover other perspectives, check the conformance of models that are not pure control-flow models, and to extend models with additional information. For example, one could derive a social network based on the interaction patterns between individuals. The social network can be based on the "handover of work" metric, i.e., the more frequent in-