Business Process Management: A Comprehensive Survey

Wil M.P. van der Aalst

Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, The Netherlands.
www.vdaalst.com

Abstract. Business Process Management (BPM) research resulted in a plethora of methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes. This survey aims to structure these results and provides an overview of the state-of-the-art in BPM. In BPM the concept of a *process model* is fundamental. Process models may be used to configure information systems, but may also be used to analyze, understand, and improve the processes they describe. Hence, the introduction of BPM technology has both managerial and technical ramifications, and may enable significant productivity improvements, cost savings, and flow-time reductions. The practical relevance of BPM and rapid developments over the last decade justify a comprehensive survey.

1 Introduction

Business Process Management (BPM) is the discipline that *combines knowledge from information technology and knowledge from management sciences and applies this to operational business processes* [1, 2]. It has received considerable attention in recent years due to its potential for significantly increasing productivity and saving costs. Moreover, today there is an abundance of *BPM systems*. These systems are *generic software systems that are driven by explicit process designs to enact and manage operational business processes* [3].

BPM can be seen as an extension of *Workflow Management* (WFM). WFM primarily focuses on the automation of business processes [4–6], whereas BPM has a broader scope: from process automation and process analysis to operations 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. However, traditional WFM technology aimed 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 traditional WFM systems and modern BPM systems, but also include systems that provide more flexibility or support specific processes [7]. For example, larger ERP (Enterprise Resource Planning) systems (e.g., SAP and Oracle), CRM (Customer Relationship Management) systems, case-handling systems, rule-based systems, call center software, and high-end middleware (e.g. WebSphere) 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. BPM techniques are not limited to WFM/BPM systems, but extend to any PAIS. In fact, BPM techniques such as *process mining* [8] can be used to discover and analyze emerging processes that are supported by systems that are not even "aware" of the processes they are used in.

The notion of a *process model* is foundational for BPM. A process model aims to capture the different ways in which a *case* (i.e., process instance) can be handled. A plethora of notations exists to model operational business processes (e.g., Petri nets, BPMN, UML, and EPCs). These notations have in common that processes are described in terms of activities (and possibly subprocesses). The ordering of these activities is modeled by describing causal 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).

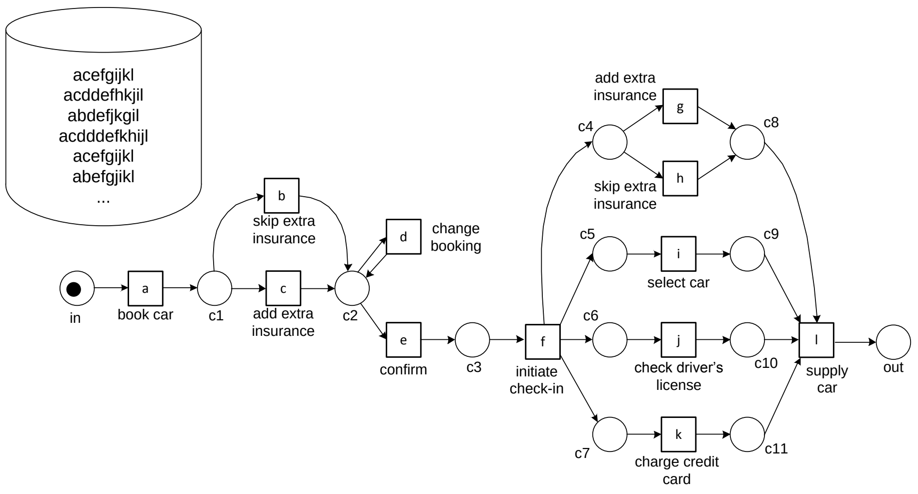

Fig. 1. A process model expressed in terms of a Petri net and an event log with some example traces.

Figure 1 shows a process model expressed in terms of a Petri net. The model allows for the scenario $\langle a, c, e, f, g, i, j, k, l \rangle$. This is the scenario where a car is booked (activity a), extra insurance is added (activity c), the booking is confirmed (activity e), the check-in process is initiated (activity f), more insurance is added (activity g), a car is selected (activity i), the license is checked (activity j), the credit card is charged (activity k), and the car is supplied (activity l). Another example scenario is $\langle a, c, d, d, e, f, h, k, j, i, l \rangle$ were the booking was changed two times (activity d) and no extra insurance was taken at check-in (activity h).

Figure 1 focuses on control-flow and does *not* model data, decisions, resources, etc. The *control-flow perspective* (modeling the ordering of activities) is often the backbone of a process model. However, other perspectives such as the *resource perspective* (modeling roles, organizational units, authorizations, etc.), the *data perspective* (modeling decisions, data creation, forms, etc.), the *time perspective* (modeling durations, deadlines, etc.), and the *function perspective* (describing activities and related applications) are also essential for comprehensive process models.

The Petri net notation is used to model the control-flow in Figure 1. However, various alternative notations (e.g., BPMN, UML, and EPCs) could have been used. Discussions on different notations tend to distract BPM professionals from the key issues. The workflow patterns [9] describe the key functionalities in a language-independent manner. Obviously, there are differences in expressiveness and suitability among languages, however, these are only relevant for the more advanced patterns. Moreover, the study in [10] revealed that business process modelers typically only use a fraction of an elaborate language like BPMN. This illustrates the disconnect between BPM standardization efforts and the real needs of BPM professionals.

The model shown in Figure 1 could have been made by hand or discovered using process mining [8]. As a matter of fact, models can have very different origins. Moreover, models may also serve very different purposes. The model in Figure 1 can be used to configure a BPM system. After configuration, new cases are handled according to the rules specified in the model. However, the model can also be used for analysis (without aiming at system support), e.g., after adding timing information and frequencies it can be used for "what-if" analysis using simulation. Sometimes, process models are merely used for discussion or training.

Figure 2 provides a high-level view on four key BPM-related activities: *model*, *enact*, *analyze*, and *manage*. Process models obtained through modeling can be used for enactment (e.g., execution using a BPM or WFM system) and analysis (e.g., what-if analysis using simulation). BPM is a continuous effort, i.e., processes need to be managed and BPM does not stop after completing the process design or system implementation. Changing circumstances may trigger process adaptations and generate new analysis questions. Therefore, it is important to continuously monitor processes (e.g., using process mining).

This paper aims to survey the maturing BPM discipline. Section 2 provides a historic overview of BPM. Section 3 further structures the BPM discipline. For example, processes are classified using BPM-relevant properties. Section 4 lists various *BPM use cases*. These use cases refer to the creation of process models and their usage to improve, enact and manage processes. Section 5 discusses six *key BPM concerns* in more detail: process modeling languages, process enactment infrastructures, process model

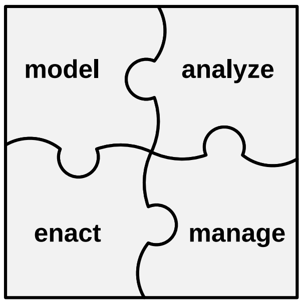

Fig. 2. A high-level view on BPM showing the four key activities: *model* (creating a process model to be used for analysis or enactment), *enact* (using a process model to control and support concrete cases), *analyze* (analyzing a process using a process model and/or event logs), and *manage* (all other activities, e.g., adjusting the process, reallocating resources, or managing large collections of related process models).

analysis, process mining, process flexibility, and process reuse. Section 6 concludes the paper with an outlook on the future of BPM.

2 History of BPM

Business Process Management (BPM) has various roots in both computer science and management science. Therefore, it is difficult to pinpoint the starting point of BPM. Since the industrial revolution, productivity has been increasing because of technical innovations, improvements in the organization of work, and the use of information technology. Adam Smith (1723-1790) showed the advantages of the division of labor. Frederick Taylor (1856-1915) introduced the initial principles of scientific management. Henry Ford (1863-1947) introduced the production line for the mass production of “black T-Fords”. It is easy to see that these ideas are used in today’s BPM systems.

Around 1950 computers and digital communication infrastructures started to influence business processes. This resulted in dramatic changes in the organization of work and enabled new ways of doing business. Today, innovations in computing and communication are still the main drivers behind change in almost all business processes. Business processes have become more complex, heavily rely on information systems, and may span multiple organizations. *Therefore, process modeling has become of the utmost importance.* Process models assist in managing complexity by providing insight and by documenting procedures. Information systems need to be configured and driven by precise instructions. Cross-organizational processes can only function properly if there is common agreement on the required interactions. As a result, process models are widely used in todays organizations.

In the last century many process modeling techniques have been proposed. In fact, the well-known Turing machine described by Alan Turing (1912-1954) can be viewed as a process model. It was instrumental in showing that many questions in computer science are undecidable. Moreover, it added a data component (the tape) to earlier transition systems. Petri nets play an even more prominent role in BPM as they are graphical and able to model concurrency. In fact, most of the contemporary BPM notations and systems use token-based semantics adopted from Petri nets. Petri nets were proposed by Carl Adam Petri (1926-2010) in 1962. This was the first formalism treating concurrency as a first-class citizen. Concurrency is very important as in business processes many things may happen in parallel. Many cases may be handled at the same time and even within a case there may be various enabled or concurrently running activities. Therefore, a BPM system should support concurrency natively.

Since the seventies there has been consensus on the modeling of data (cf. the Relational Model by Codd [11] and the Entity-Relationship Model by Chen [12]). Although there are different languages and different types of Database Management (DBM) systems, there has been consensus on the fundamental concepts for the information-centric view of information systems for decades. The process-centric view on information systems on the other hand can be characterized by the term "divergence". There is little consensus on its fundamental concepts. Despite the availability of established formal languages (e.g., Petri nets and process calculi) industry has been pushing ad-hoc/domain-specific languages. As a result there is a plethora of systems and languages available today (BPMN, BPEL, UML, EPCs, etc.), some of which will be discussed in Section 5.1.

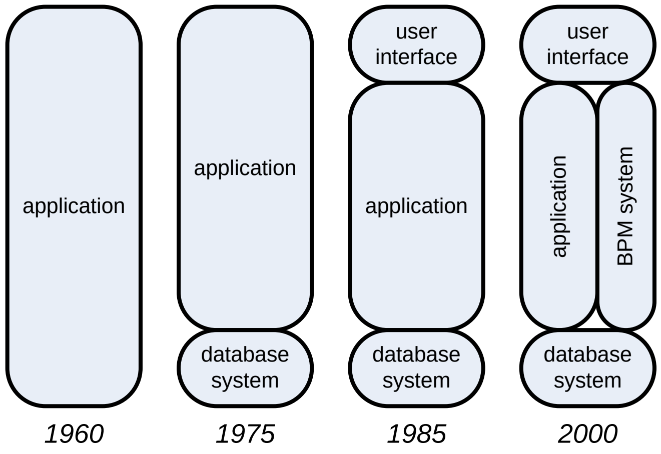

Fig. 3. Historic view on information systems development illustrating that BPM systems can be used to push "process logic" out of the application (adapted from [13])).

Figure 3 sketches the emergence of BPM systems and their role in the overall information system architecture. Initially, information systems were developed from scratch, i.e., everything had to be programmed, even storing and retrieving data. Soon people realized that many information systems had similar requirements with respect to data management. Therefore, this generic functionality was subcontracted to a DBM system. Later, generic functionality related to user interaction (forms, buttons, graphs, etc.) was subcontracted to tools that can automatically generate user interfaces. The trend to subcontract recurring functionality to generic tools continued in different areas. BPM systems can be seen in this context: a BPM system takes care of process-related aspects. Therefore, the application can focus on supporting individual/specific tasks. In the mid 1990s many WFM systems became available. These systems focused on automating workflows with little support for process analysis, process flexibility, and process management. BPM systems provide much broader support, e.g., by supporting simulation, business process intelligence, case management, etc. However, compared to the database market, the BPM market is much more diverse and there is no consensus on notations and core capabilities. This is not a surprise as process management is much more challenging than data management.

A good starting point for exploring the scientific origins of BPM is the early work on *office information systems*. In the seventies, people like Skip Ellis, Anatol Holt, and Michael Zisman already worked on so-called office information systems, which were driven by explicit process models [1, 14–22]. Ellis et al. [14, 15, 23, 16] developed office automation prototypes such as Officetalk-Zero and Officetalk-D at Xerox PARC in the late 1970s. These systems used Information Control Nets (ICN), a variant of Petri nets, to model processes. Office metaphors such as inbox, outbox, and forms were used to interact with users. The prototype office automation system SCOOP (System for Computerizing of Office Processes) developed by Michael Zisman also used Petri nets to represent business processes [20, 22, 21]. It is interesting to see that pioneers in office information systems already used Petri-net-based languages to model office procedures. During the seventies and eighties there was great optimism about the applicability of office information systems. Unfortunately, few applications succeeded. As a result of these experiences, both the application of this technology and research almost stopped for a decade. Consequently, hardly any advances were made in the eighties. In the nineties, there was a clear revival of the ideas already present in the early office automation prototypes [4]. This is illustrated by the many commercial WFM systems developed in this period.

In the mid-nineties there was the expectation that WFM systems would get a role comparable to Database Management (DBM) systems. Most information systems subcontract their data management to DBM systems and comparatively there are just a few products. However, these products are widely used. Despite the availability of WFM/BPM systems, process management is not subcontracted to such systems at a scale comparable to DBM systems. The application of "pure" WFM/BPM systems is still limited to specific industries such as banking and insurance. However, WFM/BPM technology is often hidden inside other systems. For example, ERP systems like SAP and Oracle provide workflow engines. Many other platforms include workflow-like functionality. For example, integration and application infrastructure software such as IBM's WebSphere and Cordys's Business Operations Platform (BOP) provides extensive process support. In hindsight, it is easy to see why process management cannot be subcontracted to a standard WFM/BPM system at a scale comparable to DBM systems. As illustrated by the varying support for the workflow patterns [9, 24, 25], process management is much more “thorny” than data management. *BPM is multifaceted, complex, and difficult to demarcate.* Given the variety in requirements and close connection to business concerns, it is often impossible to use generic BPM/WFM solutions. Therefore, BPM functionality is often embedded in other systems. Moreover, BPM techniques are frequently used in a context with conventional information systems.

BPM has become a mature discipline. Its relevance is acknowledged by practitioners (users, managers, analysts, consultants, and software developers) and academics. This is illustrated by the availability of many BPM systems and a range of BPM-related conferences.

In this survey we will often refer to results presented at the annual international BPM conference. The international BPM conference is celebrating its 10th anniversary and its proceedings provide a good overview of the state-of-the-art: BPM 2003 (Eindhoven, The Netherlands) [26], BPM 2004 (Potsdam, Germany) [27], BPM 2005 (Nancy, France) [28], BPM 2006 (Vienna, Austria) [29], BPM 2007 (Brisbane, Australia) [30], BPM 2008 (Milan, Italy) [31], BPM 2009 (Ulm, Germany) [32], BPM 2010 (Hoboken, USA) [33], BPM 2011 (Clermont-Ferrand, France) [34], and BPM 2012 (Tallinn, Estonia) [35]. Other sources of information are the following books on WFM/BPM: [5] (first comprehensive WFM book focusing on the different workflow perspectives and the MOBILE language), [36] (edited book that served as the basis for the BPM conference series), [4] (most cited WFM book; a Petri net-based approach is used to model, analyze and enact workflow processes), [19] (book relating WFM systems to operational performance), [7] (edited book on process-aware information systems), [6] (book on production WFM systems closely related to IBM’s workflow products), [37] (visionary book linking management perspectives to the pi calculus), [2] (book presenting the foundations of BPM, including different languages and architectures), [38] (book based on YAWL and the workflow patterns), [8] (book focusing on process mining and BPM), and [39] (book on supporting flexibility in process-aware information systems). Most of these books also provide a historical perspective on the BPM discipline.

3 Structuring the BPM Discipline

Before discussing typical BPM use cases and some of the key concerns of BPM, we first structure the domain by describing the BPM life-cycle and various classifications of processes.

Figure 4 shows the *BPM life-cycle*. In the *(re)design phase*, a process model is designed. This model is transformed into a running system in the *implementation/configuration 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 hardcoded in conventional software, this phase may take substantial time. After the system supports the designed processes, the *run & adjust phase* starts. In this phase, the processes are enacted and adjusted when needed. In the run & adjust phase, the process is not redesigned and no new software is created; only predefined controls are used to adapt or reconfigure the process. Figure 4 shows two types of analysis: *model-based analysis* and *data-based analysis*. While the system is running, event data are collected. These data can be used to analyze running processes, e.g., discover bottlenecks, waste, and deviations. This is input for the redesign phase. During this phase process models can be used for analysis. For example, simulation is used for what-if analysis or the correctness of a new design is verified using model checking.

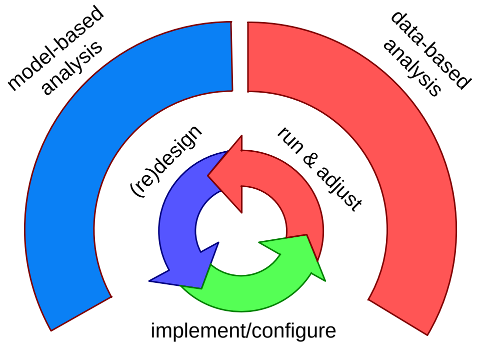

Fig. 4. The BPM life-cycle consisting of three phases: (1) *(re)design*, (2) *implement/configure* and (3) *run & adjust*.

The scope of BPM extends far beyond the implementation of business processes. Therefore, the role of model-based and data-based analyses is emphasized in Figure 4.

Business processes can be classified into human-centric and system-centric [40] or more precisely into *Person-to-Person* (P2P), *Person-to-Application* (P2A) and *Application-to-Application* (A2A) processes [7].

In P2P processes, the participants involved are primarily people, i.e. the processes predominantly involve activities that require human intervention. Job tracking, project management, and groupware tools are designed to support P2P processes. Indeed, the processes supported by these tools are not composed of fully-automated activities only. In fact, the software tools used in these processes (e.g. project tracking servers, e-mail clients, video-conferencing tools, etc.) are primarily oriented towards supporting computer-mediated interactions. Recently, the importance of social networks increased significantly (facebook, twitter, linkedin, etc.) and BPM systems need to be able to incorporate such computer-mediated human interactions [41]. The term “Social BPM” refers to exploiting such networks for process improvement.

On the other end of the spectrum, A2A processes are those that only involve activities performed by software systems. Financial systems may exchange messages and money without any human involvement, logistic information systems may automatically order products when inventory falls below a predefined threshold, Transaction processing systems, EAI platforms, and Web-based integration servers are examples of technologies to support A2A processes.

P2A processes involve both human activities and interactions between people, and activities and interactions involving applications which act without human intervention.

Most BPM/WFM systems fall in the P2A category. In fact, most information systems aim at making people and applications work in an integrated manner.

Note that the boundaries between P2P, P2A, and A2A are not crisp. Instead, there is a continuum of processes, techniques, and tools covering the spectrum from P2P (i.e. manual, human-driven) to A2A (automated, application-driven).

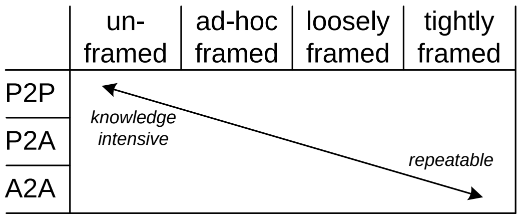

Fig. 5. Classification of processes: most processes can be found around the diagonal.

Orthogonal to the classification of processes into P2P, P2A, and A2A, we distinguish between *unframed*, *ad hoc framed*, *loosely framed*, and *tightly framed* processes [7] (cf. Figure 5).

A process is said to be *unframed* if there is no explicit process model associated with it. This is the case for collaborative processes supported by groupware systems that do *not* offer the possibility of defining process models.

A process is said to be *ad hoc framed* if a process model is defined a priori but only executed once or a small number of times before being discarded or changed. This is the case in project management environments where a process model (i.e. a project chart) is often only executed once. It is also the case in scientific computing environments, where a scientist may define a process model corresponding to a computation executed on a grid and involving multiple datasets and computing resources [42]. Such a process is often executed only once (although parts may be reused for other experiments).

A *loosely framed* process is one for which there is an a-priori defined process model and a set of constraints, such that the predefined model describes the "normal way of doing things" while allowing the actual executions of the process to deviate from this model (within certain limits). Case handling systems aim to support such processes, that is, they support the ideal process and implicitly defined deviations (e.g., skipping activities or rolling back to an earlier point in the process).

Finally, a *tightly framed* process is one which consistently follows an a-priori defined process model. Tightly framed processes are best supported by traditional WFM systems.

As Figure 5 shows, the degree of framing of the underlying processes (unframed, ad hoc, loosely, or tightly framed), and the nature of the process participants (P2P, P2A, and A2A) are correlated. Most processes are found around the diagonal. Knowledge-intensive processes tend to be less framed and more people-centric. Highly repeatable processes tend to be tightly framed and automated.

As with P2P, P2A, and A2A processes, the boundaries between unframed, ad hoc framed, loosely framed, and tightly framed processes are not crisp. In particular, there is a continuum between loosely and tightly framed processes. For instance, during its operational life a process considered to be tightly framed can start deviating from its model so often and so unpredictably, that at some point in time it may be considered to have become loosely framed. Conversely, when many instances of a loosely framed process have been executed, a common structure may become apparent, which may then be used to frame the process in a tighter manner.

Figures 4 and 5 illustrate the breadth of the BPM spectrum. A wide variety of processes – ranging from unframed and people-centric to tightly framed and fully automated – may be supported using BPM technology. Different types of support are needed in three main phases of the BPM life-cycle (cf. Figure 4). Moreover, various types of analysis can be used in these phases: some are based on models only whereas others also exploit event data. In the remainder we present a set of twenty *BPM use cases* followed by a more detailed discussion of six *key concerns*. The use cases and key concerns are used to provide a survey of the state-of-the-art in BPM research. Moreover, the proceedings of past BPM conferences are analyzed to see trends in the maturing BPM discipline.

4 BPM Use Cases

To further structure the BPM discipline and to show "how, where, and when" BPM techniques can be used, we provide a set of twenty BPM use cases. Figures 6-13 show graphical representations of these use cases. Models are depicted as pentagons marked with the letter "*M*". A model may be descriptive (*D*), normative (*N*), and/or executable (*E*). A "$D|N|E$" tag inside a pentagon means that the corresponding model is descriptive, normative, or executable. Tag "*E*" means that the model is executable. Configurable models are depicted as pentagons marked with "*CM*". Event data (e.g., an event log) are denoted by a disk symbol (cylinder shape) marked with the letter "*E*". Information systems used to support processes at runtime are depicted as squares with rounded corners and marked with the letter "*S*". Diagnostic information is denoted by a star shape marked with the letter "*D*". We distinguish between conformance-related diagnostics (star shape marked with "*CD*") and performance-related diagnostics (star shape marked with "*PD*").

The twenty atomic use cases can be chained together in so-called *composite* use cases. These composite cases correspond to realistic BPM scenarios.

4.1 Use Cases to Obtain Models

The first category of use cases we describe have in common that a process model is produced (cf. Figures 6 and 7).

Design Model (DesM) Use case *design model* (DesM) refers to the creation of a process model from scratch by a human. Figure 6 shows the creation of a model represented by a pentagon marked with the letter "*M*". This is still the most common way to

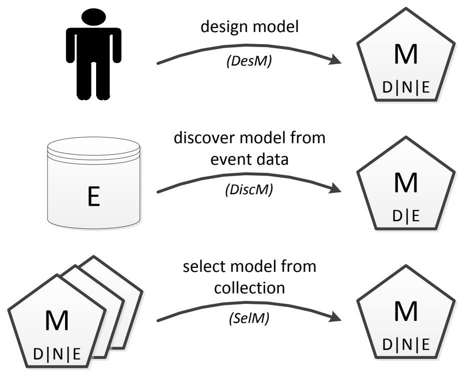

Fig. 6. Use cases to obtain a descriptive, normative, or executable process model.

create models. The hand-made model may be descriptive, normative, or executable. Descriptive models are made to describe the as-is or to-be situation. A descriptive model may describe undesirable behavior. If the model only describes the desired behavior, is is called normative. A normative model may describe a rule like "activities x and y should never be executed by the same person for a given case" even though in reality the rule is often violated and not enforced. An executable model can be interpreted unambiguously by software, e.g., to enact or verify a process. Given a state or sequence of past activities, the model can determine the set of possible next activities. A model may be executable and descriptive or normative, i.e., the three classes are not mutually exclusive and combinations are possible.

Discover Model from Event Data (DiscM) The term "Big Data" is often used to refer to the incredible growth of event data in recent years [43]. More and more organizations realize that the analysis of event data is crucial for process improvement and achieving competitive advantage over competitors. Use case *discover model from event data* (DiscM) refers to the automated generation of a process model using process mining techniques [8].

The goal of process mining is to extract knowledge about a particular (operational) process from event logs, i.e., process mining describes a family of *a-posteriori* analysis techniques exploiting the information recorded in audit trails, transaction logs, databases, etc. (cf. Section 5.4). Typically, these approaches 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). Furthermore, some mining techniques use additional information such as the performer or *originator* of the event (i.e., the person / resource executing or initiating the activity), the *timestamp* of the event, or *data elements* recorded with the event (e.g., the size of an order).

A discovery technique takes an event log and produces a model without using any a-priori information. An example is the α-algorithm [44]. 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 Figure 1, the α-algorithm is able to automatically construct the corresponding 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.

Select Model From Collection (SelM) Large organizations may have repositories containing hundreds of process models. There may be variations of the same model for different departments or products. Moreover, processes may change over time resulting in different versions. Because of these complexities, (fragments of) process models may be reinvented without reusing existing models. As a result, even more process models need to coexist, thus further complicating model management. Therefore, reuse is one of the key concerns in BPM (cf. Section 5.6).

Use case *select model from collection* (SelM) refers to the retrieval of existing process models, e.g., based on keywords or process structures. An example of a query is "return all models where activity *send invoice* can be followed by activity *reimburse*". Another example is the query "return all models containing activities that need to be executed by someone with the role *manager*".

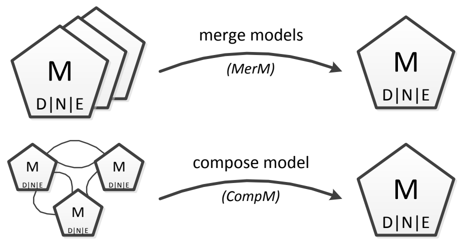

Fig. 7. Use cases to obtain a process model from other models.

Merge Models (MerM) Use case SelM selects a complete model from some repository. However, often new models are created from existing models. Use case *merge models* (MerM) refers to the scenario where different parts of different models are merged into one model. For example, the initial part of one model is composed with the final part of another process, a process model is extended with parts taken from another model, or different process models are unified resulting in a new model. Unlike classical composition the original parts may be indistinguishable.

Compose Model (CompM) Use case *compose model* (CompM) refers to the situation where different models are combined into a larger model. Unlike use case MerM the different parts can be related to the original models used in the composition.

The five use cases shown in Figures 6 and 7 all produce a model. The resulting model may be used for analysis or enactment as will be shown in later use cases.

4.2 Use Cases Involving Configurable Models

A configurable process model represents a *family of process models*, that is, a model that through configuration can be customized for a particular setting. For example, configuration may be achieved by *hiding* (i.e., bypassing) or *blocking* (i.e., inhibiting) certain fragments of the configurable process model [45]. In this way, the desired behavior is selected. From the viewpoint of generic BPM software, configurable process models can be seen as a mechanism to add "content" to these systems. By developing comprehensive collections of configurable models, particular domains can be supported. From the viewpoint of ERP software, configurable process models can be seen as a means to make these systems more process-centric, although in the latter case quite some refactoring is needed as processes are often hidden in table structures and application code. Various configurable languages have been proposed as extensions of existing languages (e.g., C-EPCs [46], C-SAP, C-BPEL) but few are actually supported by enactment software (e.g., C-YAWL [47]). Traditional reference models [48–50] can be seen as configurable process models. However, configuration is often implicit or ad-hoc and often such reference models are not executable.

Figure 8 shows three use cases related to configurable process models.

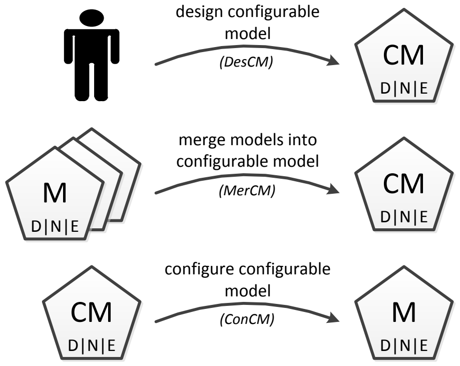
design configurable model
(DesCM)
CM
D|N|E
M
D|N|E
merge models into configurable model
(MerCM)
CM
D|N|E
CM
D|N|E
configure configurable model
(ConCM)
M
D|N|E

Fig. 8. Use cases to obtain and use configurable process models.

Design Configurable Model (DesCM) Configurable process models can be created from scratch as shown by use case *design configurable model* (DesCM). Creating a configurable model is more involved than creating an ordinary non-configurable model. For example, because of hiding and/or blocking selected fragments, the instances of a configured model may suffer from behavioral anomalies such as deadlocks and livelocks. This problem is exacerbated by the many possible configurations a model may have, and by the complex domain dependencies which may exist between various configuration options [51].

Merge Models Into Configurable Model (MerCM) A configurable process model represents a family of process models. A common approach to obtain a configurable model is to merge example members of such family into a model that is able to generate a least the example variants. The merging of model variants into a configurable model is analogous to the discovery of process models from example traces.

Figure 9 illustrates the use case *merge models into configurable model* (MerCM). Two variants of the same process are shown in the top-left corner. Variant 1 models a process where activity a is followed by activity b. After completing b, activities d and f can be executed in any order, followed by activity g. Finally, h is executed. Variant 2 also starts with activity a. However, now a is followed by activities d and f or d and e. Moreover, after completing g the process can loop back to a state where again there is a choice between d and f or d and e.

The two variants can be merged into the configurable model shown in the center of Figure 9. Activities c and e can be blocked and activity b can be hidden. If we block c and e and do not hide b (i.e., b is activated), we obtain the first variant. If we do not block c and e and hide b, we obtain the second variant.

Configure Configurable Model (ConCM) Figure 9 also illustrates use case *configure configurable model* (ConCM). This use case creates a concrete model from some configurable process model by selecting a concrete variant, i.e., from a family of process variants one member is selected. The bottom part of Figure 9 shows a variant created by blocking activities c and e and hiding activity b.

Figure 9 is a bit misleading as it only shows the control-flow. Data-related aspects and domain modeling play an important role in process configuration. For example, when configuring ERP systems like SAP R/3 the data-perspective is most prominent.

4.3 Use Cases Related to Process Execution

BPM systems are used to enact processes based on executable process models. In fact, the initial focus of WFM systems was on process automation and implementation, and not on the management, analysis and improvement of business processes (cf. Figure 10).

Refine Model (RefM) Only executable models can be enacted. Therefore, use case *refine model* (RefM) describes the scenario of converting a model tagged with "$D|N$" into a model tagged with "*E*", i.e., a descriptive or normative model is refined into a model that is also executable. To make a model executable one needs to remove all ambiguities, i.e., the supporting software should understand its meaning. Moreover, it may be necessary to detail aspects not considered relevant before. For example, it may be necessary to design a form where the user can enter data.

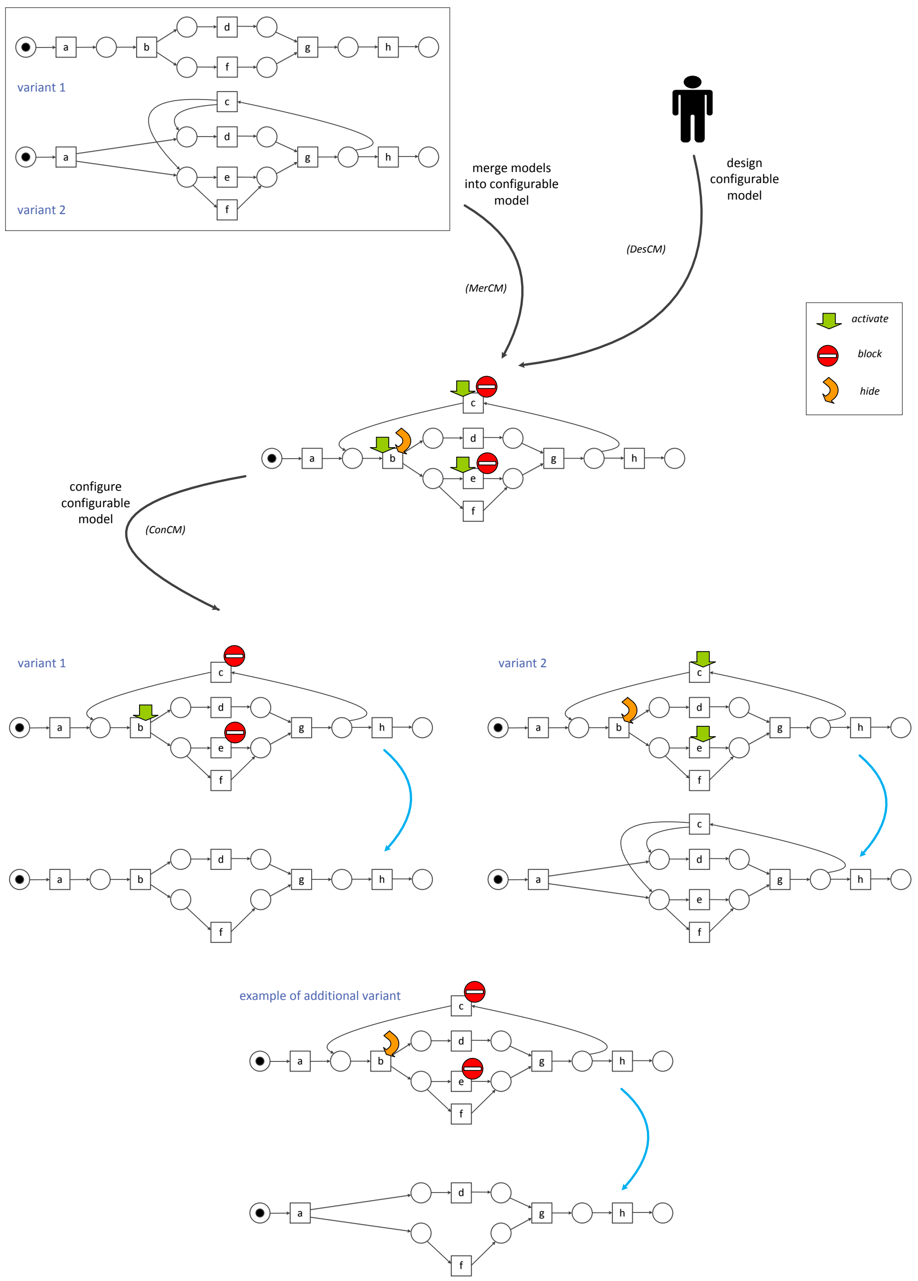
variant 1
variant 2
merge models into configurable model
(MerCM)
design configurable model
(DesCM)
activate
block
hide
configure configurable model
(ConCM)
variant 1
variant 2
example of additional variant

Fig. 9. Abstract example illustrating the use cases related to configurable process models. A configurable model may be created from scratch (use case DesCM) or from existing process models (use case MerCM). The resulting configurable model can be used to generate concrete models by hiding and blocking parts (use case ConCM).

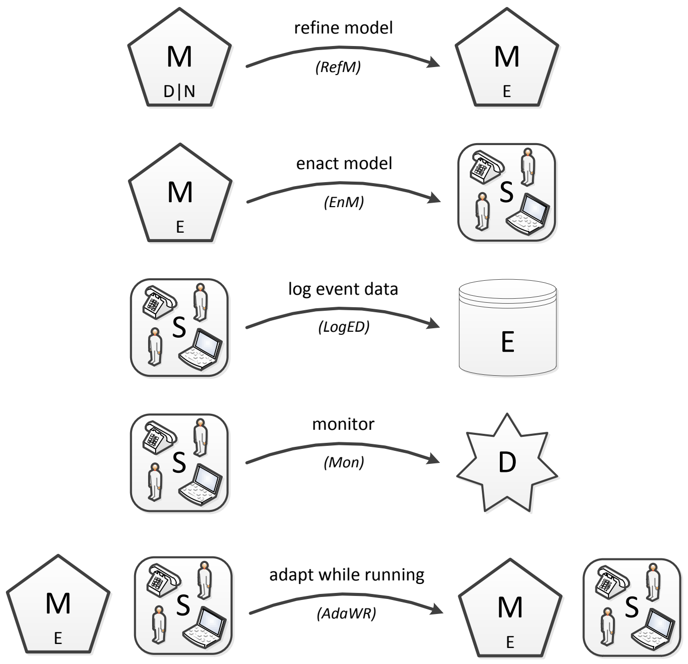

Fig. 10. Use cases to enact models, to log event data, and to monitor running processes.

Enact Model (EnM) Executable models can be interpreted by BPM systems and used to support the execution of concrete cases. Use case *enact model* (EnM) takes as input a model and as output a running system. The running system should be reliable, usable, and have a good performance. Therefore, issues like exception handling, scalability, and ergonomics play an important role. These factors are typically not modeled when discussing or analyzing processes. Yet they are vital for the actual success of the system. Therefore, Section 5.2 discusses the process enactment infrastructure as one of the key concerns of BPM.

Log Event Data (LogED) When process instances (i.e., cases) are handled by the information system, they leave traces in audit trails, transaction logs, databases, etc. Even when no BPM/WFM systems is used, relevant events are often recorded by the supporting information system. Use case *log event data* (LogED) refers to the recording of event data, often referred to as *event logs*. Such event logs are used as input for various process mining techniques. Section 5.4 discusses the use of event data as one of the key concerns of BPM.

Monitor (Mon) Whereas process mining techniques center around event data and models (e.g., models are discovered or enriched based on event logs), monitoring techniques simply measure without building or using a process model. For example, it is possible to measure response times without using or deriving a model. Modern BPM systems show dashboards containing information about Key Performance Indicators (KPIs) related to costs, responsiveness, and quality. Use case *monitor* (Mon) refers to all measurements done at runtime without actively creating or using a model.

Adapt While Running (AdaWR) BPM is all about making choices. When designing a process model choices are made with respect to the ordering of activities. At runtime, choices may be resolved by human decision making. Also process configuration is about selecting the desired behavior from a family of process variants. As will be explained in Section 5.5, flexibility can be viewed as the ability to make choices at different points in time (design-time, configuration-time, or runtime). Some types of flexibility require changes of the model at runtime. Use case *adapt while running* (AdaWR) refers to the situation where the model is adapted at runtime. The adapted model may be used by selected cases (ad-hoc change) or by all new cases (evolutionary change). Adapting the system or process model at runtime may introduce all kinds of complications. For example, by making a concurrent process more sequential, deadlocks may be introduced for already running cases.

4.4 Use Cases Involving Model-Based Analysis

Process models are predominantly used for discussion, configuration, and implementation. Interestingly, process models can also be used for analysis. This is in fact one of the key features of BPM. Instead of directly hard-coding behavior in software, models can be analyzed before being put into production.

Analyze Performance Based on Model (PerfM) Executable process models can be used to analyze the expected performance in terms of response times, waiting times, flow times, utilization, costs, etc. Use case *analyze performance based on model* (PerfM) refers to such analyses. Simulation is the most widely applied analysis technique in BPM because of its flexibility. Most BPM tools provide a simulation facility. Analytical techniques using for example queueing networks or Markov chains can also be used to compute the expected performance. However, these are rarely used in practice due to the additional assumptions needed.

Verify Model (VerM) Before a process model is put into production, one would like to get assurance that the model is correct. Consider for example the notion of soundness [13, 52]. A process model is sound if cases cannot get stuck before reaching the end (termination is always possible) and all parts of the process can be activated (no dead segments). Use case *verify model* (VerM) refers to the analysis of such properties using techniques such as model checking.

Section 5.3 elaborates on model-based analysis as one of the key concerns of BPM.

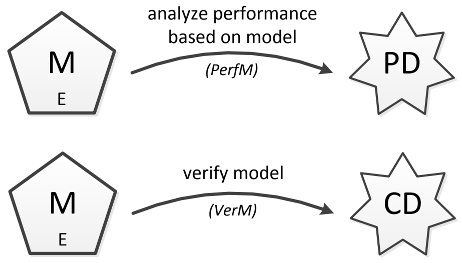

Fig. 11. Use cases for model-based analysis.

4.5 Use Cases Extracting Diagnostics From Event Data

A process model may serve as a pair of glasses that can be used to look at reality. As Figure 12 shows, we identify two use cases where diagnostic information is derived from both model and event data.

Check Conformance Using Event Data (ConfED) Event data and models can be compared to see where modeled and observed behavior deviate. For example, one may replay history on a process model and see where observed events do not "fit" the model. Use case *check conformance using event data* (ConfED) refers to all kinds of analysis aiming at uncovering discrepancies between modeled and observed behavior. Conformance checking may be done for auditing purposes, e.g., to uncover fraud or malpractices.

Analyze Performance Using Event Data (PerfED) Event data often contain timing information, i.e., events have timestamps that can be used for performance analysis. Use case *analyze performance using event data* (PerfED) refers to the combined use of models and timed event data. By replaying an event log with timestamps on a model, one can measure delays, e.g., the time in-between two subsequent activities. The result can be used to highlight bottlenecks and gather information for simulation or prediction techniques.

4.6 Use Cases Producing New Models based on Diagnostics or Event Data

Diagnostic information and event data can be used to repair, extend, or improve models (cf. Figure 13).

Repair Model (RepM) Use case ConfED can be used to see where reality and model deviate. The corresponding diagnostics can be used as input for use case *repair model* (RepM), i.e., the model is adapted to match reality better [53]. On the one hand, the resulting model should correspond to the observed behavior. On the other hand, the repaired model should be as close to the original model as possible. The challenge is to balance both concerns.

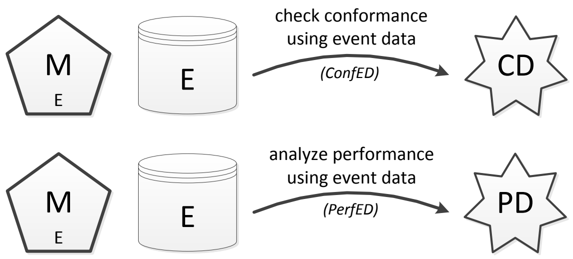
M
E
E
check conformance using event data
(ConfED)
CD
M
E
E
analyze performance using event data
(PerfED)
PD

Fig. 12. Use cases were diagnostics are obtained using both model and log.

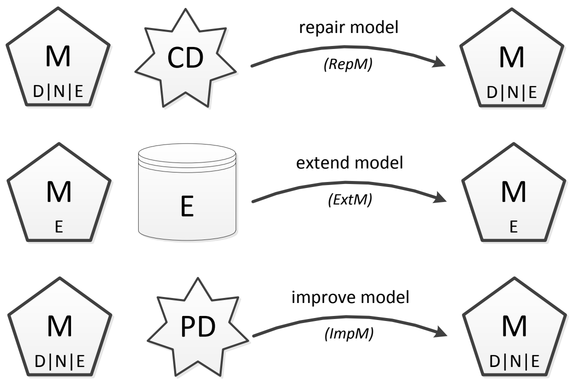

Fig. 13. Use cases to repair, extend, or improve process models.

Extend Model (ExtM) Event logs refer to activities being executed and events may be annotated with additional information such as the person/resource executing or initiating the activity, the timestamp of the event, or data elements recorded with the event. Use case *extend model* (ExtM) refers to the use of such additional information to enrich the process model. For example, timestamps of events may be used to add delay distributions to the model. Data elements may be used to infer decision rules that can be added to the model. Resource information can be used to attach roles to activities in the model. This way it is possible to extend a control-flow oriented model with additional perspectives.

Improve Model (ImpM) Performance related diagnostics obtained through use case PerfED can be used to generate alternative process designs aiming at process improvements, e.g., to reduce costs or response times. Use case *improve model* (ImpM) refers to BPM functionality helping organizations to improve processes by suggesting alternative process models. These models can be used to do "what-if" analysis. Note that unlike RepM the focus ImpM is on improving the process itself.

4.7 Composite Use Cases

The twenty atomic use cases should not be considered in isolation, i.e., for practical BPM scenarios these atomic use cases are chained together into composite use cases. Figure 14 shows three examples.

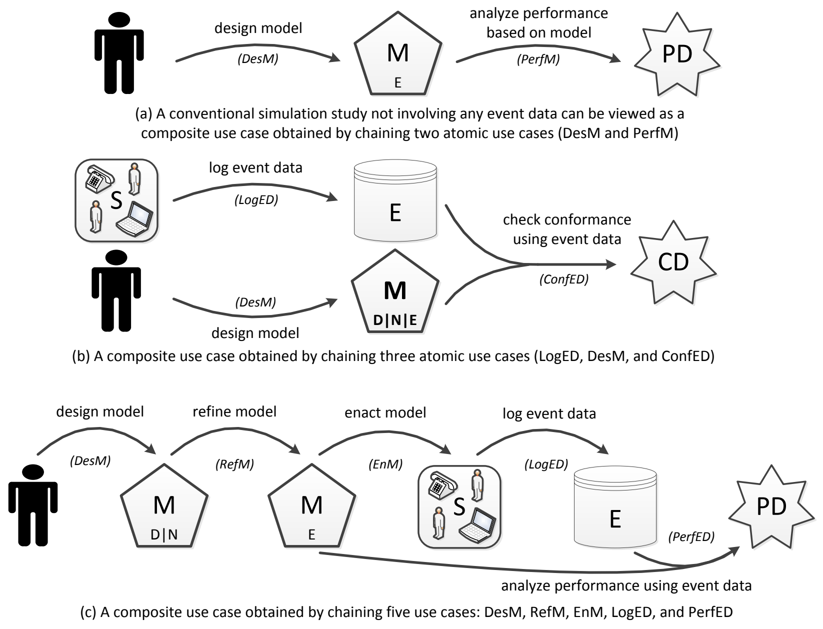
design model
(DesM)
M
E
analyze performance based on model
(PerfM)
PD
(a) A conventional simulation study not involving any event data can be viewed as a composite use case obtained by chaining two atomic use cases (DesM and PerfM)
S
log event data
(LogED)
E
check conformance using event data
(ConfED)
CD
(DesM)
design model
M
D|N|E
(b) A composite use case obtained by chaining three atomic use cases (LogED, DesM, and ConfED)
design model
(DesM)
refine model
(RefM)
enact model
(EnM)
log event data
(LogED)
M
D|N
M
E
S
E
(PerfED)
PD
analyze performance using event data
(c) A composite use case obtained by chaining five use cases: DesM, RefM, EnM, LogED, and PerfED

Fig. 14. Three composite use case obtained by chaining atomic use cases.

The first example (Figure 14(a)) is the classical scenario where a model is constructed manually and subsequently used for performance analysis. Note that the use cases *design model* (DesM) and *analyze performance based on model* (PerfM) are chained together. A conventional simulation not involving event data would fit this composite use case.

The second composite use case in Figure 14 combines three atomic use cases: the observed behavior extracted from some information system (LogED) is compared with a manually designed model (DesM) in order to find discrepancies (ConfED).

Figure 14(c) shows a composite use case composed of five atomic use cases. The initially designed model (DesM) is refined to make it executable (RefM). The model is used for enactment (EnM) and the resulting behavior is logged (LogED). The modeled behavior and event data are used to reveal bottlenecks (PerfED), i.e., performance related information is extracted from the event log and projected onto the model.

The composite use cases in Figure 14 are merely examples, i.e., a wide range of BPM scenarios can be supported by composing the twenty atomic use cases.

Table 1. Relative importance of use cases in [36], [26], [27], [28], [29], [30], [31], [32], [33], and [34]. Each of the 289 papers was tagged with, on average, 1.18 use cases (typically 1 or 2 use cases per paper). The table shows the relative frequency of each use case per year. The last row shows the average over 10 years. All rows add up to 1.

	design model	discover model from event data	select model from collection	merge models	compose model	design configurable model	merge models into configurable model	configure configurable model	refine model	enact model	log event data	monitor	adapt while running	analyze performance based on model	verify model	check conformance using event data	analyze performance using event data	repair model	extend model	improve model
year	DesM	DiscM	SelM	MerM	CompM	DesCM	MerCM	ConCM	RefM	EnM	LogED	Mon	AdaWR	PerfM	VerM	ConfED	PerfED	RepM	ExtM	ImpM
2000	0.406	0.000	0.000	0.000	0.031	0.000	0.000	0.000	0.000	0.188	0.000	0.000	0.063	0.125	0.188	0.000	0.000	0.000	0.000	0.000
2003	0.306	0.028	0.056	0.000	0.056	0.000	0.000	0.028	0.000	0.222	0.028	0.000	0.139	0.000	0.111	0.000	0.000	0.000	0.028	0.000
2004	0.348	0.130	0.000	0.000	0.043	0.000	0.000	0.000	0.000	0.217	0.000	0.000	0.043	0.087	0.087	0.000	0.000	0.000	0.000	0.043
2005	0.216	0.039	0.039	0.000	0.098	0.000	0.000	0.000	0.000	0.294	0.000	0.000	0.059	0.078	0.137	0.000	0.000	0.000	0.000	0.039
2006	0.094	0.038	0.057	0.019	0.132	0.000	0.000	0.000	0.019	0.245	0.019	0.000	0.075	0.019	0.226	0.019	0.019	0.000	0.000	0.019
2007	0.231	0.128	0.026	0.026	0.051	0.026	0.026	0.077	0.077	0.077	0.026	0.026	0.026	0.051	0.103	0.000	0.026	0.000	0.000	0.000
2008	0.227	0.045	0.023	0.000	0.045	0.000	0.023	0.000	0.023	0.182	0.045	0.000	0.023	0.091	0.136	0.000	0.045	0.023	0.068	0.000
2009	0.167	0.133	0.067	0.000	0.033	0.000	0.033	0.000	0.133	0.133	0.033	0.000	0.000	0.033	0.167	0.033	0.033	0.000	0.000	0.000
2010	0.167	0.133	0.100	0.000	0.000	0.033	0.000	0.033	0.033	0.300	0.000	0.000	0.000	0.000	0.100	0.067	0.000	0.000	0.033	0.000
2011	0.061	0.091	0.121	0.030	0.000	0.000	0.061	0.000	0.000	0.121	0.000	0.061	0.000	0.000	0.182	0.152	0.030	0.061	0.030	0.000
average	0.222	0.077	0.049	0.007	0.049	0.006	0.014	0.014	0.029	0.198	0.015	0.009	0.043	0.048	0.144	0.027	0.015	0.008	0.016	0.010

4.8 Analysis of BPM Conference Proceedings Based on Use Cases

After describing the twenty BPM uses cases, we evaluate their relative importance in BPM literature [54]. As a reference set of papers we used all papers in the proceedings of past BPM conferences, i.e., BPM 2003 - BPM 2011 ([26], [27], [28], [29], [30], [31], [32], [33], and [34]) and the edited book "Business Process Management: Models, Techniques, and Empirical Studies" [36]. The edited book [36] appeared in 2000 and can be viewed as a predecessor of the first BPM conference.

In total 289 papers were analyzed by tagging each paper with the use cases [54]. As will be discussed in Section 5.7, we also tagged each paper with the key concerns addressed. Since the BPM conference is the premier conference in the field, these 289 papers provide a representative view on BPM research over the last decade.

Most papers were tagged with one dominant use case, but sometimes more tags were used. In total, 367 tags were assigned (on average 1.18 use cases per paper). For example, the paper "Instantaneous Soundness Checking of Industrial Business Process Models" [55] presented at BPM 2009 is a typical example of a paper tagged with use case *verify model* (VerM). In [55] 735 industrial business process models are checked for soundness (absence of deadlock and lack of synchronization) using three different approaches. The paper "Graph Matching Algorithms for Business Process Model Similarity Search" [56] presented at the same conference was tagged with the use case *select model from collection* (SelM) since the paper presents an approach to rank process models in a repository based on some input model. These examples illustrate the tagging process.

By simply counting the number of tags per use case and year, the relative frequency of each use case per year can be established. For example, for BPM 2009 four papers were tagged with use case *discover model from event data* (DiscM). The total number of tags assigned to the 23 BPM 2009 papers is 30. Hence, the relative frequency of DiscM is $4/30 = 0.133$. Table 1 shows all relative frequencies including the one just mentioned. The table also shows the average relative frequency of each use case over all ten years. These averages are shown graphically in Figure 15.

Figure 15 shows that use cases *design model* (DesM) and *enact model* (EnM) are most frequent. This is not very surprising as these use cases are less specific than most other use cases. The third most frequent use case – *verify model* (VerM) – is more surprising (relative frequency of 0.144). An example paper having such a tag is [55] which was mentioned before. Over the last decade there has been considerable progress in this area and this is reflected by various verification papers presented at BPM. In this context it is remarkable that the use cases *monitor* (Mon) and *analyze performance using event data* (PerfED) have a much lower relative frequency (respectively 0.009 and 0.015). Given the practical needs of BPM one would expect more papers presenting techniques to diagnose and improve the performance of business processes.

Figure 16 shows changes of relative frequencies over time. The graph shows a slight increase in process-mining related topics. However, no clear trends are visible due to the many use cases and small number of years and papers per year. Therefore, the 289 BPM papers were also analyzed based on the six key concerns presented next (cf. Section 5.7).

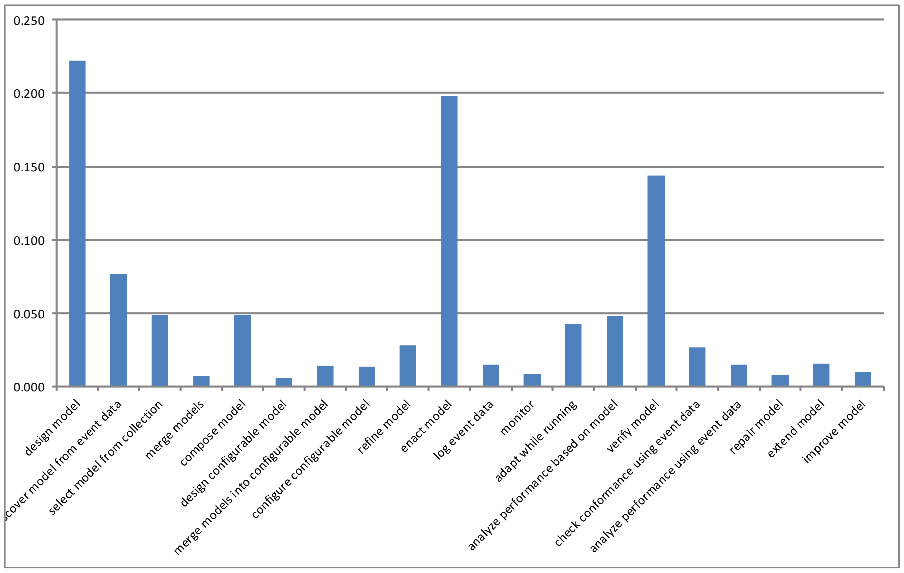

Fig. 15. Average relative importance of use cases (taken from Table 1).

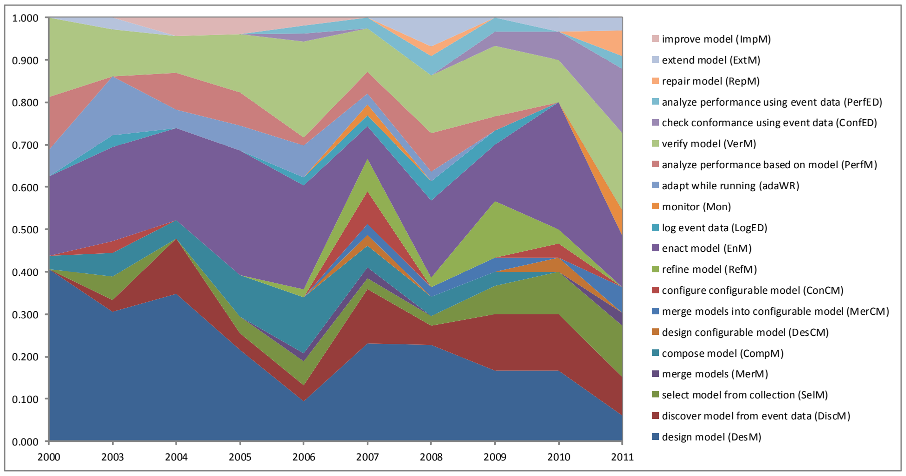

Fig. 16. Development of the relative importance of each use case plotted over time (derived from Table 1).

5 BPM Key Concerns

The use cases refer to the practical/intended use of BPM techniques and tools. However, BPM research is not equally distributed over all of these use cases. Some use cases provide important engineering or managerial challenges, but these are not BPM-specific or do not require additional BPM research. Other use cases require foundational research and are not yet encountered frequently in practice. Therefore, we now zoom in on six key concerns addressed by many BPM papers: process modeling languages, process enactment infrastructures, process model analysis, process mining, process flexibility, and process reuse.

5.1 Process Modeling Languages

The modeling and analysis of processes plays a central role in business process management. Therefore, the choice of language to represent an organization's processes is essential. Three classes of languages can be identified:

- **Formal languages**: Processes have been studied using theoretical models. Mathematicians have been using Markov chains, queueing networks, etc. to model processes. Computer scientists have been using Turing machines, transition systems, Petri nets, temporal logic and process algebras to model processes. All of these languages have in common that they have *unambiguous semantics* and allow for *analysis*.
- **Conceptual languages**: Users in practice often have problems using formal languages due to the rigorous semantics (making it impossible to leave things intentionally vague) and low-level nature. They typically prefer to use higher-level languages. Examples are BPMN (Business Process Modeling Notation, [57, 58]), EPCs (Event-Driven Process Chains, [59–61]), UML activity diagrams, etc. (see Figure 17 for some examples). These language are typically *informal*, i.e., they do not have a well-defined semantics and do not allow for analysis. Moreover, the lack of semantics makes it impossible to directly execute them.
- **Execution languages**: Formal languages typically abstract from "implementation details" (e.g., data structures, forms, and interoperability problems) and conceptual languages only provide an approximate description of the desired behavior. Therefore, more technical languages are needed for enactment. An example is the BPEL (Business Process Execution Language, [62]) language. Most vendors provide a proprietary execution language. In the latter case, the source code of the implemented tool determines the exact semantics.

Note that fragments of languages like BPMN, UML, BPEL, and EPCs have been formalized by various authors [63, 64]. However, these formalizations typically cover only selected parts of the language (e.g., abstract from data or OR-joins). Moreover, people tend to use only a small fragment of languages like BPMN [10]. To illustrate problems related to the standardization of industry-driven languages, consider the OR-join semantics described in the most recent BPMN standard [58]. Many alternative semantics have been proposed and are used by different tools and formalizations [65–68]. There is a trade-off between accuracy and performance and due to the "vicious circle" [69, 66] it is impossible to provide "clean semantics" for all cases. In fact, the OR-join semantics of [58] are not supported by any of the many tools claiming to support BPMN.

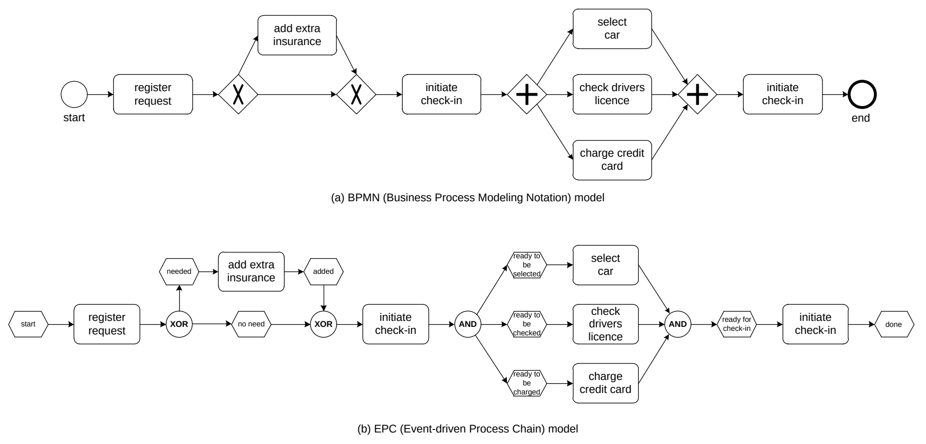

Fig. 17. Two examples of conceptual procedural languages: (a) BPMN (Business Process Modeling Notation, [58]) and (b) EPC (Event-driven Process Chain, [59]).

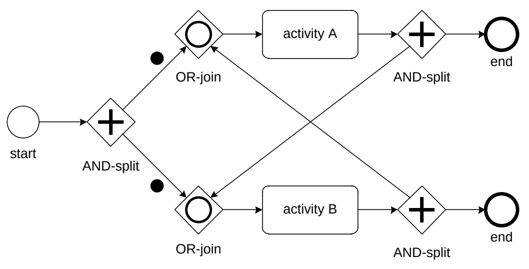

Fig. 18. An example of a so-called "vicious circle" in a BPMN model with two OR-joins.

Figure 18 illustrates the "vicious circle" paradox [69, 66]. The intuitive semantics of an OR-join is to wait for all tokens to arrive. In the state shown in Figure 18, each OR-join has a token on one of its input arcs (denoted by the two black dots). The top OR-join should occur if it cannot receive a token via its second input arc. By symmetry the same holds for the second OR-join. Suppose that one OR-join needs to wait for a second token to arrive, then also the other OR-join needs to wait due to symmetry. However, in this case the process deadlocks and no second token will be received by any of the OR-joins, i.e., none of the OR-joins should have blocked. Suppose that one OR-join does not wait for a second token to arrive, then, by symmetry, also the other OR-join can move forward. However, in this case each OR-join receives a second token and in hindsight both should have blocked. The example shown has no obvious interpretation; however, the paradox revealed by the "vicious circle" also appears in larger, more meaningful, examples where one OR-join depends on another OR-join.

Thus far we only considered procedural languages like Petri nets, BPMN, UML activity diagrams, and BPEL. Although lion's share of BPM research is focusing on such languages, there is also BPM research related to more *declarative* forms of process modeling. Procedural process models take an "inside-to-outside" approach, i.e., all execution alternatives need to be specified explicitly and new alternatives must be explicitly added to the model. Declarative models use an "outside-to-inside" approach: anything is possible unless explicitly forbidden. To illustrate the "outside-to-inside" approach of modeling we use the example shown in Figure 19. The example is expressed in terms of the *Declare* language [70, 71] and is intended to be witty; it does not model a realistic business process but illustrates the modeling constructs. The Declare model consists of four activities (a = eat food, b = feel bad, c = drink beer, and d = drink wine) and four constraints ($c1$, $c2$, $c3$, and $c4$). Without any constraints any sequence of activities is allowed as only constraints can limit the allowed behavior.

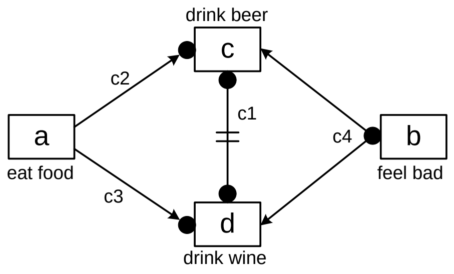

Fig. 19. Example illustrating the declarative style of process modeling where anything is possible unless explicitly forbidden by constraints.

Declare is grounded in *Linear Temporal Logic* (LTL) with finite-trace semantics, i.e., each constraint is mapped onto an LTL formula using temporal operators such as always ($\Box$), eventually ($\Diamond$), until ($\sqcup$), weak until (W), and next time ($\bigcirc$) [72, 73]. The construct connecting activities c and d is a so-called *non-coexistence constraint*. In terms of LTL constraint $c1$ means "$\neg((\Diamond c) \wedge (\Diamond d))$", i.e., $\Diamond c$ and $\Diamond d$ cannot both be true. Hence, it is not allowed that both c and d happen for the same case (beer and wine do not mix well). However, in principle, one of them can occur an arbitrary number of times. There are two *precedence constraints* ($c2$ and $c3$). The semantics of precedence constraint $c2$ which connects a to c can also be expressed in terms of LTL: "$(\neg c)\ W\ a$", i.e., c should not happen before a has happened. Since the weak until (W) is used in "$(\neg c)\ W\ a$", traces without any a and c events also satisfy the constraint. Similarly, d should not happen before a has happened: "$(\neg d)\ W\ a$". There is one branched *response constraint*: $c4$. The LTL formalization of the constraint connecting b to c and d is "$\Box(b \Rightarrow (\Diamond c \vee \Diamond d))$", i.e., every occurrence of b should eventually be followed by c or d. However, there does not need to be a one-to-one correspondence, e.g., four occurrences of activity b may be followed by just one occurrence of activity c. For example, trace $\langle a, c, c, a, b, b, b, b, c\rangle$ is allowed. Whereas in a procedural model, everything is forbidden unless explicitly enabled, a declarative model allows for anything unless explicitly forbidden. Trace $\langle a, a, c, d\rangle$ is not allowed as it violates $c1$ (cannot drink both wine and beer). Trace $\langle b, c, c\rangle$ is not allowed as it violates $c2$ (cannot drink beer before eating food). Trace $\langle a, c, b\rangle$ is not allowed as it violates $c4$ (after feeling bad one should eventually drink beer or wine). For processes with a lot of flexibility, declarative models are often more appropriate [70, 71].

Recently, more and more authors realized that conventional process modeling languages such as BPMN, UML ADs, Statecharts, BPEL, YAWL, WF-nets, and EPCs provide only a *monolithic view* on the real process of interest. The process is "flattened" to allow for a diagram that describes the life-cycle of one case in isolation. *Proclets* [74] are one of the few business process modeling languages not forcing the modeler to *straightjacket* processes into one monolithic model. Instead, processes can be decomposed into a collection of interacting proclets that may have one-to-many or many-to-many relationships (following the cardinalities in the corresponding data model). For example, one order may result in multiple deliveries and one delivery may involve order lines of different orders. This cannot be handled by the classical refinement of activities. However, order, order line, and delivery proclets may coexist independent of one another and are only loosely coupled. For example, an orderline exists because it was created in the context of order. However, the actual delivery of the corresponding item depends on inventory levels, transportation planing, and competing orders.

Object-oriented and artifact-centric approaches use ideas related to proclets [75–80]. These approaches aim to provide a better balance between process-centric and data-centric modeling.

There is an increasing interest in understanding and evaluating the comprehensibility of process models [81–83]. The connection between complexity and process model understanding has been shown empirically in recent publications (e.g. [84–87]) and mechanisms have been proposed to alleviate specific aspects of complexity (e.g. [88–90]). In [82, 83] various change patterns have been proposed. The goal of these patterns is to modify the process model to make it more understandable. The collection of patterns for concrete syntax modifications described in [82] includes mechanisms for arranging the layout, for highlighting parts of the model using enclosure, graphics, or annotations, for representing specific concepts explicitly or in an alternative way, and for providing naming guidance. A collection of patterns for abstract syntax modifications has been presented in [83]. These patterns affect the formal structure of process model elements and their interrelationships (and not just the concrete syntax). For example, a process model may be converted into a behavioral equivalent process model that is block structured and thus easier to understand.

The existence and parallel use of a plethora of languages causes many problems. The lack of consensus makes it difficult to exchange models. The gap between conceptual languages and execution languages leads to re-work and a disconnect between users and implementers. Moreover, conceptual languages and execution languages often do not allow for analysis.

The Workflow Patterns Initiative [91] was established in the late nineties with the aim of delineating the fundamental requirements that arise during business process modeling on a recurring basis and describe them in an imperative way. Based on an analysis of contemporary workflow products and modeling problems encountered in various workflow projects, a set of twenty patterns covering the control-flow perspective of BPM was created [9]. Later this initial set was extended and now also includes workflow resource patterns [24], workflow data patterns [25], exception handling patterns [92], service-interaction patterns [93], and change patterns [94].

These collections of workflow patterns can be used to compare BPM/WFM languages and systems. Moreover, they help focusing on the core issues rather than adding new notations to the "Tower of Babel for Process Languages". The lack of consensus on the modeling language to be used, resulted in a plethora of similar but subtly different languages inhibiting effective and unified process support and analysis. This "Tower of Babel" and the corresponding discussions obfuscated more foundational questions.

5.2 Process Enactment Infrastructures

The Workflow Management Coalition (WfMC) was founded in August 1993 as a international non-profit organization. In the early 1990-ties, the WfMC developed their so-called *reference model* [95, 96]. Although the detailed text describing the reference model refers to outdated standards and technologies, it is remarkable to see that after almost twenty years the reference model of the WfMC still adequately structures the desired functionality of a WFM/BPM system. Figure 20 shows an overview of the reference model. It describes the major components and interfaces within a workflow architecture. In our description of the reference model we use the original terminology. Therefore, "business processes" are often referred to as "workflows" when explaining the reference model.

The core of any WFM/BPM system is the so-called *workflow enactment service*. The workflow enactment service provides the run-time environment which takes care of the control and execution of workflows. For technical or managerial reasons the workflow enactment service may use multiple *workflow engines*. A workflow engine handles selected parts of the workflow and manages selected parts of the resources. The *process definition tools* are used to specify and analyze workflow process definitions and/or resource classifications. These tools are used at design time. In most cases, the process definition tools can also be used for business process modeling and analysis. Most WFM/BPM systems provide three process definition tools: (1) a tool with a graphical interface to define workflow processes, (2) a tool to specify resource classes (organizational model describing roles, groups, etc.), and (3) an analysis tool to analyze a specified workflow (e.g., using simulation or verification). The end-user communicates with the workflow system via the *workflow client applications*. An example of a workflow client application is the well-known *in-basket* also referred to as *work-list*. Via such an in-basket work items are offered to the end user. By selecting a work item, the user can execute a task for a specific case. If necessary, the workflow engine invokes applications via Interface 3. The *administration and monitoring tools* are used to monitor and control the workflows. These tools are used to register the progress of cases and to detect bottlenecks. Moreover, they are also used to set parameters, allocate people and handle abnormalities. Via Interface 4 the workflow system can be connected to other workflow systems.

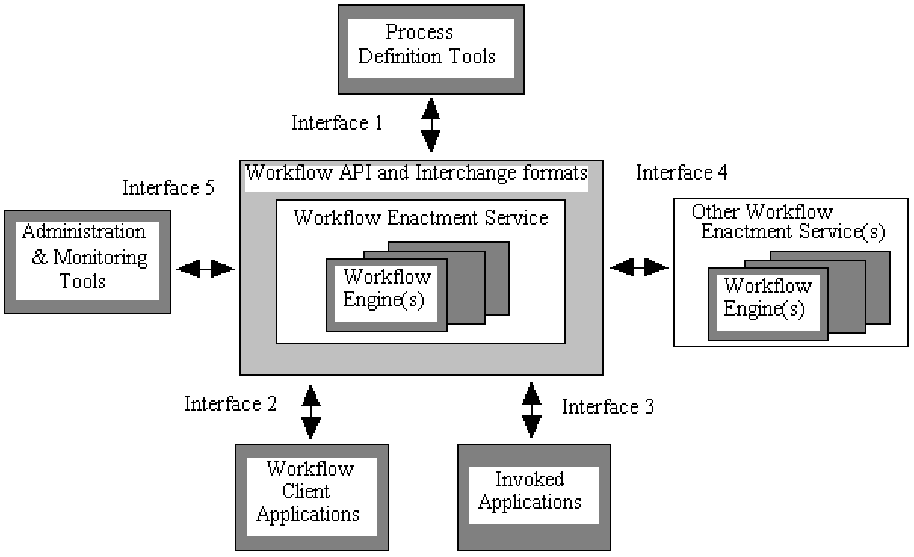
Process Definition Tools
Interface 1
Workflow API and Interchange formats
Interface 5
Interface 4
Administration & Monitoring Tools
Workflow Enactment Service
Workflow Engine(s)
Other Workflow Enactment Service(s)
Workflow Engine(s)
Interface 2
Interface 3
Workflow Client Applications
Invoked Applications

Fig. 20. Reference model of the Workflow Management Coalition (WfMC).

To standardize the five interfaces shown in Figure 20, the WfMC aimed at a common *Workflow Application Programming Interface* (WAPI). The WAPI was envisaged as a common set of API calls and related interchange formats which may be grouped together to support each of the five interfaces (cf. [96]). The WfMC also started to work on a common language to exchange process models soon after it was founded. This resulted in the Workflow Process Definition Language (WPDL) [97] presented in 1999. Although many vendors claimed to be WfMC compliant, few made a serious effort to support this language. At the same time, XML emerged as a standard for data interchange. Since WPDL was not XML-based, the WfMC started working on a new language: XPDL (XML Process Definition Language). The starting point for XPDL was WPDL. However, XPDL should not be considered as the XML version of WPDL. Several concepts have been added/changed and the WfMC remained fuzzy about the exact relationship between XPDL and WPDL. In October 2002, the WfMC released a “Final Draft” of XPDL [98]. The language developed over time, but before widespread adoption, XPDL was overtaken by the *Business Process Execution Language for Web Services* (BPEL) [99, 62]. BPEL builds on IBM’s WSFL (Web Services Flow Language) [100] and Microsoft’s XLANG (Web Services for Business Process Design) [101] and combines accordingly the features of a block structured language inherited from XLANG with those for directed graphs originating from WSFL. BPEL received considerable support from large vendors such as IBM, Oracle, etc. However, in practical terms also the relevance of BPEL is limited. Vendors tend to develop all kinds of extensions (e.g., for people-centric processes) and dialects of BPEL. Moreover, the increasing popularity of BPMN is endangering the position of BPEL (several vendors allow for the direct execution of subsets of BPMN thereby bypassing BPEL).

Furthermore, process models are rarely exchanged between different platforms because of technical problems (the “devil is in the details”) and too few use cases.

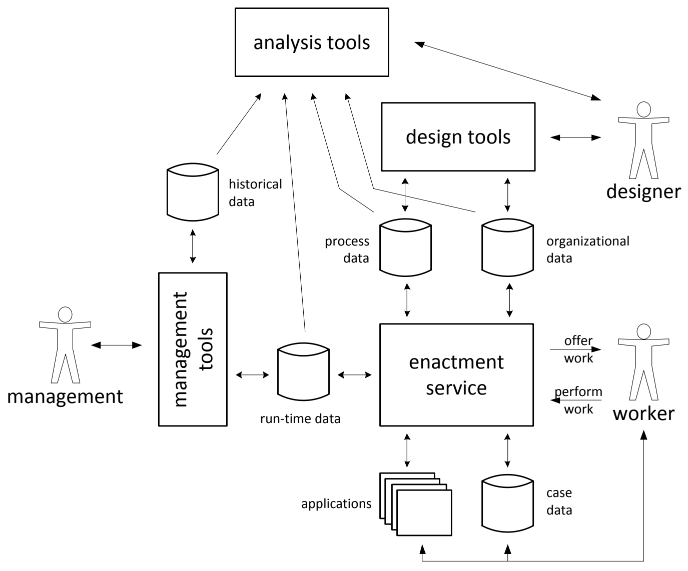

Fig. 21. BPM reference architecture.

Figure 21 shows the BPM reference architecture proposed in [1]. It is similar to the reference model of the WfMC, but the figure details the data sets used and lists the roles of the various stakeholders (management, worker, and designer). The designer uses the design tools to create models describing the processes and the structure of the organization. The manager uses management tools to monitor the flow of work and act if necessary. The worker interacts with the enactment service. The enactment service can offer work to workers and workers can search, select and perform work. To support the execution of tasks, the enactment service may launch various kinds of applications. Note that the enactment service is the core of the system deciding on “what”, “how”, “when”, and “by whom”. Clearly, the enactment service is driven by models of the processes and the organizations using the system. By merely changing these models the system evolves and adapts. This is the ultimate promise of WFM/BPM systems.

Service-Oriented Computing (SOC) has had an incredible impact on the architecture of process enactment infrastructures. The key idea of service-orientation is to *subcontract work to specialized services in a loosely coupled fashion*. In SOC, functionality provided by business applications is encapsulated within web services, i.e., software components described at a semantic level, which can be invoked by application programs or by other services through a stack of Internet standards including HTTP, XML, SOAP, WSDL and UDDI [102–107]. Once deployed, web services provided by various organizations can be inter-connected in order to implement business collaborations, leading to composite web services. Although service-orientation does not depend on a particular technology, it is often associated with standards such as HTTP, XML, SOAP, WSDL, UDDI, and BPEL. Figure 22 shows an overview of the "web services technology stack" and its relation to BPMN and BPEL.

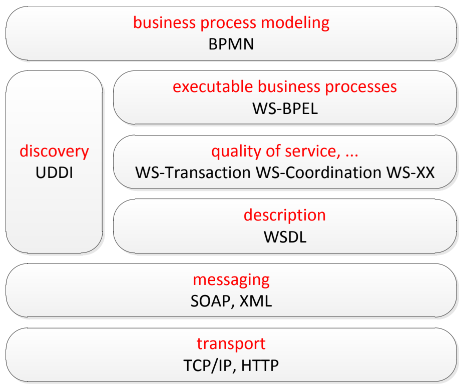

Fig. 22. Web services technology stack linking business process modeling (e.g., using the BPMN notation) to technologies to realize a SOA.

In a *Service Oriented Architecture* (SOA) services are interacting, e.g., by exchanging messages. By combining basic services more complex services can be created [103, 107]. *Orchestration* is concerned with the composition of services seen from the viewpoint of single service (the "spider in the web"). *Choreography* is concerned with the composition of services seen from a global viewpoint focusing on the common and complementary observable behavior. Choreography is particularly relevant in a setting where there is no single coordinator. The terms orchestration and choreography describe two aspects of integrating services to create end-to-end business processes. The two terms overlap somewhat and their distinction has been heavily discussed over the last decade.

SOC and SOA can be used to realize process enactment infrastructures. Processes may implement services and, in turn, may use existing services. All modern BPM/WFM systems provide facilities to expose defined processes as services and to implement activities in a process by simply calling other services. See for example the YAWL architecture [38] which completely decouples the invocation of an activity from the actual execution of the activity.

Interactions between different processes and applications may be more involved as illustrated by the *service interaction patterns* by Barros, Dumas, and Ter Hofstede [93] and the *enterprise integration patterns* by Hohpe and Woolf [108].

For the implementation of process enactment infrastructures, *cloud computing* and related technologies such as *Software as a Service* (SaaS), *Platform as a Service* (PaaS), and *Infrastructure as a Service* (IaaS) are highly relevant. SaaS, often referred to as "on-demand software", is a software delivery model in which software and associated data are centrally hosted on the cloud. The SaaS provider takes care of all hardware and software issues. A well-know example is the collection of services provided by Safesforce. In the PaaS delivery model, the consumer creates the software using tools and libraries from the provider. The consumer also controls software deployment and configuration settings. However, the provider provides the networks, servers and storage. The IaaS delivery model, also referred to as "hardware as a service", offers only computers (often as virtual machines), storage and network capabilities. The consumer needs to maintain the operating systems and application software.

The above discussion of different technologies illustrates that there are many ways to implement the functionality shown in Figure 21. There are both *functional* and *non-functional requirements* that need to be considered when implementing a process-aware information system. The different collections of workflow patterns can be used to elicit functional requirements. For example, the control-flow-oriented workflow patterns [9] can be used elicit requirements with respect to the ordering of activities. An example is the "deferred choice" pattern [9], i.e., a choice controlled by the environment rather than the WFM/BPM system. An organization needs to determine whether this pattern is important and, if so, the system should support it. The workflow resource patterns [24], workflow data patterns [25], and exception handling patterns [92] can be used in a similar fashion. However, architectural choices are mostly driven by non-functional requirements related to costs, response times, and reliability.

Several BPM research groups are concerned with the performance of WFM/BPM systems. Although the core process management technology by itself is seldom the bottleneck, some process-aware information systems need to deal with millions of cases and thousands of concurrent users. Note that process-related data are typically small compared to the data needed to actually execute activities. The process state can be encoded compactly and the computation of the next state is typically very fast compared to application-related processing. However, since WFM/BPM systems needs to control other applications, architectural considerations are important for the overall system's performance. For example, when the number of cases handled per hour grows over time, there is a need to reconfigure the system and to distribute the work over more computing nodes. Cloud computing and SaaS provide the opportunity to outsource such issues. Load balancing and system reconfiguration are then handled by the service provider.

Another concern addressed by BPM research is the *reliability* of the resulting information system. WFM/BPM systems are often the "spider in the web" connecting different technologies. For example, the BPM system invokes applications to execute particular tasks, stores process-related information in a database, and integrates different legacy and web-based systems. Different components may fail resulting in loss of data and parts of the systems that are out-of-sync. Ideally, the so-called *ACID properties* (Atomicity, Consistency, Isolation and Durability) are ensured by the WFM/BPM system; *atomicity*: an activity is either successfully completed in full (commit) or restarts from the very beginning (rollback), *consistency*: the result of an activity leads to a consistent state, *isolation*: if several tasks are carried out simultaneously, the result is the same as if they had been carried out entirely separately, and *durability*: once a task is successfully completed, the result must be saved persistently to ensure that work cannot be lost. In the second half of the nineties many database researchers worked on so-called workflow transactions, i.e., long-running transactions ensuring the ACID properties at a business process level [109, 40, 110–113]. Business processes need to be executed in a partly uncontrollable environment where people and organizations may deviate and software components and communication infrastructures may malfunction. Therefore, the BPM system needs to be able to deal with failures and missing data. Research on workflow transactions [109, 40, 110–113] aims to gracefully handle exceptions and maintain system integrity at all times.

Related to reliability are *security* concerns. WFM/BPM systems should ensure that only authorized people can execute activities and access data [114]. Role-Based Access Control (RBAC, [115]) techniques can be applied in this setting. The workflow resource patterns [24] also incorporate RBAC functionalities. Moreover, process-specific security patterns such as the "four-eyes principle" (the same person may not execute two dependent tasks for the same case even if the person has the appropriate role for both tasks) are incorporated. Cloud computing and SaaS technologies fuel new security-related anxieties. Multi-tenancy, i.e., multiple organizations using the same system, is interesting from a cost perspective. Costs are shared by different organizations using economies of scale. Moreover, load balancing and reconfiguration can be supported in a better manner when many tenants are sharing a large common infrastructure. For example, smaller organizations may share a workflow engine whereas larger organizations use many engines at the same time. This is all handled by the service provider. For the service consumer these system (re)configurations are invisible. However, multi-tenancy implies that different, possibly competing, organizations are using the same cloud or SaaS system. Therefore, the process infrastructure should ensure that information from one tenant cannot leak to another tenant.

5.3 Process Model Analysis

There are two mainstream approaches for model-based analysis: *verification* and *performance analysis*. Verification is concerned with the correctness of a system or process. Performance analysis focuses on flow times, waiting times, utilization, and service levels. Unlike process-mining, these approaches do not use event data and perform analysis using just the model.

A typical correctness property used for verification is the *soundness* notion [13, 52]. Soundness was originally defined for *workflow nets* (WF-nets) but it applies to all modeling techniques. A WF-net is a Petri net with a dedicated source place where the process starts, and a dedicated sink place where the process ends. Moreover, all nodes are on a path from source to sink. A token on the source place denotes the initial state. The state with just a token on the sink place denotes the desired end state. Such a WF-net models the *life-cycle of cases* of a given kind. Examples of cases are insurance claims, job applications, customer orders, replenishment orders, patient treatments, and credit applications. The process model is instantiated once for each case. Each of these process instances has a well-defined start ("case creation") and end ("case completion"). In-between these points, activities are conducted according to a predefined procedure. One model may be instantiated many times. For example, the process of handling insurance claims may be executed for thousands or even millions of claims. These instances can be seen as copies of the same WF-net, i.e., tokens of different cases are not mixed.

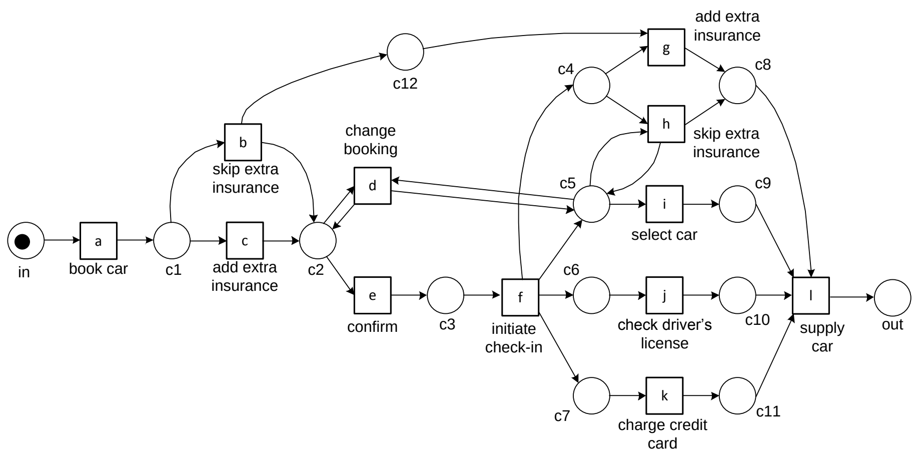

Fig. 23. A WF-net that is not sound: activity d is dead (can never be executed), cases may deadlock in the state with a token in $c4$, $c9$, $c10$, and $c11$, and a token may be left behind in place $c12$.

Not every WF-net represents a correct process. The modeled process may exhibit errors such as deadlocks, activities that can never become active, livelocks, and improper termination (i.e., garbage being left in the process after completion). Consider for example the WF-net shown in Figure 23 exhibiting several problems.

A WF-net is *sound* if and only if (a) from any reachable state it is possible to reach a state with a token in the sink place (*option to complete*), (b) any reachable state having a token in the sink place does not have a token in any of the other places (*proper completion*), and (c) for any transition there is a reachable state enabling it (*absence of dead parts*) [13, 52]. The WF-net shown in Figure 23 obviously violates all three properties. For subclasses of WF-nets, soundness can be analyzed without constructing the state space. For example, for free-choice Petri nets, i.e., processes where choice and synchronization can be separated, soundness can be checked by analyzing the rank of the corresponding incidence matrix [13, 116]. Hence, soundness can be checked in polynomial time for free-choice WF-nets. Invariants can often be used to diagnose soundness problems, e.g., the absence of particular place and transition invariants for the short-circuited WF-net provides possible causes for non-soundness. However, most of the more interesting verification questions require the exploration of (a part of) the state space. See [13, 65, 117, 118, 52, 119–125, 55, 126–131, 68] for examples of verification techniques analyzing soundness-related properties for workflows and business processes

Soundness is a generic property, but sometimes a more specific property needs to be investigated, e.g., "the ticket was checked for all rejected requests". Such properties can be expressed in *temporal logic* [72, 73]. As mentioned earlier Linear Temporal Logic

(LTL) is an example of a temporal logic that, in addition to classical logical operators, uses temporal operators such as: always ($\Box$), eventually ($\Diamond$), until ($\sqcup$), weak until (W), and next time ($\bigcirc$). The expression $\Diamond b \Rightarrow \Diamond g$ means that for all cases in which b (*skip extra insurance*) is executed also g (*add extra insurance*) is executed. Another example is $\Box(e \Rightarrow \Diamond l)$ that states that any occurrence of e will eventually be followed by l (after confirmation eventually a car is supplied). Model checking techniques can be used to check such properties [72].

Another verification task is the comparison of two models. For example, the implementation of a process is compared to the high-level specification of the process. There exist different equivalence notions (trace equivalence, branching bisimilarity, etc.) [132, 133]. Trace equivalence considers two transition systems to be equivalent if their execution sequences are the same. More refined notions like (branching) bisimilarity also take the moment of choice into account [132, 133]. Two process models are bisimilar if the first model can "mimic any move" of the second, and vice versa. Consider for example the processes $P = a.(b+c)$ and $Q = a.b+a.c$. Both process can generate traces $\langle a,b\rangle$ and $\langle a,c\rangle$. However, in process P the choice between b and c is made after the occurrence of a whereas in Q this choice is made upfront, i.e., before the concurrence of a. To understand that such differences are relevant replace a, b and c by "take exam", "pass", and "fail" respectively.

Also in the context of services soundness-like properties have been investigated [134, 117, 135–144]. These techniques focus on uncovering problems related to interactions between different parties or services. For example, one service is waiting for the other service to make the first move and vice versa. Note that one can easily design services that cannot interoperate with any other service. The approach using the so-called operating guidelines [144] computes a finite characterization of all partner services, i.e., services that can interoperate well with a given service.

Configurable models represent families of process models [47, 145, 146, 46, 147]. A configurable model can be configured to obtain a *specific* process model that is subsequently used to handle individual cases, for instance, to process customer orders. Various configurable languages have been proposed as extensions of existing languages (e. g., C-EPCs [46], C-iEPCs [146], C-WF-nets [148], C-SAP and C-BPEL [47]) but few are actually supported by enactment software (e. g., C-YAWL [47]). Process configuration is notoriously difficult as there may be all kinds of interdependencies between configuration decisions. In fact, an incorrect configuration may lead to behavioral issues such as deadlocks and livelocks. The approach presented in [148] derives propositional logic constraints from configurable process models that, if satisfied by a configuration step, guarantee the behavioral correctness of the configured model. The approach in [51] ensures this by using partner synthesis: for a configurable process model a finite representation of all correct configurations is generated.

There are various tools to verify process/workflow models. A classical example is Woflan that is tailored towards checking soundness [149]. Also workflow systems such as YAWL [150] provide verification capabilities [68]. The tool Wendy [151] is an example of a tool tailored towards partner synthesis. See [55, 126] for a comparative evaluation of several verification tools checking soundness-related properties.

Obviously, model-based analysis is not limited to correctness. In fact, from a management point of view, performance analysis is more relevant. The performance of a process or organization can be defined in different ways. Typically, three dimensions of performance are identified: *time*, *cost* and *quality*. For each of these performance dimensions different *Key Performance Indicators* (KPIs) can be defined. When looking at the *time dimension* the following performance indicators can be identified:

- The *lead time* (also referred to as flow time) is the total time from the creation of the case to the completion of the case. In terms of a WF-net, this is the time it takes to go from source place to sink place. One can measure the average lead time over all cases. However, the degree of variance may also be important, i.e., it makes a difference whether all cases take more or less two weeks or if some take just a few hours whereas others take more than one month. The *service level* is the percentage of cases having a lead time lower than some threshold value, e.g., the percentage of cases handled within two weeks.
- The *service time* is the time actually worked on a case. One can measure the service time per activity (e.g., the average time needed to make a decision is 35 minutes) or for the entire case. Note that in case of concurrency the overall service time (i.e., summing up the times spent on the various activities) may be longer than the lead time. However, typically the service time is just a fraction of the lead time (minutes versus weeks).
- The *waiting time* is the time a case is waiting for a resource to become available. This time can be measured per activity or for the case as a whole. An example is the waiting time for a customer who wants to talk to a sales representative. Another example is the time a patient needs to wait before getting a knee operation. Again one may be interested in the average or variance of waiting times. It is also possible to focus on a service level, e.g., the percentage of patients that has a knee operation within three weeks after the initial diagnosis.
- The *synchronization time* is the time an activity is not yet fully enabled and waiting for an external trigger or another parallel branch. The time the case is partially enabled (i.e., waiting for synchronization rather than an available resource) is counted as synchronization time.

Performance indicators can also be defined for the *cost dimension*. Different costing models can be used, e.g., Activity Based Costing (ABC), Time-Driven ABC, and Resource Consumption Accounting (RCA) [152]. The costs of executing an activity may be fixed or depend on the type of resource used, its utilization, or the duration of the activity. Resource costs may depend on the utilization of resources. A key performance indicator in most processes is the *average utilization* of resources over a given period, e.g., an operating room in a hospital has been used 85% of the time over the last two months.

The *quality dimension* typically focuses on the "product" or "service" delivered to the customer. Like costs, this can be measured in different ways. One example is customer satisfaction measured through questionnaires. Another example is the average number of complaints per case or the number of product defects.

Whereas verification focuses on the (logical) correctness of the modeled process, performance analysis aims at improving processes with respect to time, cost, or quality.

Within the context of operations management many analysis techniques have been developed [153–156]. Some of these techniques "optimize" the model given a particular performance indicator. For example, integer programming or Markov decision problems can be used to find optimal policies. For typical BPM problems "what if" analyses using simulation, queueing models, or Markov models are often most appropriate. Analytical models typically require many assumptions and can only be used to answer particular questions. Therefore, one often needs to resort to *simulation*. Most BPM tools provide simulation capabilities.

Although many organizations have tried to use simulation to analyze their business processes at some stage, few are using simulation in a structured and effective manner. This may be caused by a lack of training and limitations of existing tools. However, there are also several additional and more fundamental problems. First of all, simulation models tend to *oversimplify* things. In particular the behavior of resources is often modeled in a rather naïve manner. People do not work at constant speeds and need to distribute their attention over multiple processes. This can have dramatic effects on the performance of a process and, therefore, such aspects should not be "abstracted away" [157, 158]. Second, various *artifacts readily available are not used as input for simulation*. Modern organizations store events in logs and some may have accurate process models stored in their WFM/BPM systems. Also note that in many organizations, the state of the information system accurately reflects the state of the business processes supported by this system. Nevertheless, such information (i.e., event logs and status data) is rarely used for simulation or a lot of manual work is needed to feed this information into the model. Third, the focus of simulation is mainly on "design" whereas managers would also like to use simulation for *"operational decision making"*, i.e., solving the concrete problem at hand rather than some abstract future problem. Fortunately, *short-term simulation* [157] can provide answers for questions related to "here and now". The key idea is to start all simulation runs from the current state and focus on the analysis of the transient behavior. This way a "fast forward button" into the future is provided [8, 157].

Verification and performance analysis heavily rely on the availability of high quality models. When the models and reality have little in common, model-based analysis does not make much sense. For example, some process model may be internally consistent and satisfy all kinds of desirable properties. However, if the model describes a highly idealized version of reality, it may be useless for governance and auditing purposes as in reality all kinds of deviations may take place. Similar comments hold for simulation models. It may be that the model predicts a significant improvement whereas in reality this is not the case because the model is based on flawed assumptions. All of these problems stem from *a lack of alignment between hand-made models and reality.* Process mining, discussed next, aims to address these problems by establishing a direct connection between the models and actual low-level event data about the process.

5.4 Process Mining

As information systems are becoming more and more intertwined with the operational processes they support, multitudes of events are recorded by these systems. The goal of *process mining* is to use such event data to extract process related information, e.g., to automatically discover a process model by observing events recorded by some system or to check the conformance of a given model by comparing it with reality [8, 159]. This provides 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. 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. Hence, despite the omnipresence of event data, most organizations diagnose problems based on fiction rather than facts.

Process mining is an emerging discipline providing comprehensive sets of tools to provide fact-based insights and to support process improvements [8, 160]. 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.

Figure 24 shows the *process mining framework* described in [8]. The top of the diagram shows an external "world" consisting of business processes, people, and organizations supported by some information system. The information system records information about this "world" in such a way that events logs can be extracted. The term *provenance* used in Figure 24 emphasizes the systematic, reliable, and trustworthy recording of events. The term provenance originates from scientific computing, where it refers to the data that is needed to be able to reproduce an experiment [42, 161]. *Business process provenance* aims to systematically collect the information needed to reconstruct what has actually happened in a process or organization [162]. When organizations base their decisions on event data it is essential to make sure that these describe history well. Moreover, from an auditing point of view it is necessary to ensure that event logs cannot be tampered with. Business process provenance refers to the set of activities needed to ensure that history, as captured in event logs, "cannot be rewritten or obscured" such that it can serve as a reliable basis for process improvement and auditing.

As shown in Figure 24, event data can be partitioned into *"pre mortem"* and *"post mortem"* event logs. "Post mortem" event data refer to information about cases that have completed, i.e., these data can be used for process improvement and auditing, but not for influencing the cases they refer to. "Pre mortem" event data refer to cases that have not yet completed. If a case is still running, i.e., the case is still "alive" (pre mortem), then it may be possible that information in the event log about this case (i.e., current data) can be exploited to ensure the correct or efficient handling of this case.

"Post mortem" event data are most relevant for *off-line process mining*, e.g., discovering the control-flow of a process based on one year of event data. For *online process mining* mixtures of "pre mortem" (current) and "post mortem" (historic) data are needed. For example, historic information can be used to learn a predictive model. Sub-

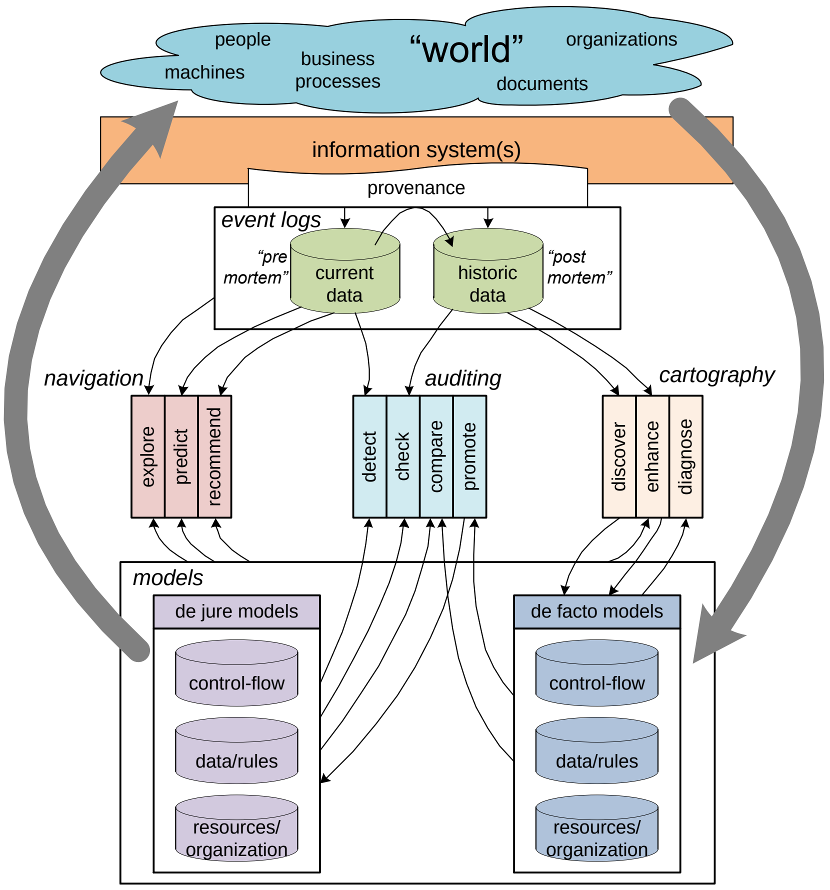

Fig. 24. Overview of the process mining spectrum.

sequently, information about a running case is combined with the predictive model to provide an estimate for the remaining flow time of the case.

The process mining framework described in [8] also distinguishes between two types of models: *"de jure models"* and *"de facto models"*. *A de jure model is normative*, i.e., it specifies how things should be done or handled. For example, a process model used to configure a BPM system is normative and forces people to work in a particular way. *A de facto model is descriptive* and its goal is not to steer or control reality. Instead, de facto models aim to capture reality. As shown in Figure 24 both de jure and de facto models may cover multiple perspectives including the *control-flow perspective* ("How?"), the *organizational perspective* ("Who?"), and the *case perspective* ("What?"). The control-flow perspective describes the ordering of activities. The organizational perspective describes resources (worker, machines, customers, services, etc.)

and organizational entities (roles, departments, positions, etc.). The case perspective describes data and rules.

In the middle of Figure 24 ten process mining related activities are depicted. These ten activities are grouped into three categories: *cartography*, *auditing*, and *navigation*. The activities in the cartography category aim at making "process maps". The activities in the auditing category all involve a de jure model that is confronted with reality in the form of event data or a de facto model. The activities in the navigation category aim at improving a process while it is running.

Activity *discover* in Figure 24 responds to use case DiscM (discover model from event data) described earlier. Lion's share of process mining research has been devoted to this activity [8, 163]. A discovery technique takes an event log and produces a model without using any additional a-priori information. An example is the α-algorithm [44] that takes an event log and produces a Petri net explaining the behavior recorded in the log. 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.

Since the mid-nineties several groups have been working on techniques for process discovery [160, 44, 164–169]. In [170] an overview is given of the early work in this domain. The idea to apply process mining in the context of workflow management systems was introduced in [164]. In parallel, Datta [166] looked at the discovery of business process models. Cook et al. investigated similar issues in the context of software engineering processes [165]. Herbst [171] was one of the first to tackle more complicated processes, e.g., processes containing duplicate tasks.

Most of the classical approaches have problems dealing with concurrency. The α-algorithm [44] is an example of a simple technique that takes concurrency as a starting point. However, this simple algorithm has problems dealing with complicated routing constructs and noise (like most of the other approaches described in literature). Process discovery is very challenging because techniques need to balance four criteria: *fitness* (the discovered model should allow for the behavior seen in the event log), *precision* (the discovered model should not allow for behavior completely unrelated to what was seen in the event log), *generalization* (the discovered model should generalize the example behavior seen in the event log), and *simplicity* (the discovered model should be as simple as possible). This makes process discovery a challenging and highly relevant research topic.

Activity *enhance* in Figure 24 corresponds to use cases RepM (repair model) and ExtM (extend model). When existing process models (either discovered or hand-made) can be related to events logs, it is possible to enhance these models. The connection can be used to repair models [53] or to extend them [172–175].

Activity *diagnose* in Figure 24 does not directly use event logs and focuses on classical model-based process analysis as discussed in Section 5.3.

Activity *detect* compares de jure models with current "pre mortem" data (events of running process instances) with the goal to detect deviations at run-time. The moment a predefined rule is violated, an alert is generated [176–178].

Activity *check* in Figure 24 refers to use case ConfED (check conformance using event data). Historic "post mortem" data can be cross-checked with de jure models. The goal of this activity is to pinpoint deviations and quantify the level of compliance. Various conformance checking techniques have been proposed in literature [179–188]. For example, in [187] the fitness of a model is computed by comparing the number of missing and remaining tokens with the number of consumed and produced tokens during replay. The more sophisticated technique described in [179–181] creates as so-called *alignment* which relates a trace in the event log to an execution sequence of the model that is as similar as possible. Ideally, the alignment consists of steps where log and model agree on the activity to be executed. Steps where just the model "makes a move" or just the log "makes a move" have a predefined penalty. This way the computation of fitness can be turned into an optimization problem: for each trace in the event log an alignment with the lowest costs is selected. The resulting alignments can be used for all kinds of analysis since any trace in the event log is related to an execution sequence of the model. For example, timestamps in the model can be used to compute bottlenecks and extend the model with performance information (see activity *enhance* in Figure 24).

Activity *compare* highlights differences and commonalities between a de jure model and a de facto model. Traditional equivalence notions such as trace equivalence, bisimilarity, and branching bisimilarity [132, 133] can only be used to determine equivalence using a predefined equivalence notion, e.g., these techniques cannot be used to distinguish between very similar and highly dissimilar processes. Other notions such a graph-edit distance tend to focus on the syntax rather than the behavior of models. Therefore, recent BPM research explored various alternative similarity notions [189, 56, 190–193] Also note the *Greatest Common Divisor* (GCD) and *Least Common Multiple* (LCM) notions defined for process models in [194]. The GCD captures the common parts of two or more models. The LCM embeds all input models. We refer to [189] for a survey and empirical evaluation of some similarity notions.

Activity *promote* takes (parts of) de facto models and converts these into (parts of) de jure models, i.e., models used to control or support processes are improved based on models learned from event data. By promoting proven "best practices" to de jure models, existing processes can be improved.

The activities in the cartography and auditing categories in Figure 24 can be viewed as "backward-looking". The last three activities forming the navigation category are "forward-looking" and are sometimes referred to as *operational support* [8]. For example, process mining techniques can be used to make predictions about the future of a particular case and guide the user in selecting suitable actions. When comparing this with a car navigation system from TomTom or Garmin, this corresponds to functionalities such predicting the arrival time and guiding the driver using spoken instructions.

Activity *explore* in Figure 24 visualizes running cases and compares these cases with similar cases that were handled earlier. The combination of event data and models can be used to explore business processes at run-time and, if needed, trigger appropriate actions.

By combining information about running cases with models (discovered or hand-made), it is possible to make predictions about the future, e.g., predicting the remaining flow time or the probability of success. Figure 24 shows that activity *predict* uses current data and models (often learned over historic data). Various techniques have been proposed in BPM literature [195–197]. Note that already a decade ago Staffware provided a so-called "prediction engine" using simulation [198].

Activity *recommend* in Figure 24 aims to provide functionality similar to the guidance given by car navigation systems. The information used for predicting the future can also be used to recommend suitable actions (e.g. to minimize costs or time) [176, 199]. Given a set of possible next steps, the most promising step is recommended. For each possible step, simply assume that the step is made and predict the resulting performance (e.g., remaining flow time). The resulting predictions can be compared and used to rank the possible next steps.

The ten activities in Figure 24 illustrate that process mining extends far beyond process discovery. The increasing availability and growing volume of event data suggest that the importance of process mining will continue to grow in the coming years.

5.5 Process Flexibility

Effective business processes must be able to accommodate changes in the environment in which they operate, e.g., new laws, changes in business strategy, or emerging technologies. The ability to encompass such changes is termed *process flexibility* and is definitely a key concern of BPM as is reflected by various publications [200–207]. Modern processes and information systems need to be able to deal with both foreseen and unforeseen changes. This quality of a process – termed flexibility – reflects its ability to deal with such changes, by varying or adapting those parts of the business process that are affected by them, whilst retaining the essential format of those parts that are not impacted by the variations. Indeed, flexibility is as much about what should stay the same in a process as what should be allowed to change [208, 209].

In [209] a taxonomy of process flexibility is presented. The taxonomy identifies four main flexibility types: flexibility by *definition*, flexibility by *deviation*, flexibility by *underspecification*, and flexibility by *change* (cf. Figure 25).

Flexibility by definition is the ability to incorporate alternative execution paths within a process definition at design time such that selection of the most appropriate execution path can be made at runtime for each process instance. For example, an XOR-split defined at design time adds the ability to select one or more activities for subsequent execution from a set of available activities. Parallelism defined at design time leaves the actual ordering of activities open and thus provides more flexibility than sequential routing. All WFM/BPM systems support this type of flexibility. However, declarative languages make it easier to defer choices to runtime.

The classical workflow patterns mentioned earlier [9, 91] can be viewed as a classification of "flexibility by definition" mechanisms for procedural languages. For example, the "deferred choice" pattern [9] leaves the resolution of a choice to the environment at runtime. Note that a so-called "flower place" in a Petri net, i.e., a place with many transitions that have this place as only input and output place, provides a lot of flexibility. Also declarative languages like Declare [70, 71] can be used to provide a lot of flexibility at runtime. (As discussed in Section 5.1, declarative models use an "outside-to-inside" approach: anything is possible unless explicitly forbidden.)

Flexibility by deviation is the ability for a process instance to deviate at runtime from the execution path prescribed by the original process without altering the process

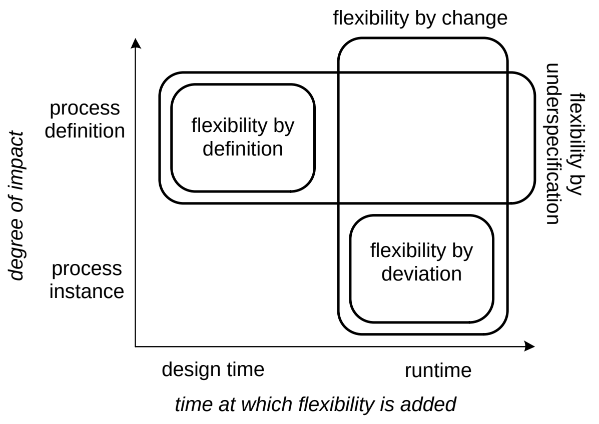

Fig. 25. Taxonomy of process flexibility identifying four main flexibility types: flexibility by *definition*, flexibility by *deviation*, flexibility by *underspecification*, and flexibility by *change*.

definition itself. The deviation can only encompass changes to the execution sequence for a specific process instance, and does not require modifications of the process definition. Typical deviations are *undo*, *redo*, and *skip*.

The BPM|one system of Perceptive/Lexmark (based on the FLOWer system developed by Pallas Athena) is a system that provides various mechanisms for deviations at runtime. The case handling paradigm [200] supported by BPM|one allows the user to skip or redo activities (if not explicitly forbidden and assuming the user is authorized to do so). Moreover, data can be entered earlier or later because the state is continuously recomputed based on the available data.

Flexibility by underspecification is the ability to execute an incomplete process specification, i.e., a model that does not contain sufficient information to allow it to be executed to completion. An incomplete process specification contains one or more so-called *placeholders*. These placeholders are nodes which are marked as underspecified (i.e., “holes” in the specification) and whose content is specified during the execution of the process. The manner in which these placeholders are ultimately enacted is determined by applying one of the following approaches: *late binding* (the implementation of a placeholder is selected from a set of available process fragments) or *late modeling* (a new process fragment is constructed in order to complete a given placeholder). For late binding, a process fragment has to be selected from an existing set of fully predefined process fragments. This approach is limited to selection, and does not allow a new process fragment to be constructed. For late modeling, a new process fragment can be developed from scratch or composed from existing process fragments.

In the context of YAWL [150], the so-called *worklets* approach [201] has been developed which allows for late binding and late modeling. Late binding is supported through so-called “ripple-down rules”, i.e., based on context information the user can be guided to selecting a suitable fragment. In [210] the term “pockets of flexibility” was introduced to refer to the placeholder for change. In [211] an explicit notion of "vagueness" is introduced in the context of process modeling. The authors propose model elements such as arc conditions and task ordering to be deliberately omitted from models in the early stages of modeling. Moreover, parts of the process model can be tagged as "incomplete" or "unspecified".

Flexibility by change is the ability to modify a process definition at run-time such that one or all of the currently executing process instances are migrated to a new process definition. Changes may be introduced both at the process instance and the process type levels. A *momentary change* (also known as change at the instance level) is a change affecting the execution of one or more selected process instances. An example of a momentary change is the postponement of registering a patient that has arrived to the hospital emergency center: treatment is started immediately rather than spending time on formalities first. Such a momentary change performed on a given process instance does not affect any future instances. An *evolutionary change* (also known as change at the type level) is a change caused by modification of the process definition, potentially affecting all new process instances. A typical example of the evolutionary change is the redesign of a business process to improve the overall performance characteristics by allowing for more concurrency. Running process instances that are impacted by an evolutionary or a momentary change need to be handled properly. If a running process instance is transferred to the new process, then there may not be a corresponding state (called the "dynamic change bug" in [203]).

Flexibility by change is very challenging and has been investigated by many researchers. The ability to adapt the structure of running workflow was investigated in the context of the WASA system [207]. In the context of the ADEPT system, flexibility by change has been examined in detail [205, 206]. This work shows that changes can introduce all kinds of anomalies (missing data, deadlocks, double work, etc.). For example, it is difficult to handle both momentary changes and evolutionary changes at the same time, e.g., an ad-hoc change made for a specific instance may be affected by a later change at the type level. The declarative workflow system Declare has been extended to support both evolutionary and momentary changes [204] thus illustrating that a declarative style of modeled simplifies the realization of all kinds of flexility support.

See also [208, 212, 40, 213–215, 210] for other classifications of flexibility.

5.6 Process Reuse

BPM initiatives within larger organizations resulted in collections of hundreds or even thousands of process models. Such large collections of process models provide new challenges, sometimes referred to as "BPM-in-the-large" [216]. A recent survey [217] shows that since 2005 there has been a growing research interest in the management of large collections of business process models. The survey also refers to examples of large collections, e.g., Suncorp's process model repository containing more than 6,000 insurance-related processes. Organizations having hundreds or thousands of process models often have problems maintaining these models. Some models may be outdated, parts of models may have been duplicated, and due to mergers there may be different models for similar or even identical processes. Reuse is limited, i.e., even though many processes share activities, subprocesses, and organizational entities, processes are often modeled from scratch. BPM research aims to support the reuse of process modeling efforts.

Process model repositories allow for the storage and retrieval of process models. Most business process modeling tools, e.g., tools like ARIS [218, 219], provide such facilities. The well-known SAP reference model consisting of over 600 non-trivial process models (expressed in terms of EPCs) has been distributed with the ARIS toolset. A more recent initiative is APROMORE [220, 221], an advanced process model repository providing a rich set of features for the analysis, management and usage of large sets of process models.

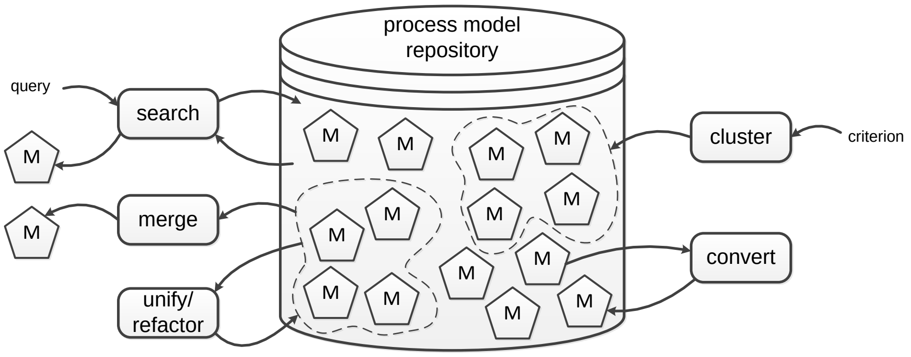

Fig. 26. Overview of the main activities related to the management of large process model collections.

Figure 26 shows various activities related to the management of large collections of business process models stored in some repository.

Activity *search* in Figure 26 refers to use case SelM (select model from collection). Given a query, a set of models is returned. The returned models are often ranked based on some metric (e.g., similarity or popularity). The query may refer to *syntax* (i.e., structure and labels) or *behavior*. Example queries referring to only the syntax are "Find all process models that contain both activity X and Y", "Find all process models containing activities executed by people having role R", and "Find all process models containing activities accessing data element D". An example of a query that also refers to behavior is "Find all process models where activity X is always followed by Y". Sometimes behavior can be derived from the syntax, e.g., for free-choice nets [116, 130]. Queries referring to behavior typically use some temporal logic, e.g., LTL with standard temporal operators such as always ($\Box$), eventually ($\Diamond$), until ($\sqcup$), weak until (W), and next time ($\bigcirc$) [72, 73]. Such queries can be formulated graphically using a language like Declare [70, 71]. Another query language is the Business Process Model Notation Query (BPMN-Q) language [222]. BPMN-Q can be used to define patterns using an extension of the BPMN syntax. Both Declare and BPMN-Q can also be used for compliance checking.

A *model similarity search* [56, 189, 191, 190] is a particular query looking for the model most similar to a given model. For model similarity searches both syntax and behavior can be used. For example, given one model one may want to find another model that has the smallest edit distance (i.e., the number of atomic edit operation to convert one model into another model). However, two behavioral equivalent models may have many different syntactical differences. Therefore, various approaches consider (an abstraction of) behavior. Since it is often intractable to compare state spaces or execution sequences, these approaches use abstractions of models such as direct succession [8] or eventual succession [189, 223].

Queries can refer to multiple perspectives. However, current research seems to focus on control-flow related queries.

Activity *merge* in Figure 26 corresponds to use cases MerM (merge models) and MerCM (merge models into configurable model). A set of models is merged into a single model that captures (most of) the behavior of the original models. For example, in [224] models of ten Dutch municipalities are merged into configurable process models [47, 146, 46]. Different techniques for process model merging have been proposed in literature [225, 145, 226, 227]. When merging process models it is interesting to analyze commonalities and differences. In the context of inheritance of dynamic behavior, notions such as the Greatest Common Divisor (GCD) and Least Common Multiple (LCM) of process model variants have been defined [194]. When merging models it is often not sufficient to just consider the syntax of the model. Also behavioral issues need to be considered. For example, a sequential process may be embedded in a more concurrent model.

In [227], three requirements are listed for model merging. First of all, the behavior of the merged model should subsume the behaviors of all input models. Any execution sequence possible in one of the original models should be possible according to the merged model (possibly after configuration). Second, it should be possible to trace back each element in the merged model. For example, for each activity in the merged model it should be indicated from which of the input models it originated. Third, given the merged model it should be possible to reconstruct each of the input models, i.e., each of the input models should correspond to a configuration of the resulting merged model. For example, in Figure 9 the two input models can be reconstructed from the configurable model by selecting appropriate configurations.

The approaches described in [225, 145, 226, 227, 224] produce configurable process models [47, 146, 46]. In [228, 229] an approach is presented that does not produce a configurable model and does not aim to address the three requirements listed in [227]. This approach produces a model that has the smallest edit distance to all original models, i.e., modification rather than configuration is used to create process model variants.

Activity *cluster* in Figure 26 aims to identify a set of related process models. For example, models may be clustered in groups based on similarity search [189]. Clusters of related models may be used as input for merging, unification, or refactoring.

Activity *unify/refactor* in Figure 26 takes a set of models as input and aims to improve these models by aligning them, removing redundancies, and applying modeling conventions consistently. Note that large collections of process models often have overlapping process fragments without explicitly reusing parts. Shared subprocesses may be modeled differently, models may use different conventions, and there may be different versions of the same processes. Model similarity search can be used to identify possible redundancies before adding a new model.

Activity *convert* in Figure 26 refers to the various mappings from one notation to another notation. As described in use case RefM (refine model) a conceptual model may be converted into an executable model. It may also be converted into a formal model that allows for analysis. Often a repository contains models using different formats while referring to the same process. It is far from trivial to keep all of these models consistent, e.g., changes in the conceptual model should be reflected in the executable model.

A general problem effecting all activities in Figure 26, is the use for informal text. The same activity may be labeled "approve claim" in one process and "evaluate insurance claim" in another process. As a result the correspondence between both activities may be missed and redundancies and inconsistencies remain unnoticed. To determine the similarity between activity names in different models one can use naïve approaches such as the string edit distance [230] or linguistic similarity (e.g., similarity based on WordNet [231]). However, it is better to use a common ontology. Semantic technologies [232] aim to address obvious problems related to string edit distance and linguistic similarity. However, in practice, few process model collections use a common ontology. Therefore, in most cases, semantical annotations still need to be added to process models before being able to use semantic technologies.

5.7 Evolution of Key Concerns in BPM Conference Proceedings

As for the use cases, the papers in [36], [26], [27], [28], [29], [30], [31], [32], [33], and [34] were tagged with one, or sometimes more, key concerns [54]. A total of 342 tags were assigned to the 289 papers (1.18 tag per paper on average). The tags were used to determine the relative frequencies listed in Table 2. For example, for BPM 2010 four papers were tagged with key concern *process reuse*. The total number of tags for BPM 2010 is 25. Hence, the relative frequency is $4/25 = 0.16$. The bottom row gives the average relative frequency of each concern over all 10 years.

Figure 27 shows the average relative frequency of each concern in a graphical manner. As expected, the first three concerns are most frequent. The fourth and sixth concern (process mining and process reuse) are gaining importance, whereas the relative frequency of the process flexibility concern seems to decrease over time (see Figure 28).

It should be noted that the tagging of the 289 papers with use cases and key concerns is highly subjective. It is unlikely that two BPM experts would use precisely the same tags for all papers. For example, to tag a paper one needs to decide what the key contribution of the paper is. Many papers are rather broad and difficult to classify. For example, papers on topics such as "Social BPM", "BPM Maturity", "BPM in Healthcare", and "BPM Security" cannot be tagged easily, because these topics seem orthogonal to the uses cases and key concerns. This explains why broad use cases like *design model* (DecM) and *enact model* (EnM) score relatively high.

The key concerns were identified before tagging the papers [54]. In hindsight there seem to be at least three potentially missing concerns: *process integration*, *patterns*,

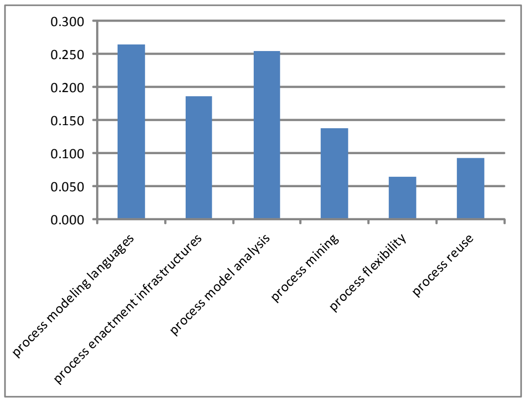

Fig. 27. Average relative importance of concerns (based on Table 2).

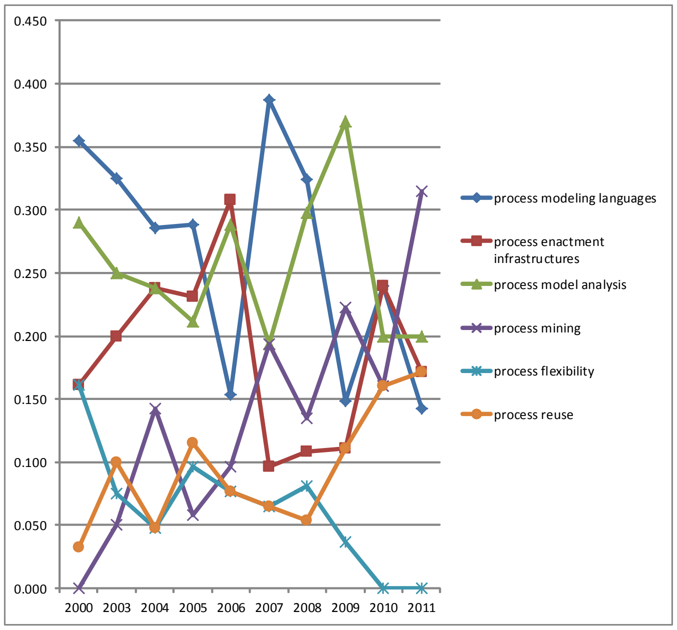

Fig. 28. The importance of each concern plotted over time thus showing changes in the relative attention for each concern over time (based on Table 2).

Table 2. Relative importance of concerns over the years

year	process modeling languages	process enactment infrastructures	process model analysis	process mining	process flexibility	process reuse
2000	0.355	0.161	0.290	0.000	0.161	0.032
2003	0.325	0.200	0.250	0.050	0.075	0.100
2004	0.286	0.238	0.238	0.143	0.048	0.048
2005	0.288	0.231	0.212	0.058	0.096	0.115
2006	0.154	0.308	0.288	0.096	0.077	0.077
2007	0.387	0.097	0.194	0.194	0.065	0.065
2008	0.324	0.108	0.297	0.135	0.081	0.054
2009	0.148	0.111	0.370	0.222	0.037	0.111
2010	0.240	0.240	0.200	0.160	0.000	0.160
2011	0.143	0.171	0.200	0.314	0.000	0.171
average	0.265	0.187	0.254	0.137	0.064	0.093

and *collaboration*. Many papers are concerned with web services and other technologies (e.g., SaaS, PaaS, clouds, and grids) to integrate processes. These are now tagged as *process enactment infrastructures* (second concern). In the BPM proceedings there are various papers proposing new patterns collections or evaluating existing languages using the well-known workflow patterns [9, 24]. These are now tagged as *process modeling languages* (first concern). Another recurring concern seems to collaboration, e.g., collaborative modeling or system development.

Given a process, different *perspectives* can be considered: the *control-flow* perspective ("What activities need to be executed and how are they ordered?"), the *organizational* perspective ("What are the organizational roles, which activities can be executed by a particular resource, and how is work distributed?"), the *case/data* perspective ("Which characteristics of a case influence a particular decision?"), and the *time* perspective ("What are the bottlenecks in my process?"), etc. The use cases and key concerns are neutral/orthogonal with respect to these perspectives. Although most papers focus on the control-flow perspective, there are several papers that focus on the organizational perspective, e.g., papers dealing with optimal resource allocations or role-based access control. It would have been useful to add additional tags to papers based on the perspectives considered.

Despite these limitations, Tables 1 and 2 provide a nice overview of developments in the BPM discipline. Comparing papers published in the early BPM proceedings with papers published in more recent BPM proceedings clearly shows that the BPM discipline progressed at a remarkable speed. The understanding of process modeling languages improved and analysis techniques have become much more powerful.

6 Outlook

Over the last decade there has been a growing interest in Business Process Management (BPM). Practitioners have been using BPM technologies to model, improve, and enact business processes. Today, a plethora of BPM systems and tools is available. Academics have been developing new techniques and approaches to support more advanced forms of BPM. This survey describes the state-of-the-art in BPM. The BPM discipline has been structured in various ways and developments have been put in their historic context. The core of the survey is based on a set of twenty BPM use cases and six BPM key concerns. The use cases show "how, where, and when" BPM techniques can be used. The six key concerns highlight important research areas within the BPM discipline. Table 3 relates the BPM use cases and BPM key concerns. As shown, the six key concerns cover the twenty use cases well.

The BPM discipline has developed at an amazing speed. However, a careful analysis of BPM literature also reveals some weaknesses.

Many papers introduce a new modeling language. The need for such new languages is often unclear, and, in many cases, the proposed language is never used again after publication. A related problem is that many papers spend more time on presenting the context of the problem rather than the actual analysis and solution. For example, there are papers proposing a new verification technique for a language introduced in the same paper. Consequently, the results cannot be used or compared easily.

Many papers cannot be linked to one of the twenty use cases in a straightforward manner. Authors seem to focus on originality rather than relevance and show little concern for real-life use cases. One could argue that some of these papers propose solutions for rather exotic or even non-existing problems.

Our use-case based analysis of existing literature shows that various use cases are neglected by both BPM researchers and BPM software. For example, use cases related to improving the performance of processes seem to be neglected. It is remarkable that there are hardly any tools that provide suggestions for redesigning processes. Simulation tools just provide "what-if" analysis without suggesting better alternatives. Moreover, business "intelligence" tools do not use event data to suggest better process designs. The active classification of tools and publications using the use cases may simulate academics and practitioners to focus on process improvement scenarios.

Many papers describe implementation efforts; however, frequently the software is not available for the reader. Moreover, regrettably, many of the research prototypes seem to "disappear" after publication. As a result, research efforts get lost.

Many papers include case studies, e.g., to test a new technique or system, which is good. Unfortunately, most case studies seem rather artificial. Often the core contribution of the paper is not really evaluated or the case study is deliberately kept vague.

To address the weaknesses just mentioned, authors and tool developers are encouraged to clearly state which of the BPM use cases their results (algorithms, procedures, tools, etc.) aim to support. The twenty use cases presented in this paper can serve as the starting point for a commonly agreed-upon taxonomy of BPM use cases. The current use cases could be subdivided in more specific ones. Such a structuring would hopefully result in collections of benchmark problems, comparable to the datasets used in

Table 3. Relation between the twenty use cases and six key concerns (+ = related and ++ = strongly related).

use case	process modeling languages	process enactment infrastructures	process model analysis	process mining	process flexibility	process reuse
design model (DesM)	++		+		+	
discover model from event data (DiscM)				++		
select model from collection (SelM)						++
merge models (MerM)						++
compose model (CompM)						+
design configurable model (DesCM)	+					++
merge models into configurable model (MerCM)						++
configure configurable model (ConCM)						++
refine model (RefM)	+	+				+
enact model (EnM)	+	++			+	
log event data (LogED)		+		++		
monitor (Mon)		+		+		
adapt while running (AdaWR)		+			++	
analyze performance based on model (PerfM)			++			
verify model (VerM)			++			+
check conformance using event data (ConfED)				++		
analyze performance using event data (PerfED)				++		
repair model (RepM)				++		
extend model (ExtM)				++		
improve model (ImpM)			++	+		

data mining and model checking competitions. Practitioners and academics are encouraged to share open-source software and data sets (collections of process models, event logs, etc.). Currently, many prototypes are developed from scratch and "fade onto oblivion" when the corresponding research project ends. Moreover, it is often impossible to compare different approaches in a fair manner as experiments are incomparable or cannot be reproduced. Given the importance of BPM, these weaknesses need to be tackled urgently. This survey is a modest attempt to guide BPM research towards the real key challenges in our field.

References

1. W.M.P. van der Aalst. Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 1–65. Springer-Verlag, Berlin, 2004.
2. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, Berlin, 2007.
3. W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske. Business Process Management: A Survey. In W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, editors, *International Conference on Business Process Management (BPM 2003)*, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, Berlin, 2003.
4. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2004.
5. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
6. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
7. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
8. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin, 2011.
9. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
10. M. Zur Muehlen and J. Recker. How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In Z. Bellahsene and M. Léonard, editors, *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08)*, volume 5074 of *Lecture Notes in Computer Science*, pages 465–479. Springer-Verlag, Berlin, 2008.
11. E.F. Codd. A Relational Model for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, June 1970.
12. P.P. Chen. The Entity-Relationship Model: Towards a unified view of Data. *ACM Transactions on Database Systems*, 1:9–36, Jan 1976.
13. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
14. C.A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, pages 225–240, Boulder, Colorado, 1979. ACM Press.

15. C.A. Ellis and G. Nutt. Workflow: The Process Spectrum. In A. Sheth, editor, *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems*, pages 140–145, Athens, Georgia, May 1996.
16. C.A. Ellis and G.J. Nutt. Office Information Systems and Computer Science. *ACM Computing Surveys*, 12(1):27–60, 1980.
17. A.W. Holt. Coordination Technology and Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 278–296. Springer-Verlag, Berlin, 1985.
18. A.W. Holt, H. Saint, R. Shapiro, and S. Warshall. Final Report on the Information Systems Theory Project. Technical Report RADC-TR-68-305, Griffiss Air Force Base, New York, 1968.
19. M. zur Muehlen. *Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems*. Logos, Berlin, 2004.
20. M.D. Zisman. *Representation, Specification and Automation of Office Procedures*. PhD thesis, University of Pennsylvania, Warton School of Business, 1977.
21. M.D. Zisman. Office Automation: Revolution or Evolution. *Sloan Management Review*, 13(3):1–16, 1978.
22. M.D. Zisman. Use of Production Systems for Modeling Asynchronous Concurrent Processes. *Pattern-Directed Inference Systems*, pages 53–68, 1978.
23. C.A. Ellis and G.J. Nutt. *Computer Science and Office Information Systems*. Xerox, Palo Alto Research Center, 1979.
24. N. Russell, W.M.P.van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In O. Pastor and J. Falcao e Cunha, editors, *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, volume 3520 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, Berlin, 2005.
25. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. In L. Delcambre, C. Kop, H.C. Mayr, J. Mylopoulos, and O. Pastor, editors, *24nd International Conference on Conceptual Modeling (ER 2005)*, volume 3716 of *Lecture Notes in Computer Science*, pages 353–368. Springer-Verlag, Berlin, 2005.
26. W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, editors. *International Conference on Business Process Management (BPM 2003)*, volume 2678 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2003.
27. J. Desel, B. Pernici, and M. Weske, editors. *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2004.
28. W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors. *International Conference on Business Process Management (BPM 2005)*, volume 3649 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2005.
29. S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors. *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2006.
30. G. Alonso, P. Dadam, and M. Rosemann, editors. *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2007.
31. M. Dumas, M. Reichert, and M.C. Shan, editors. *International Conference on Business Process Management (BPM 2008)*, volume 5240 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2008.

32. U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors. *International Conference on Business Process Management (BPM 2009)*, volume 5701 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2009.
33. R. Hull, J. Mendling, and S. Tai, editors. *International Conference on Business Process Management (BPM 2010)*, volume 6336 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2010.
34. S. Rinderle, F. Toumani, and K. Wolf, editors. *International Conference on Business Process Management (BPM 2011)*, volume 6896 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2011.
35. A. Barros, A. Gal, and E. Kindler, editors. *International Conference on Business Process Management (BPM 2012)*, volume 7481 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2012.
36. 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.
37. H. Smith and P. Fingar. *Business Process Management: The Third Wave*. Meghan Kiffer Press, 2006.
38. A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell. *Modern Business Process Automation: YAWL and its Support Environment*. Springer-Verlag, Berlin, 2010.
39. M. Reichert and B. Weber. *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Springer-Verlag, Berlin, 2012.
40. D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.
41. R. Medina-Mora, T. Winograd, R. Flores, and F. Floris. The Action Workflow Approach to Workflow Management Technology. In M. Mantel and R. Baecker, editors, *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work (CSCW'92)*, pages 281–288. ACM, New York, 1992.
42. B. Ludaescher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.
43. J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers. Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute, 2011.
44. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
45. F. Gottschalk, W.M.P. van der Aalst, and H.M. Jansen-Vullers. Configurable Process Models: A Foundational Approach. In J. Becker and P. Delfmann, editors, *Reference Modeling: Efficient Information Systems Design Through Reuse of Information Models*, pages 59–78. Physica-Verlag, Springer, Heidelberg, Germany, 2007.
46. M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1):1–23, 2007.
47. F. Gottschalk, W.M.P. van der Aalst, M.H Jansen-Vullers, and M. La Rosa. Configurable Workflow Models. *International Journal of Cooperative Information Systems*, 17(2):177–221, 2008.
48. T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.
49. J. Becker, P. Delfmann, and R. Knackstedt. Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In J. Becker and

P. Delfmann, editors, *Reference Modeling: Efficient Information Systems Design Through Reuse of Information Models*, pages 27–58. Physica-Verlag, Springer, Heidelberg, Germany, 2007.

50. P. Fettke and P. Loos. Classification of Reference Models - A Methodology and its Application. *Information Systems and e-Business Management*, 1(1):35–53, 2003.
51. W.M.P. van der Aalst, N. Lohmann, and M. La Rosa. Ensuring Correctness During Process Configuration Via Partner Synthesis. *Information Systems*, 37(6):574–592, 2012.
52. W.M.P. van der Aalst, K.M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, and M.T. Wynn. Soundness of Workflow Nets: Classification, Decidability, and Analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.
53. D. Fahland and W.M.P. van der Aalst. Repairing Process Models to Reflect Reality. In A. Barros, A. Gal, and E. Kindler, editors, *International Conference on Business Process Management (BPM 2012)*, volume 7481 of *Lecture Notes in Computer Science*, pages 229–245. Springer-Verlag, Berlin, 2012.
54. W.M.P. van der Aalst. A Decade of Business Process Management Conferences: Personal Reflections on a Developing Discipline. In A. Barros, A. Gal, and E. Kindler, editors, *International Conference on Business Process Management (BPM 2012)*, volume 7481 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 2012.
55. D. Fahland, C. Favre, B. Jobstmann, J. Koehler, N. Lohmann, H. Völzer, and K. Wolf. Instantaneous Soundness Checking of Industrial Business Process Models. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors, *Business Process Management (BPM 2009)*, volume 5701 of *Lecture Notes in Computer Science*, pages 278–293. Springer-Verlag, Berlin, 2009.
56. R. Dijkman, M. Dumas, and L. Garcia-Banuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors, *Business Process Management (BPM 2009)*, volume 5701 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, Berlin, 2009.
57. S.A. White et al. Business Process Modeling Notation (BPML), Version 1.0, 2004.
58. OMG. Business Process Model and Notation (BPMN). Object Management Group, formal/2011-01-03, 2011.
59. G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.
60. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
61. A.W. Scheer. *Business Process Engineering: Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.
62. A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guzar, N. Kartha, C.K. Liu, R. Khalaf, Dieter Koenig, M. Marin, V. Mehta, S. Thatte, D. Rijn, P. Yendluri, and A. Yiu. Web Services Business Process Execution Language Version 2.0 (OASIS Standard). WS-BPEL TC OASIS, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, 2007.
63. C. Ouyang, W.M.P. van der Aalst, S. Breutel, M. Dumas, A.H.M. ter Hofstede, and H.M.W. Verbeek. Formal Semantics and Analysis of Control Flow in WS-BPEL. *Science of Computer Programming*, 67(2-3):162–198, 2007.
64. C. Ouyang, M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede, and J. Mendling. From Business Process Models to Process-oriented Software Systems. *ACM Transactions on Software Engineering and Methodology*, 19(1):1–37, 2009.
65. W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.

66. E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. *Data and Knowledge Engineering*, 56(1):23–40, 2006.
67. H. Völzer. A New Semantics for the Inclusive Converging Gateway in Safe Processes. In R. Hull, J. Mendling, and S. Tai, editors, *Business Process Management (BPM 2010)*, volume 6336 of *Lecture Notes in Computer Science*, pages 294–309. Springer-Verlag, Berlin, 2010.
68. M.T. Wynn, H.M.W. Verbeek, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Business Process Verification: Finally a Reality! *Business Process Management Journal*, 15(1):74–92, 2009.
69. W.M.P. van der Aalst, J. Desel, and E. Kindler. On the Semantics of EPCs: A Vicious Circle. In M. Nüttgens and F.J. Rump, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, November 2002. Gesellschaft für Informatik, Bonn.
70. W.M.P. van der Aalst, M. Pesic, and H. Schonenberg. Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - Research and Development*, 23(2):99–113, 2009.
71. M. Montali, M. Pesic, W.M.P. van der Aalst, F. Chesani, P. Mello, and S. Storari. Declarative Specification and Verification of Service Choreographies. *ACM Transactions on the Web*, 4(1):1–62, 2010.
72. E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts and London, UK, 1999.
73. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1991.
74. W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *International Journal of Cooperative Information Systems*, 10(4):443–482, 2001.
75. ACSI. Artifact-Centric Service Interoperation (ACSI) Project Home Page. `www.acsi-project.eu`.
76. K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su. Towards Formal Analysis of Artifact-Centric Business Process Models. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 288–304. Springer-Verlag, Berlin, 2007.
77. D. Cohn and R. Hull. Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. *IEEE Data Engineering Bulletin*, 32(3):3–9, 2009.
78. D. Fahland, M. De Leoni, B. van Dongen, and W.M.P. van der Aalst. Many-to-Many: Some Observations on Interactions in Artifact Choreographies. In D. Eichhorn, A. Koschmider, and H. Zhang, editors, *Proceedings of the 3rd Central-European Workshop on Services and their Composition (ZEUS 2011)*, CEUR Workshop Proceedings. CEUR-WS.org, 2011.
79. N. Lohmann. Compliance by Design for Artifact-Centric Business Processes. In S. Rinderle, F. Toumani, and K. Wolf, editors, *Business Process Management (BPM 2011)*, volume 6896 of *Lecture Notes in Computer Science*, pages 99–115. Springer-Verlag, Berlin, 2011.
80. A. Nigam and N.S. Caswell. Business artifacts: An Approach to Operational Specification. *IBM Systems Journal*, 42(3):428–445, 2003.
81. J. Mendling, H. Reijers, and W.M.P. van der Aalst. Seven Process Modeling Guidelines (7PMG). *Information and Software Technology*, 52(2):127–136, 2010.
82. M. La Rosa, A.H.M. ter Hofstede, P. Wohed, H.A. Reijers, J. Mendling, and W.M.P. van der Aalst. Managing Process Model Complexity via Concrete Syntax Modifications. *IEEE Transactions on Industrial Informatics*, 7(2):255–265, 2011.

83. M. La Rosa, P. Wohed, J. Mendling, A.H.M. ter Hofstede, H.A. Reijers, and W.M.P. van der Aalst. Managing Process Model Complexity via Abstract Syntax Modifications. *IEEE Transactions on Industrial Informatics*, 7(4):614–629, 2011.
84. A.A. Abdul, G.K.T. Wei, G.M. Muketha, and W.P. Wen. Complexity Metrics for Measuring the Understandability and Maintainability of Business Process Models using Goal-Question-Metric (GQM). *International Journal of Computer Science and Network Security*, 8(5):219–225, 2008.
85. K.B. Lassen and W.M.P. van der Aalst. Complexity Metrics for Workflow Nets. *Information and Software Technology*, 51(3):610–626, 2009.
86. J. Mendling, H.A. Reijers, and J. Cardoso. What Makes Process Models Understandable? In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, Berlin, 2007.
87. J. Recker, M. zur Muehlen, K. Siau, J. Erickson, and M. Indulska. Measuring Method Complexity: UML versus BPMN. In *Americas Conference on Information Systems (AMCIS 2009)*, pages 1–12. AIS, 2009.
88. A. Streit, B. Pham, and R. Brown. Visualisation Support for Managing Large Business Process Specifications. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *International Conference on Business Process Management (BPM 2005)*, volume 3649 of *Lecture Notes in Computer Science*, pages 205—219. Springer-Verlag, Berlin, 2005.
89. B. Weber, M. Reichert, J. Mendling, and H.A. Reijers. Refactoring Large Process Model Repositories. *Computers in Industry*, 62(5):467–486, 2011.
90. V. Gruhn and R. Laue. Reducing the Cognitive Complexity of Business Process Models. In G. Baciu, Y. Wang, Y. Yao, W. Kinsner, K. Chan, and L. Zadeh, editors, *IEEE Conference on Cognitive Informatics (ICCI 2009)*, pages 339–345, 2009.
91. Workflow Patterns Home Page. http://www.workflowpatterns.com.
92. N. Russell, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Workflow Exception Patterns. In E. Dubois and K. Pohl, editors, *Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06)*, volume 4001 of *Lecture Notes in Computer Science*, pages 288–302. Springer-Verlag, Berlin, 2006.
93. A. Barros, M. Dumas, and A. ter Hofstede. Service Interaction Patterns. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *International Conference on Business Process Management (BPM 2005)*, volume 3649 of *Lecture Notes in Computer Science*, pages 302–318. Springer-Verlag, Berlin, 2005.
94. B. Weber, M. Reichert, and S. Rinderle-Ma. Change Patterns and Change Support Features: Enhancing Flexibility in Process-Aware Information Systems. *Data and Knowledge Engineering*, 66(3):438–466, 2008.
95. L. Fischer, editor. *Workflow Handbook 2003, Workflow Management Coalition*. Future Strategies, Lighthouse Point, Florida, 2003.
96. P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.
97. WFMC. Workflow Management Coalition Workflow Standard: Interface 1 – Process Definition Interchange Process Model (WFMC-TC-1016). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 1999.
98. 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.
99. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Lan-

guage for Web Services, Version 1.1. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.

100. F. Leymann. Web Services Flow Language, Version 1.0, 2001.
101. S. Thatte. XLANG Web Services for Business Process Design, 2001.
102. W.M.P. van der Aalst. Don't Go With The Flow: Web Services Composition Standards Exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003.
103. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services Concepts, Architectures and Applications*. Springer-Verlag, Berlin, 2004.
104. B. Benatallah, F. Casati, and F. Toumani. Representing, Analysing and Managing Web Service Protocols. *Data and Knowledge Engineering*, 58(3):327–357, 2006.
105. F. Casati, E. Shan, U. Dayal, and M.C. Shan. Business-Oriented Management of Web Services. *Communications of the ACM*, 46(10):55–60, 2003.
106. M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: A Research Roadmap. *International Journal of Cooperative Information Systems*, 17(2):223–255, 2008.
107. L.J. Zhang, J. Zhang, and H. Cai. *Services Computing, Core Enabling Technology of the Modern Services Industry*. Springer-Verlag, Berlin, 2007.
108. G. Hohpe and B. Woolf. *Enterprise Integration Patterns*. Addison-Wesley Professional, Reading, MA, 2003.
109. G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Günthör, and C. Mohan. Advanced Transaction Models in Workflow Contexts. In *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*. IEEE Computer Society, 1996.
110. D. Kuo, M. Lawley, C. Liu, and M.E. Orlowska. A General Model for Nested Transactional Workflows. In *Proceedings of the International Workshop on Advanced Transaction Models and Architecture (ATMA '96)*, pages 18–35, Bombay, India, 1996.
111. A. Reuter and F. Schwenkreis. ConTracts - A Low-Level Mechanism for Building General-Purpose Workflow Management-Systems. *Data Engineering Bulletin*, 18(1):4–10, 1995.
112. G. Vossen. Transactional Workflows (tutorial). In F. Bry, R. Ramakrishnan, and K. Ramamohanarao, editors, *Proceedings of the 5th International Conference on Deductive and Object-Oriented Databases (DOOD'97)*, volume 1341 of *Lecture Notes in Computer Science*, pages 20–25. Springer-Verlag, Berlin, 1997.
113. G. Weikum and G. Vossen. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann Publishers, San Francisco, CA, 2002.
114. E. Bertino, E. Ferrari, and V. Atluri. The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. *ACM Transactions on Information and System Security*, 22(1):65–104, 1999.
115. D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.
116. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
117. W.M.P. van der Aalst. Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries. *Information and Management*, 37(2):67–75, March 2000.
118. W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques. In 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*, pages 161–183. Springer-Verlag, Berlin, 2000.

119. K.M. van Hee, N. Sidorova, and M. Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In W.M.P. van der Aalst and E. Best, editors, *Application and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 335–354. Springer-Verlag, Berlin, 2003.
120. K.M. van Hee, N. Sidorova, and M. Voorhoeve. Generalised Soundness of Workflow Nets Is Decidable. In J. Cortadella and W. Reisig, editors, *Application and Theory of Petri Nets 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 197–215. Springer-Verlag, Berlin, 2004.
121. W.M.P. van der Aalst and A.H.M. ter Hofstede. Verification of Workflow Task Structures: A Petri-net-based Approach. *Information Systems*, 25(1):43–69, 2000.
122. A. Basu and R.W. Blanning. A Formal Approach to Workflow Analysis. *Information Systems Research*, 11(1):17–36, 2000.
123. H.H. Bi and J.L. Zhao. Applying Propositional Logic to Workflow Verification. *Information Technology and Management*, 5(3-4):293–318, 2004.
124. Y. Choi and J. Zhao. Decomposition-based Verification of Cyclic workflows. In D.A. Peled and Y-K. Tsay, editors, *Proceedings of Automated Technology for Verification and Analysis (ATVA 2005)*, volume 3707 of *Lecture Notes in Computer Science*, pages 84–98, Taipei, Taiwan, 2005. Springer-Verlag.
125. R. Eshuis. Symbolic Model Checking of UML Activity Diagrams. *ACM Transactions on Software Engineering Methodology*, 15(1):1–38, 2006.
126. D. Fahland, C. Favre, J. Koehler, N. Lohmann, H. Völzer, and K. Wolf. Analysis on Demand: Instantaneous Soundness Checking of Industrial Business Process Models. *Data and Knowledge Engineering*, 70(5):448–466, 2011.
127. S. Fan, W.C. Dou, and J. Chen. Dual Workflow Nets: Mixed Control/Data-Flow Representation for Workflow Modeling and Verification. In *Advances in Web and Network Technologies, and Information Management (APWeb/WAIM 2007 Workshops)*, volume 4537 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, Berlin, 2007.
128. X. Fu, T. Bultan, and J. Su. Formal Verification of e-Services and Workflows. In C. Bussler, R. Hull, S. McIlraith, M. Orlowska, B. Pernici, and J. Yang, editors, *Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop (WES 2002)*, volume 2512 of *Lecture Notes in Computer Science*, pages 188–202. Springer-Verlag, Berlin, 2002.
129. A.H.M. ter Hofstede, M.E. Orlowska, and J. Rajapakse. Verification Problems in Conceptual Workflow Specifications. *Data and Knowledge Engineering*, 24(3):239–256, 1998.
130. B. Kiepuszewski, A.H.M. ter Hofstede, and W.M.P. van der Aalst. Fundamentals of Control Flow in Workflows. *Acta Informatica*, 39(3):143–209, 2003.
131. W. Sadiq and M.E. Orlowska. Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models. In M. Jarke and A. Oberweis, editors, *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99)*, volume 1626 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, Berlin, 1999.
132. R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.
133. R. Milner. *Communication and Concurrency*. Prentice-Hall, Inc., 1989.
134. W.M.P. van der Aalst. Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.
135. W.M.P. van der Aalst. Inheritance of Interorganizational Workflows: How to Agree to Disagree Without Loosing Control? *Information Technology and Management Journal*, 4(4):345–389, 2003.

136. W.M.P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K. Wolf. From Public Views to Private Views: Correctness-by-Design for Services. In M. Dumas and H. Heckel, editors, *Proceedings of the 4th International Workshop on Web Services and Formal Methods (WS-FM 2007)*, volume 4937 of *Lecture Notes in Computer Science*, pages 139–153. Springer-Verlag, Berlin, 2008.
137. W.M.P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K. Wolf. Multiparty Contracts: Agreeing and Implementing Interorganizational Processes. *The Computer Journal*, 53(1):90–106, 2010.
138. J.A. Fisteus, L.S. Fernández, and C.D. Kloos. Formal Verification of BPEL4WS Business Collaborations. In K. Bauknecht, M. Bichler, and B. Proll, editors, *Proceedings of the 5th International Conference on Electronic Commerce and Web Technologies (EC-Web '04)*, volume 3182 of *Lecture Notes in Computer Science*, pages 79–94, Zaragoza, Spain, August 2004. Springer-Verlag, Berlin.
139. H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based Verification of Web Service Composition. In *Proceedings of 18th IEEE International Conference on Automated Software Engineering (ASE)*, pages 152–161, Montreal, Canada, October 2003.
140. X. Fu, T. Bultan, and J. Su. WSAT: A Tool for Formal Analysis of Web Services. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV)*, volume 3114 of *Lecture Notes in Computer Science*, pages 510–514. Springer-Verlag, Berlin, 2004.
141. E. Kindler, A. Martens, and W. Reisig. Inter-Operability of Workflow Applications: Local Criteria for Global Soundness. In 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*, pages 235–253. Springer-Verlag, Berlin, 2000.
142. M. Koshkina and F. van Breugel. Verification of Business Processes for Web Services. Technical report CS-2003-11, York University, October 2003. Available from: http://www.cs.yorku.ca/techreports/2003/.
143. N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg. Analyzing Interacting BPEL Processes. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, Berlin, 2006.
144. N. Lohmann, P. Massuthe, and K. Wolf. Operating Guidelines for Finite-State Services. In J. Kleijn and A. Yakovlev, editors, *28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007, Siedlce, Poland, June 25-29, 2007, Proceedings*, volume 4546 of *Lecture Notes in Computer Science*, pages 321–341. Springer-Verlag, Berlin, 2007.
145. F. Gottschalk, T. Wagemakers, M.H. Jansen-Vullers, W.M.P. van der Aalst, and M. La Rosa. Configurable Process Models: Experiences From a Municipality Case Study. In P. van Eck, J. Gordijn, and R. Wieringa, editors, *Advanced Information Systems Engineering, Proceedings of the 21st International Conference on Advanced Information Systems Engineering (CAiSE'09)*, volume 5565 of *Lecture Notes in Computer Science*, pages 486–500. Springer-Verlag, Berlin, 2009.
146. M. La Rosa, M. Dumas, A. ter Hofstede, and J. Mendling. Configurable Multi-Perspective Business Process Models. *Information Systems*, 36(2):313–340, 2011.
147. A. Schnieders and F. Puhlmann. Variability Mechanisms in E-Business Process Families. In W. Abramowicz and H.C. Mayr, editors, *Proceedings of the 9th International Conference on Business Information Systems (BIS'06)*, volume 85 of *LNI*, pages 583–601. GI, 2006.
148. W.M.P. van der Aalst, M. Dumas, F. Gottschalk, A.H.M. ter Hofstede, M. La Rosa, and J. Mendling. Preserving Correctness During Business Process Model Configuration. *Formal Aspects of Computing*, 22(3):459–482, 2010.
149. H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001.

150. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
151. N. Lohmann and D. Weinberg. Wendy: A Tool to Synthesize Partners for Services. In J. Lilius and W. Penczek, editors, *Applications and Theory of Petri Nets 2010*, volume 6128 of *Lecture Notes in Computer Science*, pages 279–307. Springer-Verlag, Berlin, 2010.
152. B.D. Clinton and A. van der Merwe. Management Accounting: Approaches, Techniques, and Management Processes. *Cost Management*, 20(3):14–22, 2006.
153. J.A. Buzacott. Commonalities in Reengineered Business Processes: Models and Issues. *Management Science*, 42(5):768–782, 1996.
154. J.J. Moder and S.E. Elmaghraby. *Handbook of Operations Research: Foundations and Fundamentals*. Van Nostrand Reinhold,New York, 1978.
155. H. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*, volume 2617 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2003.
156. R. Wild. *Production and Operations Management : Principles and Techniques*. Cassell, London, 1989.
157. W.M.P. van der Aalst. Business Process Simulation Revisited. In J. Barjis, editor, *Enterprise and Organizational Modeling and Simulation*, volume 63 of *Lecture Notes in Business Information Processing*, pages 1–14. Springer-Verlag, Berlin, 2010.
158. W.M.P. van der Aalst, J. Nakatumba, A. Rozinat, and N. Russell. Business Process Simulation. In J. vom Brocke and M. Rosemann, editors, *Handbook on Business Process Management*, International Handbooks on Information Systems, pages 313–338. Springer-Verlag, Berlin, 2010.
159. IEEE Task Force on Process Mining. Process Mining Manifesto. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. Springer-Verlag, Berlin, 2012.
160. W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business Process Mining: An Industrial Application. *Information Systems*, 32(5):713–732, 2007.
161. S. Davidson, S. Cohen-Boulakia, A. Eyal, B. Ludaescher, T. McPhillips, S. Bowers, M. Anand, and J. Freire. Provenance in Scientific Workflow Systems. *Data Engineering Bulletin*, 30(4):44–50, 2007.
162. F. Curbera, Y. Doganata, A. Martens, N. Mukhi, and A. Slominski. Business Provenance: A Technology to Increase Traceability of End-to-End Operations. In R. Meersman and Z. Tari, editors, *Proceedings of the 16th International Conference on Cooperative Information Systems, CoopIS 2008, OTM 2008, Part I*, volume 5331 of *Lecture Notes in Computer Science*, pages 100–119. Springer-Verlag, Berlin, 2008.
163. B.F. van Dongen, A.K. Alves de Medeiros, and L. Wenn. Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. In K. Jensen and W.M.P. van der Aalst, editors, *Transactions on Petri Nets and Other Models of Concurrency II*, volume 5460 of *Lecture Notes in Computer Science*, pages 225–242. Springer-Verlag, Berlin, 2009.
164. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, volume 1377 of *Lecture Notes in Computer Science*, pages 469–483. Springer-Verlag, Berlin, 1998.
165. 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.
166. A. Datta. Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research*, 9(3):275–301, 1998.
167. B.F. van Dongen and W.M.P. van der Aalst. Multi-Phase Process Mining: Building Instance Graphs. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T.W. Ling, editors, *International Con-*

ference on Conceptual Modeling (ER 2004), volume 3288 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, Berlin, 2004.

168. B.F. van Dongen and W.M.P. van der Aalst. Multi-Phase Mining: Aggregating Instances Graphs into EPCs and Petri Nets. In D. Marinescu, editor, *Proceedings of the Second International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management*, pages 35–58. Florida International University, Miami, Florida, USA, 2005.
169. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
170. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
171. 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.
172. A. Rozinat and W.M.P. van der Aalst. Decision Mining in ProM. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*, pages 420–425. Springer-Verlag, Berlin, 2006.
173. A. Rozinat, M. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C. Fidge. Workflow Simulation for Operational Decision Support. *Data and Knowledge Engineering*, 68(9):834–850, 2009.
174. A. Rozinat, R.S. Mans, M. Song, and W.M.P. van der Aalst. Discovering Colored Petri Nets From Event Logs. *International Journal on Software Tools for Technology Transfer*, 10(1):57–74, 2008.
175. A. Rozinat, R.S. Mans, M. Song, and W.M.P. van der Aalst. Discovering Simulation Models. *Information Systems*, 34(3):305–327, 2009.
176. W.M.P. van der Aalst, M. Pesic, and M. Song. Beyond Process Mining: From the Past to Present and Future. In B. Pernici, editor, *Advanced Information Systems Engineering, Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10)*, volume 6051 of *Lecture Notes in Computer Science*, pages 38–52. Springer-Verlag, Berlin, 2010.
177. F.M. Maggi, M. Montali, and W.M.P. van der Aalst. An Operational Decision Support Framework for Monitoring Business Constraints. In J. de Lara and A. Zisman, editors, *International Conference on Fundamental Approaches to Software Engineering (FASE 2012)*, volume 7212 of *Lecture Notes in Computer Science*, pages 146–162. Springer-Verlag, Berlin, 2012.
178. F.M. Maggi, M. Westergaard, M. Montali, and W.M.P. van der Aalst. Runtime Verification of LTL-Based Declarative Process Models. In S. Khurshid and K. Sen, editors, *Runtime Verification (RV 2011)*, volume 7186 of *Lecture Notes in Computer Science*, pages 131–146. Springer-Verlag, Berlin, 2012.
179. W.M.P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *WIREs Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
180. A. Adriansyah, B. van Dongen, and W.M.P. van der Aalst. Conformance Checking using Cost-Based Fitness Analysis. In C.H. Chi and P. Johnson, editors, *IEEE International Enterprise Computing Conference (EDOC 2011)*, pages 55–64. IEEE Computer Society, 2011.

181. A. Adriansyah, B.F. van Dongen, and W.M.P. van der Aalst. Towards Robust Conformance Checking. In M. zur Muehlen and J. Su, editors, *BPM 2010 Workshops, Proceedings of the Sixth Workshop on Business Process Intelligence (BPI2010)*, volume 66 of *Lecture Notes in Business Information Processing*, pages 122–133. Springer-Verlag, Berlin, 2011.
182. T. Calders, C. Guenther, M. Pechenizkiy, and A. Rozinat. Using Minimum Description Length for Process Mining. In *ACM Symposium on Applied Computing (SAC 2009)*, pages 1451–1455. ACM Press, 2009.
183. 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.
184. S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens. Robust Process Discovery with Artificial Negative Events. *Journal of Machine Learning Research*, 10:1305–1340, 2009.
185. J. Munoz-Gama and J. Carmona. A Fresh Look at Precision in Process Conformance. In R. Hull, J. Mendling, and S. Tai, editors, *Business Process Management (BPM 2010)*, volume 6336 of *Lecture Notes in Computer Science*, pages 211–226. Springer-Verlag, Berlin, 2010.
186. J. Munoz-Gama and J. Carmona. Enhancing Precision in Process Conformance: Stability, Confidence and Severity. In N. Chawla, I. King, and A. Sperduti, editors, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, Paris, France, April 2011. IEEE.
187. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.
188. J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens. A Robust F-measure for Evaluating Discovered Process Models. In N. Chawla, I. King, and A. Sperduti, editors, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pages 148–155, Paris, France, April 2011. IEEE.
189. R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling. Similarity of Business Process Models: Metrics and Evaluation. *Information Systems*, 36(2):498–516, 2011.
190. T. Jin, J. Wang, and L. Wen. Efficient Retrieval of Similar Workflow Models Based on Structure. In *OTM 2011*, volume 7044 of *Lecture Notes in Computer Science*, pages 56–63. Springer-Verlag, Berlin, 2011.
191. T. Jin, J. Wang, and L. Wen. Efficient Retrieval of Similar Workflow Models Based on Behavior. In *APWeb 2012*, volume 7235 of *Lecture Notes in Computer Science*, pages 677–684. Springer-Verlag, Berlin, 2012.
192. J. Mendling, B.F. van Dongen, and W.M.P. van der Aalst. On the Degree of Behavioral Similarity between Business Process Models. In M. Nuettgens, F.J. Rump, and A. Gadatsch, editors, *Proceedings of Sixth Workshop on Event-Driven Process Chains (WI-EPK 2007)*, pages 39–58, St. Augustin, November 2007. Gesellschaft für Informatik, Bonn.
193. M. Weidlich, R. Dijkman, and M. Weske. Behavioral Equivalence and Compatibility of Business Process Models with Complex Correspondences. *Computer Journal*, 2012.
194. 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.
195. W.M.P. van der Aalst, M.H. Schonenberg, and M. Song. Time Prediction Based on Process Mining. *Information Systems*, 36(2):450–475, 2011.
196. B.F. van Dongen, R.A. Crooy, and W.M.P. van der Aalst. Cycle Time Prediction: When Will This Case Finally Be Finished? In R. Meersman and Z. Tari, editors, *Proceedings of the 16th International Conference on Cooperative Information Systems, CoopIS 2008, OTM 2008, Part I*, volume 5331 of *Lecture Notes in Computer Science*, pages 319–336. Springer-Verlag, Berlin, 2008.

197. H.A. Reijers. Case Prediction in BPM Systems: A Research Challenge. *Journal of the Korean Institute of Industrial Engineers*, 33:1–10, 2006.
198. Staffware. *Staffware Process Suite Version 2 – White Paper*. Staffware PLC, Maidenhead, UK, 2003.
199. H. Schonenberg, B. Weber, B.F. van Dongen, and W.M.P. van der Aalst. Supporting Flexible Processes Through Recommendations Based on History. In M. Dumas, M. Reichert, and M.C. Shan, editors, *International Conference on Business Process Management (BPM 2008)*, volume 5240 of *Lecture Notes in Computer Science*, pages 51–66. Springer-Verlag, Berlin, 2008.
200. W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.
201. M. Adams, A.H.M. ter Hofstede, W.M.P. van der Aalst, and D. Edmond. Dynamic, Extensible and Context-Aware Exception Handling for Workflows. In F. Curbera, F. Leymann, and M. Weske, editors, *Proceedings of the OTM Conference on Cooperative information Systems (CoopIS 2007)*, volume 4803 of *Lecture Notes in Computer Science*, pages 95–112. Springer-Verlag, Berlin, 2007.
202. S. Dustdar. Caramba - A Process-Aware Collaboration System Supporting Ad Hoc and Collaborative Processes in Virtual Teams. *Distributed and Parallel Databases*, 15(1):45–66, 2004.
203. C.A. Ellis, K. Keddara, and G. Rozenberg. Dynamic Change within Workflow Systems. In N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan, editors, *Proceedings of the Conference on Organizational Computing Systems*, pages 10 – 21, Milpitas, California, August 1995. ACM SIGOIS, ACM Press, New York.
204. M. Pesic, M. H. Schonenberg, N. Sidorova, and W.M.P. van der Aalst. Constraint-Based Workflow Models: Change Made Easy. In F. Curbera, F. Leymann, and M. Weske, editors, *Proceedings of the OTM Conference on Cooperative information Systems (CoopIS 2007)*, volume 4803 of *Lecture Notes in Computer Science*, pages 77–94. Springer-Verlag, Berlin, 2007.
205. M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
206. S. Rinderle, M. Reichert, and P. Dadam. Correctness Criteria For Dynamic Changes in Workflow Systems: A Survey. *Data and Knowledge Engineering*, 50(1):9–34, 2004.
207. M. Weske. Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. In R. Sprague, editor, *Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Science (HICSS-34)*. IEEE Computer Society Press, Los Alamitos, California, 2001.
208. W.M.P. van der Aalst and S. Jablonski. Dealing with Workflow Change: Identification of Issues and Solutions. *International Journal of Computer Systems, Science, and Engineering*, 15(5):267–276, 2000.
209. H. Schonenberg, R. Mans, N. Russell, N. Mulyar, and W.M.P. van der Aalst. Process Flexibility: A Survey of Contemporary Approaches. In J. Dietz, A. Albani, and J. Barjis, editors, *Advances in Enterprise Engineering I*, volume 10 of *Lecture Notes in Business Information Processing*, pages 16–30. Springer-Verlag, Berlin, 2008.
210. S. Sadiq, W. Sadiq, and M. Orlowska. Pockets of Flexibility in Workflow Specification. In *Proceedings of the 20th International Conference on Conceptual Modeling (ER 2001)*, volume 2224 of *Lecture Notes in Computer Science*, pages 513–526. Springer-Verlag, Berlin, 2001.
211. T. Herrmann, M. Hoffmann, K.U. Loser, and K. Moysich. Semistructured Models are Surprisingly Useful for User-Centered Design. In G. De Michelis, A. Giboin, L. Karsenty, and R. Dieng, editors, *Designing Cooperative Systems (Coop 2000)*, pages 159–174. IOS Press, Amsterdam, 2000.

212. M. Adams. *Facilitating Dynamic Flexibility and Exception Handling for Workflows*. Phd thesis, Queensland University of Technology, 2007.
213. P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, and M. Teschke. A Comprehensive Approach to Flexibility in Workflow Management Systems. In G. Georgakopoulos, W. Prinz, and A.L. Wolf, editors, *Work Activities Coordination and Collaboration (WACC'99)*, pages 79–88, San Francisco, February 1999. ACM press.
214. M. Pesic. *Constraint-based Workflow Management Systems: Shifting Control to Users*. Phd thesis, Eindhoven University of Technology, May 2008.
215. S. Rinderle, M. Reichert, and P. Dadam. Evaluation of Correctness Criteria for Dynamic Workflow Changes. In W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, editors, *International Conference on Business Process Management (BPM 2003)*, volume 2678 of *Lecture Notes in Computer Science*, pages 41–57. Springer-Verlag, Berlin, 2003.
216. C. Houy, P. Fettke, P. Loos, W.M.P. van der Aalst, and J. Krogstie. Business Process Management in the Large. *Business and Information Systems Engineering*, 3(6):385–388, 2011.
217. R. Dijkman, M. La Rosa, and H.A. Reijers. Managing Large Collections of Business Process Models: Current Techniques and Challenges. *Computers in Industry*, 63(2):91–97, 2012.
218. A.W. Scheer. *Business Process Engineering: ARIS-Navigator for Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1995.
219. A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.
220. M.C. Fauvet, M. La Rosa, M. Sadegh, A. Alshareef, R.M. Dijkman, L. Garcia-Banuelos, H.A. Reijers, W.M.P. van der Aalst, M. Dumas, and J. Mendling. Managing Process Model Collections with APROMORE. In P. Maglio, M. Weske, J. Yang, and M. Fantinato, editors, *Proceedings of Service-Oriented Computing (ICSOC 2010)*, volume 6470 of *Lecture Notes in Computer Science*, pages 699–701. Springer-Verlag, Berlin, 2010.
221. M. La Rosa, H.A. Reijers, W.M.P. van der Aalst, R.M. Dijkman, J. Mendling, M. Dumas, and L. Garcia-Banuelos. APROMORE: An Advanced Process Model Repository. *Expert Systems With Applications*, 38(6):7029–7040, 2011.
222. A. Awad, M. Weidlich, and M. Weske. Visually Specifying Compliance Rules and Explaining Their Violations for Business Processes. *Journal of Visual Languages and Computing*, 22(1):30–55, 2011.
223. M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske. Process Compliance Measurement Based on Behavioral Profiles. *Information Systems*, 36(7):1009–1025, 2011.
224. J.J.C.L. Vogelaar, H.M.W. Verbeek, B. Luka, and W.M.P. van der Aalst. Comparing Business Processes to Determine the Feasibility of Configurable Models: A Case Study. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops, International Workshop on Process Model Collections (PMC 2011)*, volume 100 of *Lecture Notes in Business Information Processing*, pages 50–61. Springer-Verlag, Berlin, 2012.
225. F. Gottschalk, W.M.P. van der Aalst, and M.H. Jansen-Vullers. Merging Event-driven Process Chains. In R. Meersman and Z. Tari, editors, *Proceedings of the 16th International Conference on Cooperative Information Systems, CoopIS 2008, OTM 2008, Part I*, volume 5331 of *Lecture Notes in Computer Science*, pages 418–426. Springer-Verlag, Berlin, 2008.
226. M. La Rosa, M. Dumas, R. Uba, and R.M. Dijkman. Merging Business Process Models. In R. Meersman, T. Dillon, and P. Herrero, editors, *International Conference on Cooperative Information Systems (CoopIS 2010)*, volume 6426 of *Lecture Notes in Computer Science*, pages 96–113. Springer-Verlag, Berlin, 2010.
227. M. La Rosa, M. Dumas, R. Uba, and R.M. Dijkman. Business Process Model Merging: An Approach to Business Process Consolidation. *ACM Transactions on Software Engineering and Methodology*, 22(2), 2012.

228. C. Li, M. Reichert, and A. Wombacher. Discovering Reference Models by Mining Process Variants Using a Heuristic Approach. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors, *Business Process Management (BPM 2009)*, volume 5701 of *Lecture Notes in Computer Science*, pages 344–362. Springer-Verlag, Berlin, 2009.
229. C. Li, M. Reichert, and A. Wombacher. The MINADEPT Clustering Approach for Discovering Reference Process Models Out of Process Variants. *International Journal of Cooperative Information Systems*, 19(3-4):159–203, 2010.
230. V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.
231. G. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.
232. M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel. Semantic Business Process Management: A Vision Towards Using Semantic Web services for Business Process Management. In *IEEE International Conference on e-Business Engineering (ICEBE 2005)*, pages 535 – 540, 2005.