PROCESS-AWARE INFORMATION SYSTEMS: DESIGN, ENACTMENT, AND ANALYSIS

INTRODUCTION

Information technology has changed business processes within and between enterprises. More and more work processes are being conducted under the supervision of information systems that are driven by process models. Examples are workflow management systems such as FileNet P8, Staffware, WebSphere, FLOWer, and YAWL and enterprise resource planning (ERP) systems such as SAP and Oracle. Moreover, many domain-specific systems have components driven by (process) models. It is hard to imagine enterprise information systems that are unaware of the processes taking place. Although the topic of business process management using information technology has been addressed by consultants and software developers in depth, more fundamental approaches toward such *process-aware information systems* (PAISs) have been rare (1). Only since the 1990s have researchers started to work on the foundations of PAISs.

The goal of this article is to (a) provide an overview of PAISs and put these systems in a historical context, (b) to show their relevance and potential to improve business processes, dramatically, and (c) to discuss some more advanced topics to provide insights in current challenges and possible inhibitors. Before going into more detail, we first provide some definitions and give an overview of the different types of PAISs.

PAISs play an important role in *business process management* (BPM). Many definitions of BPM exist. Here we will use the following definition: "Business process management (BPM) is a field of knowledge that combines knowledge from information technology and knowledge from management sciences and applies this to operational business processes." BPM can be viewed as an extension of *workflow management* (WFM), which primarily focuses on the automation of business processes.

Figure 1 shows the relation among PAISs, BPM, and WFM. Note that the term "PAIS" refers to software, whereas the terms "BPM" and "WFM" refer to fields of knowledge in which PAISs can be used. Workflow management systems (WFMSs) can be seen as a particular kind of PAISs where the emphasis is on process automation rather

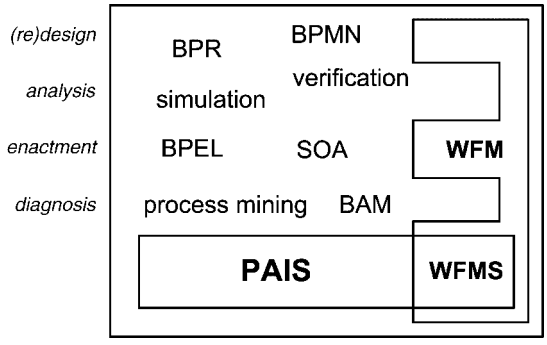

Figure 1. Relating PAISs to other approaches and tools in BPM.

Wiley Encyclopedia of Computer Science and Engineering, edited by Benjamin Wah.

than on redesign and analysis. A definition of WFMS could be "a generic software system that is driven by explicit process designs to enact and manage operational business processes." Clearly, a WFMS should be process-aware and generic in the sense that it is possible to modify the processes it supports. Note that the process designs automated by a WFMS are often graphical and that the focus is on structured processes that need to handle many cases.

Although a WFMS can be seen as a prototypical example of a PAIS, not all PAISs can be classified as pure WFMSs. As shown in Fig. 1, WFMSs are considered to be a subclass of all PAISs. Many examples of systems are process-aware, but they do not provide a generic approach to the modeling and enactment of operational business processes. For example, there may be systems where processes are hard-coded and cannot be modified. For example, many processes supported by an ERP system (e.g., SAP R/3) are hard-coded in software and can only be modified through explicit configuration parameters; that is, the set of possible variation points is predefined, and no notion of a process model can be modified freely. Many organizations have developed software to support processes without using a WFMS; for example, many banks, hospitals, electronic shops, insurance companies, and municipalities have created custom-made software to support processes. These systems are process-aware but are developed without using a WFMS. Another difference between PAISs and WFMSs is the fact that process automation is just one aspect of BPM. Process analysis and diagnosing existing processes clearly extend the scope beyond pure process automation.

Figure 1 also shows some more terms that are relevant in this context (BPR, SOA, BAM, etc.). Business process redesign (BPR) is concerned with finding better process designs. BPR efforts can be supported and enabled by PAISs. Business activity monitoring (BAM) uses information about running processes extracted from PAISs. Process mining techniques can be used to analyze this information and to come up with ideas to improve processes. Recently, the so-called *service-oriented architecture* (SOA) has been proposed as a platform for realizing PAISs. SOA is an architectural style whose goal is to achieve loose coupling among interacting parties. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by different pieces of software. The provider and consumer may reside in the same organization but also within different organizations. By using the SOA, it becomes easier to compose and maintain PAISs, because application functionality can be wrapped into servers that are invoked using a BPEL engine. The Business Process Execution Language (BPEL) (2) is the de facto standard for process execution in a SOA environment. In BPEL one can specify processes and enact them using the process engines of systems such as IBM's WebSphere or Oracle BPEL.

BPEL is a textual XML-based language, and its constructs are close to programming (2). People talk about "programming in the large" illustrating that it is not easy for nonprogrammers to model processes using BPEL. Therefore, languages such as BPMN (Business Process Modeling Notation) (3) have been proposed. Note that many modeling tools support languages similar to BPMN (e.g., ARIS and Protos). Figure 1 shows that the emphasis of execution languages like BPEL is on enactment, whereas languages like BPMN, EPCs, and Protos focus more on (re)design. Note that BPMN is not executable and has no formal semantics. However, in many cases, it is possible to generate some BPEL template code (3,4).

A PAIS requires the modeling of different perspectives (e.g., control flow, information, and organization/resources). This article will mention the different perspectives, but it will focus primarily on the control-flow perspective. Moreover, we use a particular technique to model this perspective: Petri nets (5).

In the remainder of this article, we will first put BPM and related PAIS technology in their historical context (see the next Section 2). Then, we discuss models for process design. As PAIS are typically driven by explicit models, it is important to use the right techniques. Therefore, we discuss techniques for the analysis of process models. We will argue that it is vital to have techniques to assert the correctness of workflow designs. Based on this argument, we focus on systems for process enactment (i.e., systems that actually make the "work flow" based on a model of the processes and organizations involved). Finally, we focus on two more advanced topics: process flexibility and process mining.

BUSINESS PROCESS MANAGEMENT FROM A HISTORICAL PERSPECTIVE

To show the relevance of PAISs, it is interesting to put them in a historical perspective (6). Consider Fig. 2, which shows some ongoing trends in information systems. This figure shows that today's information systems consist of several layers. The center is formed by the operating system (i.e., the software that makes the hardware work). The second layer consists of generic applications that can be used in a wide range of enterprises. Moreover, these applications are typically used within multiple

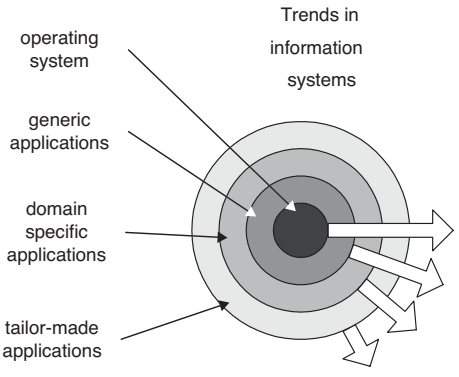

Figure 2. Trends relevant for business process management.

departments within the same enterprise. Examples of such generic applications are a database management system, a text editor, and a spreadsheet program. The third layer consists of domain-specific applications. These applications are only used within specific types of enterprises and departments. Examples are decision support systems for vehicle routing, call center software, and human resource management software. The fourth layer consists of tailor-made applications. These applications are developed for specific organizations.

In the 1960s, the second and third layer were missing. Information systems were built on top of a small operating system with limited functionality. As no generic nor domain-specific software was available, these systems mainly consisted of tailor-made applications. Since then, the second and third layer have developed and the ongoing trend is that the four circles are increasing in size (i.e., they are moving to the outside while absorbing new functionality). Today's operating systems offer much more functionality. Database management systems that reside in the second layer offer functionality that used to be in tailor-made applications. As a result of this trend, the emphasis has shifted from programming to assembling of complex software systems. The challenge no longer is the coding of individual modules but the orchestrating and gluing together of pieces of software from each of the four layers.

Another trend is the shift from data to processes. The 1970s and 1980s were dominated by data-driven approaches. The focus of information technology was on storing and retrieving information, and as a result, data modeling was the starting point for building an information system. The modeling of business processes was often neglected, and processes had to adapt to information technology. Management trends such as business process reengineering illustrate the increased emphasis on processes. As a result, system engineers are resorting to a more process-driven approach.

The last trend we would like to mention is the shift from carefully planned designs to redesign and organic growth. Because of to the omnipresence of the Internet and its standards, information systems change on the fly. Few systems are built from scratch. In most cases, existing applications are partly used in the new system. As a result, software development is much more dynamic.

The trends shown in Fig. 2 provide a historical context for PAISs. PAISs are either separate applications residing in the second layer or are integrated components in the domain-specific applications (i.e., the third layer). Notable examples of PAISs residing in the second layer are WFMSs (7–9) such as Staffware, FileNet P8, and COSA, and case handling systems such as FLOWer. Middleware platforms such as IBM's WebSphere provide a workflow engine (typically based on BPEL (2). Moreover, many open-source WFMSs exist (cf. ActiveBPEL, EnhydraShark, jBPM, and YAWL). Note that leading ERP systems populating the third layer also offer a workflow management module. The workflow engines of SAP, Baan, PeopleSoft, Oracle, and JD Edwards can be considered integrated PAISs. The idea to isolate the management of business processes in a separate component is consistent with the three trends identified. PAISs can be used to avoid hard-coding the work processes into tailor-made applications and thus support the shift from programming to assembling. Moreover, process orientation, redesign, and organic growth are supported. For example, today's workflow management systems can be used to integrate existing applications and to support process change by merely changing the workflow diagram. Given these observations, we hope to have demonstrated the practical relevance of PAISs. In the remainder of this article, we will focus more on the scientific importance of these systems. Moreover, for clarity, we will often restrict the discussion to clear-cut business process management systems such as WFMSs.

An interesting starting point from a scientific perspective is the early work on office information systems. In the 1970s, people like Skip Ellis (10), Anatol Holt (11), and Michael Zisman (12) already worked on so-called office information systems, which were driven by explicit process models. It is interesting to see that the three pioneers in this area independently used Petri net variants to model office procedures. During the 1970s and 1980s, 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 the research almost stopped for a decade. Consequently, hardly any advances were made in the 1980s. In the 1990s, there again was a huge interest in these systems. The number of WFMSs developed in the past decade and the many papers on workflow technology illustrate the revival of office information systems. Today WFMSs are readily available. However, their application is still limited to specific industries such as banking and insurance. As indicated by Skip Ellis, it is important to learn from these ups and downs. The failures in the 1980s can be explained by both technical and conceptual problems. In the 1980s, networks were slow or not present at all, there were no suitable graphical interfaces, and proper development software was missing. However, more fundamental problems also existed: A unified way of modeling processes was missing, and the systems were too rigid to be used by people in the workplace. Most of the technical problems have been resolved by now. However, the more conceptual problems remain. Good standards for business process modeling are still missing, and even today's WFMSs are too rigid.

One of the great challenges of PAISs is to offer both support and flexibility. Today's systems typically are too rigid, thus, forcing people to work around the system. One of the problems is that software developers and computer scientists are typically inspired by processes inside a computer system rather than by processes outside a computer. As a result, these engineers think in terms of control systems rather than in terms of support systems, which explains why few of the existing WFMSs allow for the so-called implicit choice (i.e., a choice resolved by the environment rather than by the system).

To summarize we would like to state that, although the relevance of PAISs is undisputed, many fundamental problems remain to be solved. In the remainder of this article, we will try to shed light on some of these problems.

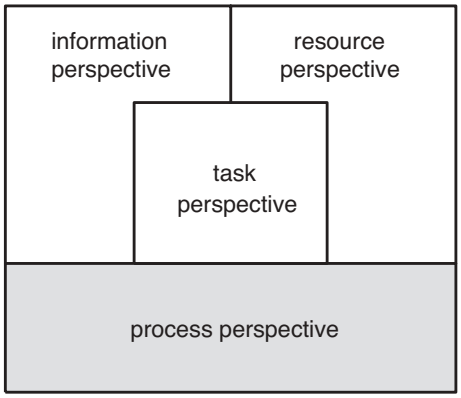

Figure 3. Perspectives of models driving PAISs.

MODELS FOR PROCESS DESIGN

PAISs are driven by models of processes and organizations (1). By changing these models, the behavior of the system adapts to its environment and changing requirements. These models cover different perspectives. Figure 3 shows some of the perspectives relevant for PAISs (9). The process perspective describes the control flow (i.e., the ordering of tasks). The information perspective describes the data that are used. The resource perspective describes the structure of the organization and identifies resources, roles, and groups. The task perspective describes the content of individual steps in the processes. Each perspective is relevant. However, the process perspective is dominant for the type of systems addressed in this article.

Many techniques have been proposed to model the process perspective. Some of these techniques are informal in the sense that the diagrams used have no formally defined semantics. These models are typically very intuitive, and the interpretation shifts depending on the modeler, application domain, and characteristics of the business processes at hand. Examples of informal techniques are ISAC, DFD, SADT, and IDEF. These techniques may serve well for discussing work processes. However, they are inadequate for directly driving information systems because they are incomplete and subject to multiple interpretations. Therefore, more precise ways of modeling are required.

Figure 4 shows an example of an order handling process modeled in terms of a so-called workflow net (13). Workflow nets are based on the classic Petri net model invented by Carl Adam Petri in the early 1960s (5). The squares are the active parts of the model and correspond to tasks. The circles are the passive parts of the model and are used to represent states. In the classic Petri net, the squares are named transitions and the circles places. A workflow net models the lifecycle of one case. Examples of cases are insurance claims, tax declarations, and traffic violations. Cases are represented by tokens, and in this case, the token in *start* corresponds to an order. Task *register* is a so-called AND-split and is enabled in the state shown. The arrow indicates that this task requires human intervention. If a person executes this task, the token is removed from place *start* and two tokens are produced: one for *c1* and one for *c2*. Then, in parallel, two tasks are enabled: *check_availability* and *send_bill*. Depending on the eagerness of the workers executing these two tasks, either *check_available* or *send_bill* is executed first. Suppose *check_availability* is executed first. If the ordered goods are available, they can be shipped by executing task *ship_goods*. If they are not available, either a replenishment order is issued or not. Note that *check_availability* is an OR-split and produces one token for *c3*, *c4*, or *c5*. Suppose that not all ordered goods are available but that the appropriate replenishment orders were already issued. A token is produced for *c3*, and task *update* becomes enabled. Suppose that at this point in time task *send_bill* is executed, resulting in the state with a token in *c3* and *c6*. The token in *c6* is input for two tasks. However, only one of these tasks can be executed, and in this state, only *receive_payment* is enabled. Task *receive_payment* can be executed the moment the payment is received. Task *reminder* is an AND-join/AND-split and is blocked until the bill is sent and the goods have been shipped. Note that the reminder is sent after a specified period as indicated by the clock symbol. However, it is only

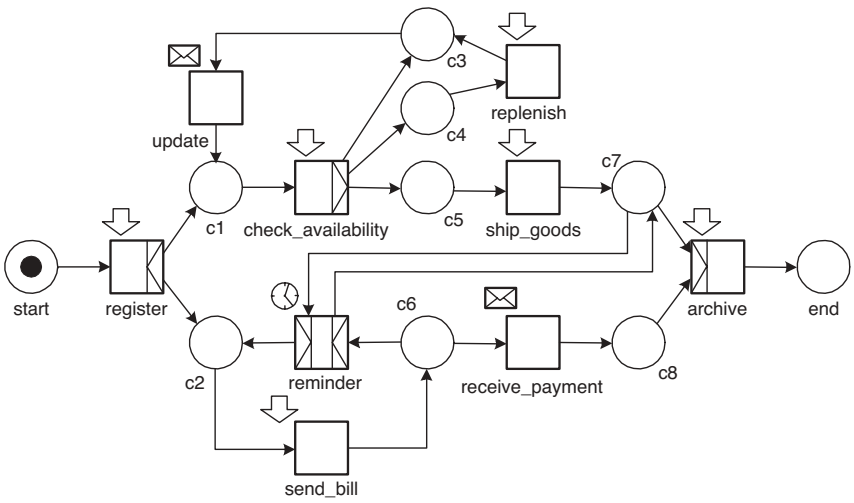

Figure 4. A WF net modeling the handling of orders. The top part models the logistical part of the process, whereas the bottom part models the financial part.

possible to send a remainder if the goods have been actually shipped. Assume that in the state with a token in *c3* and *c6,* task *update* is executed. This task does not require human involvement and is triggered by a message of the warehouse indicating that relevant goods have arrived. Again *check_availability* is enabled. Suppose that this task is executed and that the result is positive. In the resulting state, *ship_goods* can be executed. Now there is a token in *c6* and *c7*, thus enabling task *reminder*. Executing task *reminder* again enables the task *send_bill*. A new copy of the bill is sent with the appropriate text. It is possible to send several reminders by alternating *reminder* and *send_bill*. However, let us assume that after the first loop, the customer pays resulting in a state with a token in *c7* and *c8*. In this state, the AND-join *archive* is enabled and executing this task results in the final state with a token in *end*.

This very simple workflow net shows some of the routing constructs relevant for business process modeling. Sequential, parallel, conditional, and iterative routing are present in this model. There also are more advanced constructs such as the choice between *receive_payment* and *reminder*, which is a so-called *implicit choice* (14) because it is not resolved by the system but by the environment of the system. The moment the bill is sent, it is undetermined whether *receive_payment* or *reminder* will be the next step in the process. Another advanced construct is the fact that task *reminder* is blocked until the goods have been shipped. The latter construct is a so-called *milestone* (14). The reason that we point out both constructs is that many systems have problems supporting these fundamental process patterns (14).

Workflow nets have clear semantics. The fact that one can play the so-called "token game" using a minimal set of rules shows the fact that these models are executable. None of the informal informal techniques mentioned before (i.e., ISAC, DFD, SADT, and IDEF) have formal semantics. Besides these informal techniques, many formal techniques also exist. Examples are the many variants of process algebra and statecharts. The reason we prefer to use a variant of Petri nets is threefold (13):

- Petri nets are graphical and yet precise.
- Petri nets offer an abundance of analysis techniques.
- Petri nets treat states as first-class citizens.

The latter point deserves some more explanation. Many techniques for business process modeling focus exclusively on the active parts of the process (i.e., the tasks), which is very strange because in many administrative processes, the actual processing time is measured in minutes and the flow time is measured in days. This process for measuring means that most of the time cases are in between two subsequent tasks. Therefore, it is vital to model these states explicitly.

At the beginning of this section, we mentioned that there are informal techniques (without formal semantics) and rigorous formal methods such as Petri nets. Over the last two decades, many semiformal methods have been proposed (i.e., in between the two extreme classes mentioned earlier). These methods are informal, however, because the models that need to be transformed into executable code for more rigorous interpretations are added afterward. The UML (Unified Modeling Language) (15) in an example of such a language. It has become the de facto standard for software development. UML has four diagrams for process modeling. UML supports variants of statecharts, and its activity diagrams are inspired by Petri nets (i.e., a token-based semantics is used). Many notations exist that are at the same level as UML activity diagrams. BPMN (3) diagrams and event-driven process chains (EPCs) (16) are examples of such languages. Many researchers are trying to provide solid semantics for UML, EPCs, BPMN, BPEL, and so on. For subsets of these languages, there are formalizations in terms of Petri nets and transition systems. These formalizations typically reveal ambiguous constructs in the corresponding language.

Note that the goal of this article is not to advocate Petri nets as an end-user modeling language. UML, EPCs, and BPMN provide useful constructs that support the workflow designer. However, Petri nets serve as an important foundation for PAIS technology. Without such foundations it is impossible to reason about semantics, correctness, completeness, and so on. A nice illustration is the OR-join in EPC and BPMN models that have semantics leading to paradoxes such as the "vicious circle" (16). Moreover, a solid foundation can be used for analysis as will be shown next.

TECHNIQUES FOR PROCESS ANALYSIS

Many PAISs allow organizations to change their processes by merely changing the models. The models are typically graphical and can be changed ability to change easily. This provides more flexibility than conventional information systems. However, by reducing the threshold for change, errors are introduced more easily. Therefore, it is important to develop suitable analysis techniques. However, it is not sufficient to just develop these techniques. It is as least as important to look at methods and tools to make them applicable in a practical context.

Traditionally, most techniques used for the analysis of business processes originate from operations research. All students taking courses in operations management will learn to apply techniques such as simulation, queuing theory, and Markovian analysis. The focus mainly is on *performance analysis,* and less attention is paid to the correctness of models. *Verification* and *validation* are often neglected. As a result, systems fail by not providing the right support or even break down. Verification is needed to check whether the resulting system is free of logical errors. Many process designs suffer from deadlocks and livelocks that could have been detected using verification techniques. Validation is needed to check whether the system actually behaves as expected. Note that validation is context dependent, whereas verification is not. A system that deadlocks is not correct in any situation. Therefore, verifying whether a system exhibits deadlocks is context independent. Validation is context dependent and can only be done with knowledge of the intended business process.

To illustrate the relevance of validation and verification and to demonstrate some of the techniques available, we

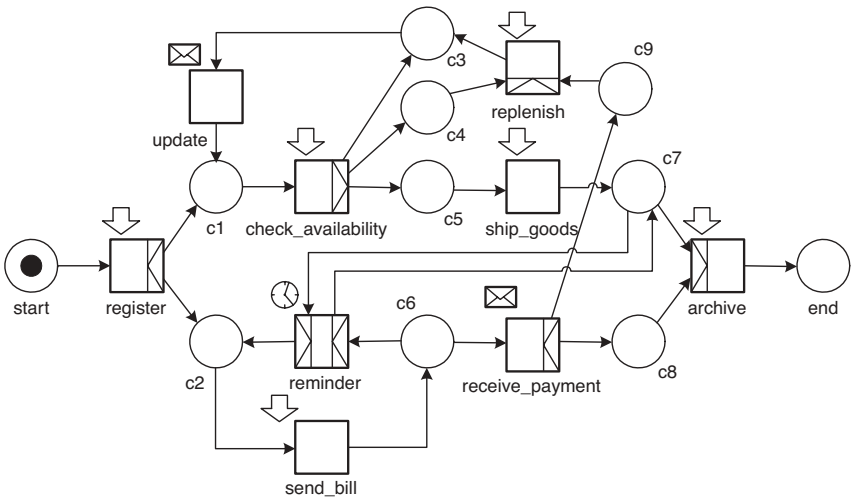

Figure 5. An incorrect WF net. Place *c9* has been added to model that a replenishment order can only be placed if the customer has paid. However, because of this addition, the process can deadlock and a token is left behind in the process if no replenishments are needed.

return to the workflow net shown in Fig. 4. This workflow process allows for the situation where a replenishment is issued before any payment is received. Suppose that we want to change the design such that replenishments are delayed until receiving payment. An obvious way to model this change is to connect task *receive_payment* with *replenish* using an additional place *c9* as shown in Fig. 5. Although this extension seems to be correct at first glance, the resulting workflow net exhibits several errors. The workflow will deadlock if a second replenishment is needed and something is left behind in the process if no replenishments are needed. These logical errors can be detected without any knowledge of the order handling process. For verification, application-independent notions of correctness are needed. One of these notions is the so-called *soundness property*(13). A workflow net is sound if an only if the workflow contains no dead parts (i.e., tasks that can never be executed), from any reachable state it is always possible to terminate, and the moment the workflow terminates all places except the sink place are empty. Note that soundness rules out logical errors such as deadlocks and livelocks. The notion of soundness is applicable to any workflow language. An interesting observation is that soundness corresponds to liveness and boundedness of the short-circuited net (13). The latter properties have been studied extensively (17). As a result, powerful analysis techniques and tools can be applied to verify the correctness of a workflow design. Practical experience shows that many errors can be detected by verifying the soundness property. Moreover, Petri net theory can also be applied to guide the designer toward the error.

Soundness does not guarantee that the workflow net behaves as intended. Consider, for example, the workflow net shown in Fig. 6. Compared with the original model, the shipment of goods is skipped if some goods are not available. Again this idea may seem like a good one at first glance. However, customers are expected to pay even if the goods are never delivered. In other words, task *receive_payment* needs to be executed although task *ship_goods* may never be executed. The latter error can only be detected using knowledge about the context. Based on this context, one may decide whether this is acceptable. Few analysis techniques exist to support this kind of validation automatically. The only means of validation offered by today's WFMSs is gaming and simulation.

An interesting technique to support validation is inheritance of dynamic behavior. Inheritance can be used as a technique to compare processes. Inheritance relates subclasses with superclasses (18). A workflow net is a subclass of a superclass workflow net if certain dynamic properties are preserved. A subclass typically contains more tasks. If by hiding and/or blocking tasks in the subclass one obtains

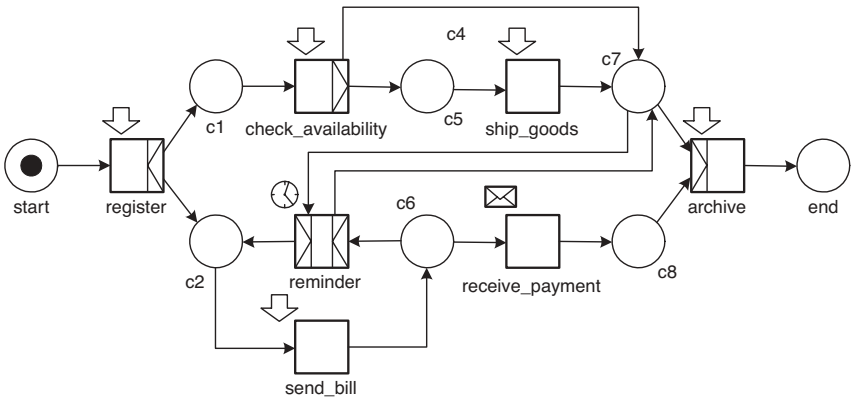

Figure 6. A sound but incorrect WF net. The shipping of goods is no longer guaranteed.

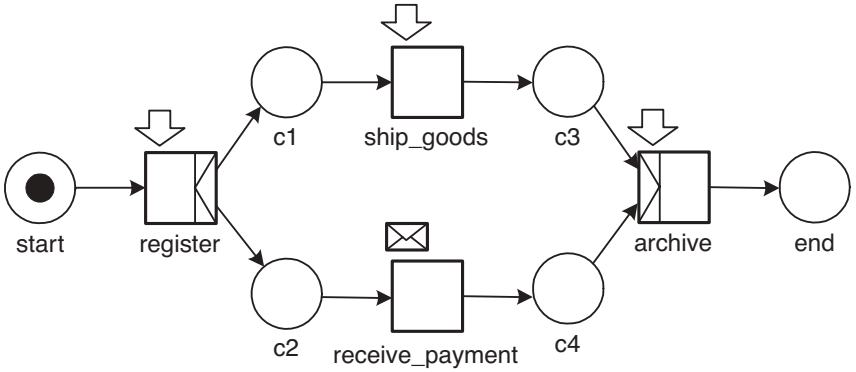

Figure 7. A superclass WF net.

the superclass, the subclass inherits the dynamics of the superclass.[1] The superclass can be used to specify the minimal properties the workflow design should satisfy. By merely checking whether the actual design is a subclass of the superclass, one can validate the essential properties. Consider, for example, Fig. 7. This workflow net describes the minimal requirements the order handling process should satisfy. The tasks *register*, *ship_goods*, *receive_payment*, and *archive* are mandatory. Tasks *ship_goods* and *receive_payment* may be executed in parallel but should be preceded by *register* and followed by *archive*. The original order handling process shown in Fig. 4 is a subclass of this superclass. Therefore, the minimal requirements are satisfied. However, the order handling process shown in Fig. 6 is not a subclass. The fact that task *ship_goods* can be skipped demonstrates that not all properties are preserved.

Inheritance of dynamic behavior is a very powerful concept that has many applications. Inheritance-preserving transformation rules and transfer rules offer support at design time and at run time (19). Subclass–superclass relationships also can be used to enforce correct processes in an e-commerce setting. If business partners only execute subclass processes of some common contract process, then the overall workflow will be executed as agreed. It should be noted that workflows crossing the borders of organizations are particularly challenging from a verification and validation point of view. Errors resulting from miscommunication between business partners are highly disruptive and costly. Therefore, it is important to develop techniques and tools for the verification and validation of these processes. For example, in the context of SOA-based processes (e.g., BPEL processes), the so-called open WF nets (OWF-nets) (20,21) are used to study notions such as controllability and accordance.

Few mature tools aiming at the verification of workflow processes exist. Woflan (22) is one of the notable exceptions. Figure 8 shows a screenshot of Woflan. Woflan combines state-of-the-art scientific results with practical applications (22). Woflan can interface with WFMSs such as Staffware, Websphere, Oracle BPEL, COSA, and YAWL. It can also interface with BPR tools such as Protos and process mining tools such as ProM (23). Workflow processes designed using any of these tools can be verified for correctness. It turns out that the challenge is not to decide whether the design is sound. The real challenge is to provide diagnostic information that guides the designer to the error. Woflan also supports the inheritance notions mentioned before. Given two workflow designs, Woflan can decide whether one is a subclass of the other. Tools such as Woflan illustrate the benefits of a more fundamental approach. Recently, several tools have been developed for the analysis of BPEL processes. The Tools4BPEL toolset consisting of Fiona, LoLa, and BPEL2oWFN is an example of a state-of-the-art BPEL analyzer (21).

To conclude this section, we would like to refer to a study reported in Ref. 24. This study shows that of the 604 process models in the SAP R/3 Reference Model, 20% contain errors that can easily be discovered using verification. Since the middle of the 1990s, the SAP R/3 Reference Model has been available in different versions to support the implementation and configuration of the SAP system. The reference model is not only included in the SAP

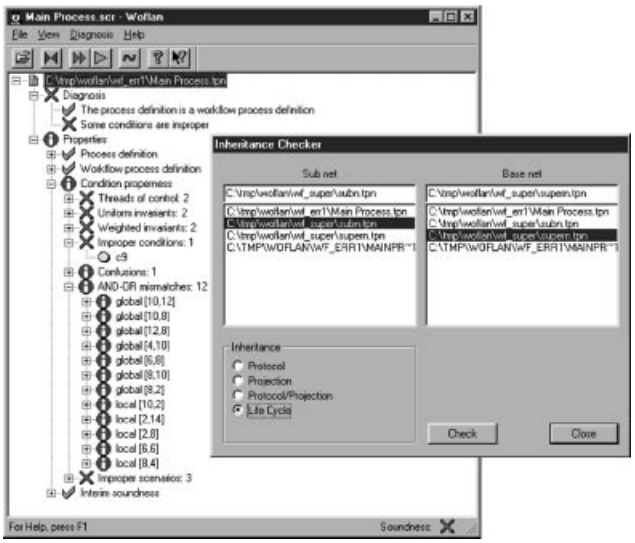

Figure 8. A screenshot showing the verification and validation capabilities of Woflan.

[1]We have identified four notions of inheritance. In this article, we only refer to life-cycle inheritance.

system, but also it is shipped with the business process modeling tools ARIS of IDS Scheer or NetProcess of IntelliCorp. The reference model covers several modeling perspectives such as data and organization structure, but the main emphasis is on 604 nontrivial business processes represented as EPCs. More than 20% of these EPCs contain errors stemming from incorrect combinations of connector elements such as deadlocks and livelocks. A deadlock describes a situation in a process model where a customer order remains waiting for an activity to complete that can never be executed. A simple pattern leading to a deadlock is an XOR split is joined with an AND. A livelock is an infinite loop (i.e., it is impossible to move beyond a certain point and terminate). The many errors in the SAP R/3 Reference Model illustrate the need for rigorous analysis techniques.

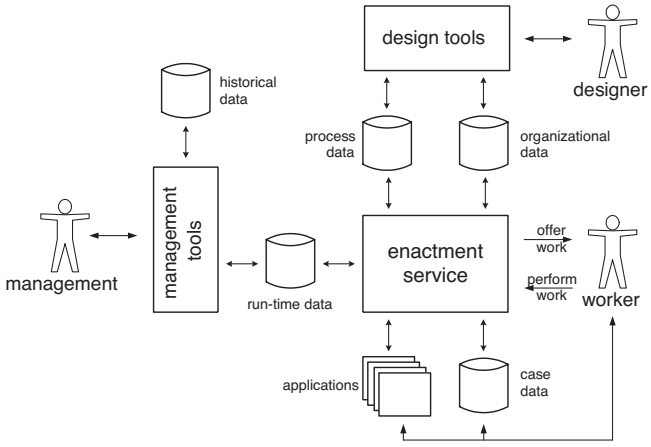

Figure 9. The architecture of a PAIS.

SYSTEMS FOR PROCESS ENACTMENT

Progress in computer hardware has been incredible. In 1964 Gordon Moore predicted that the number of elements on a produced chip would double every 18 months.[2] Up until now, Moore's law still applies. Information technology has also resulted in a spectacular growth of the information being gathered. The commonly used term "information overload" illustrates this growth. Already in 2003, it is estimated that for each individual (i.e., child, man, and woman), 800 megabytes of data are gathered each year (25). The Internet and the World Wide Web have made an abundance of information available at low costs. However, despite the apparent progress in computer hardware and information processing, many information systems leave much to be desired. One problem is that process logic is mixed with application logic. As a result, it is difficult to change a system and people need to "work around the system" rather than getting adequate support. To improve flexibility and reliability, process logic should be separated from application logic. These observations justify the use of solid models and analysis techniques, as discussed before.

Thus far, the focus of this article has been on the design and analysis of work processes. Now it is time to focus on the systems to enact these work processes. Fig. 9 shows the typical architecture of a business process management system. 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. By merely changing these models, the system evolves and adapts, which is the ultimate promise of PAISs.

However, PAIS systems are not the "silver bullet" that solves all problems (i.e., "there is no such thing as a free lunch"), and rigorous modeling is needed to capture processes adequately. Moreover, existing WFMSs still have problems supporting flexibility.

Today's WFMSs have an architecture consistent with Fig. 9. Consider, for example, the screenshots of Staffware shown in Fig. 10. Staffware is one of the leading WFMSs.

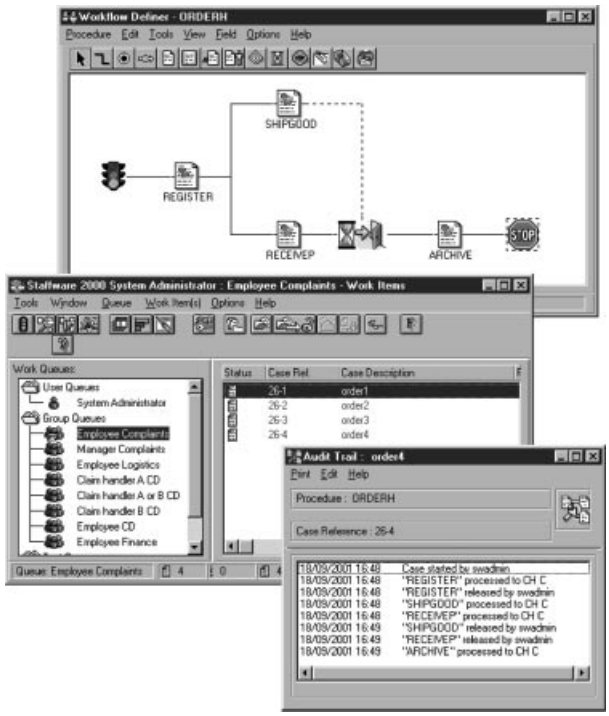

Figure 10. The Graphical Workflow Definer, Work Queue, and Audit Trail of Staffware.

[2]Moore (founder of Intel), commenting on the growth of the microelectronics industry in 1964, noted a doubling of the number of elements on a produced chip once every 12 months. For a decade that meant a growth factor of approximately 1000. Today, when Moore's Law is quoted, the time constant typically quoted is 18 months. However, some argue that a constant of 24 months is more appropriate.

The top window shows the design tool of Staffware while defining a simple workflow process. Work is offered through so-called work queues. One worker can have multiple work queues, and one work queue can be shared among multiple workers. The window in the middle shows the set of available work queues (left) and the content of one of these work queues (right). The bottom window shows an audit trail of a case. The three windows show only some of the capabilities offered by contemporary workflow management systems. It is fairly straightforward to map these windows onto the architecture. In other processes-aware information systems such as, for example, enterprise resource planning systems, one will find the architecture shown in Fig. 9 embedded in a larger architecture.

The architecture shown in Fig. 9 assumes a centralized enactment service. Inside a single organization such an assumption may be realistic. However, in a cross-organizational setting, this is not the case. Fortunately, most vendors now support the SOA, mentioned earlier. In a SOA, tasks are subcontracted to other parties (i.e., what is one task for the service consumer may be a complex process for a service consumer). The Web services stack using standards such as WSDL and BPEL facilitates the development of cross-organizational workflows.

Despite the acceptance of PAISs, the current generation of products leaves much to be desired. To illustrate, we focus on the current generation of WFMSs. We will use Fig. 9 to identify five problems.

First of all, there is a lack of good standards for workflow management. There is, for example, not a good standard for exchanging process models. Existing formats have no clearly defined semantics and fail to capture many routing constructs. Current standards for workflow management are incomplete, inconsistent, at the wrong abstraction level, and mainly driven by the commercial interests of workflow vendors. The Workflow Management Coalition (WfMC) has been trying to standardize workflow processes since the early 1990s. This effort resulted in the Workflow Process Definition Language (WPDL) and the XML Process Definition Language (XPDL). Only a few vendors actively supported these standards. The standards had no clearly defined semantics and encouraged vendors to make product-specific extensions. The BPEL (2) emerged later and is currently the de facto standard for process execution in a SOA environment. However, this language also has no clearly defined semantics and is at a technical level (26). BPEL has the look and feel of a programming language rather than a high-level modeling language.

Second, the expressive power (i.e., the ability to represent complex work processes) of the current generation of WFMSs is insufficient. Several evaluations revealed that the classic WFMSs support less than half of the desirable workflow patterns (14). As an example, consider the workflow process shown in Fig. 4. Few systems can handle the implicit choice and milestone construct identified before. Fortunately, modern systems (e.g., based on BPEL) support more patterns.

A third problem is the lack of understanding of how people actually work. Work processes are more than the ordering of tasks. Work is embedded in a social context. A better understanding of this context is needed to make systems socially aware as well. Modeling processes as if people are "machines" is a too limited view of reality. It is vital to empower workers and to provide more flexibility.

The fourth problem is the limited support for workflow analysis. As indicated before, there are powerful techniques for workflow analysis. However, few systems embed advanced analysis techniques. Besides model-based verification, validation, and performance analysis, new types of analysis are possible. The combination of historical and run-time data, on the one hand, and workflow designs, on the other, offers breathtaking possibilities. Historical data can be used to obtain stochastic data about routing and timing. Using run-time data to reconstruct the current state in a simulation model allows for *on-the-fly simulation*. Simulation based on the current state, historical data, and a good model offers high-quality information about potential problems in the near future. Historical data can also be used for *process mining*. The goal of workflow mining is to derive process models from transaction logs.

Finally, many technical problems remain. Some of these problems can be resolved using Internet-based technology and standards. However, many problems related to the integration of components and long-lived transactions remain unsolved. Since the early 1990s (8) many database researchers have been focusing on transactional aspects of workflows. Note that an instance of a workflow can be viewed as a long running transaction [e.g., some cases (such as a mortgage or insurance) may run dozens of years] (27).

In the remainder, we would like to focus on two particular challenges: *process flexibility* and *process mining*.

CHALLENGE: FLEXIBILITY

Adaptability has become one of the major research topics in the area of workflow management (28). Today's WFMSs and many other PAISs have problems supporting flexibility. As a result, these systems are not used to support dynamically changing business processes or the processes are supported in a rigid manner (i.e., changes are not allowed or handled outside of the system). These problems have been described and addressed extensively in the literature (29–36). Nevertheless, many problems related to flexibility remain unsolved.

In this section, we provide a taxonomy of flexibility because it is probably the biggest challenge today's PAISs are facing. To clarify things, we focus on WFMSs rather than on the broader class of PAISs.

To start, let us identify the different *phases* of a process (instance) in the context of a WFMS:

- *Design time*. At design time, a generic process model is created. This model cannot be enacted because it is not connected to some organizational setting.
- *Configuration time*. At configuration time, a generic model is made more specific and connected to some organizational context that allows it to be instantiated.
- *Instantiation time*. At instantiation time, a process instance is created to handle a particular case (e.g., a customer order or travel request).

- *Run time*. At run time, the process instance is executed according to the configured model. The different activities are being enacted as specified.
- *Auditing time*. At auditing time, the process instance has completed; however, its audit trail is still available and can be inspected and analyzed.

Flexibility plays a role in most phases. At design time, some modeling decisions can be postponed to run time. At run time, one can decide to deviate from the model, and at instantiation time, one can change the process model used for the particular instance. When it comes to flexibility, we identify three *flexibility mechanisms*:

- *Defer, i.e., decide to decide later*. This flexibility mechanism deliberately leaves freedom to maneuver at a later phase. Examples are the use of a declarative process modeling language that allows for the "under specification" of processes and the use of late binding (i.e., the process model has a "hole" that needs to be filled in a later phase).
- *Change, i.e., decide to change model*. Most researchers have addressed flexibility issues by allowing for change. Decisions made at an earlier phase may be revisited. For example, for premium customers, the process may be adapted in an ad hoc manner. The change may refer to the model for a single instance (adhoc change) or to all future cases (evolutionary change). In both cases, a change can create inconsistencies. For evolutionary change, cases may also need to be migrated.
- *Deviate, i.e., decide to ignore model*. The third mechanism is to deviate from the model (e.g., tasks are skipped even if the model does not allow for this to happen). In many environments, it is desirable that people are in control (i.e., the system can only suggest activities but not force them to happen).

Figure 11 relates the two dimensions just mentioned. Based on the different phases and the three mechanisms, different types of flexibility are classified. Note that we did not mention any examples of flexibility at auditing time. After the process instance completes, it is not possible to defer, change, or violate things, because this would imply fraud. Figure 11 can be used to characterize the support for flexibility of a concrete WFMS. Unfortunately, today's systems support only a few forms of flexibility, thus limiting the applicability of PAISs.

It is impossible to provide a complete overview of the work done on flexibility in workflows. The reader is referred to Ref. 28 for another taxonomy. Many authors have focused on the problems related to change (31–34) (cf. the cell "change at run time" in Fig. 11). The problem of changing a process while instances are running was first mentioned in Ref. 31. In the context of ADEPT (32,34), many problems have been addressed. See Ref. 34 for an excellent overview of problems related to dynamic change (the cell "change at run time" in Fig. 11). Other authors approach the problem by avoiding change, for example, either by using a more declarative language (33) or by late binding (30,35,36) (also referred to as worklets, pockets of flexibility, or process fragments). These approaches fit into the column "defer" in Fig. 11. Another interesting approach is provided by the system FLOWer of Pallas Athena. This system uses the so-called "case handling" concept to provide more flexibility (29). Most of the ideas of case handling relate to the "defer" and "deviate" columns in Fig. 11.

CHALLENGE: PROCESS MINING

Process mining has emerged as a way to analyze systems and their actual use based on the event logs they produce (37,38). Process mining always starts with *event logs*. Events logs may originate from all kinds of PAISs. Examples are classic WFMSs, ERP systems (e.g., SAP), case handling systems (e.g., FLOWer), PDM systems (e.g., Windchill), CRM systems (e.g., Microsoft Dynamics CRM), middleware (e.g., IBM's WebSphere), hospital information systems (e.g., Chipsoft), and so on. These systems provide very detailed information about the activities that have been executed.

The goal of process mining is to extract information (e.g., process models) from these logs (i.e., process mining describes a family of *a posteriori* analysis techniques exploiting the information recorded in the event logs). Typically, these approaches assume that it is possible to record events sequentially 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).

	defer (decide to decide later)	*change (decide to change model)*	*deviate (decide to ignore model)*
design time	e.g., defer to run-time by using late binding or declarative modeling	N/A	N/A
configuration time	e.g., defer configuration decisions	e.g., remodel parts of the process at configuration time	e.g., violate a configuration constraint
instantiation time	e.g., defer the selection of parameters or process fragments	e.g., modify model for a particular customer	N/A
run time	N/A	e.g., change model for running instance or migrate instance to new model	e.g., skip or redo a task while this is not specified
auditing time	N/A	N/A	N/A

Figure 11. Classification of the different types of flexibility based on the phase and mechanism.

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).

Process mining addresses the problem that most "process/system owners" have limited information about what is actually happening. In practice, there is often a significant gap between what is prescribed or supposed to happen and what *actually* happens. Only a concise assessment of reality, which process mining strives to deliver, can help in verifying process models and ultimately be used in system or process redesign efforts.

The idea of process mining is to discover, monitor, and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs. We consider three basic types of process mining (Fig. 12):

- *Discovery*: No a priori model exists (i.e., based on an event log, some model is constructed). For example, using the α-algorithm (37), a process model can be discovered based on low-level events.
- *Conformance*: An a priori model exists. This model is used to check whether reality conforms to the model. For example, a process model may indicate that purchase orders of more than one million Euro require two checks. Another example is the checking of the four-eyes principle. Conformance checking may be used to detect deviations, to locate and explain these deviations, and to measure the severity of these deviations.
- *Extension*: An a priori model exists. This model is extended with a new aspect or perspective (i.e., the goal is not to check conformance but to enrich the model with the data in the event log). An example is the extension of a process model with performance data (i.e. some a priori process model is used on which bottlenecks are projected).

Traditionally, process mining has been focusing on *discovery* (i.e., deriving information about the original process model, the organizational context, and execution properties from enactment logs). An example of a technique addressing the control flow perspective is the α-algorithm, which constructs a Petri net model (17) describing the behavior observed in the event log. However, process mining is not limited to process models (i.e., control flow) and recent process mining techniques are more and more focusing on other perspectives (e.g., the organizational perspective or the data perspective). For example, there are approaches to extract social networks from event logs and analyze them using social network analysis. This allows organizations to monitor how people, groups, or software/system components are working together.

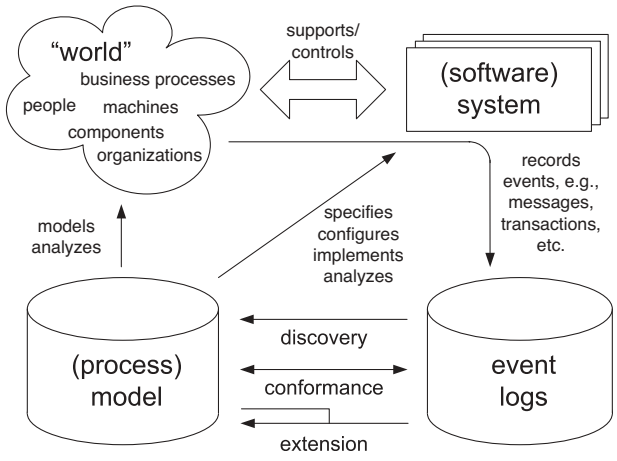

Figure 12. Three types of process mining: (1) discovery, (2) conformance, and (3) extension.

Conformance checking compares an a priori model with the observed behavior as recorded in the log. In Ref. 39, it is shown how a process model (e.g., a Petri net) can be evaluated in the context of a log using metrics such as "fitness" (Is the observed behavior possible according to the model?) and "appropriateness" (Is the model "typical" for the observed behavior?). However, it is also possible to check conformance based on organizational models, predefined business rules, temporal formulas, quality of service (QoS) definitions, and so on.

There are different ways to *extend* a given process model with additional perspectives based on event logs (e.g., decision mining, performance analysis, and user profiling). Decision mining, also referred to as decision point analysis, aims at the detection of data dependencies that affect the routing of a case. Starting from a process model, one can analyze how data attributes influence the choices made in the process based on past process executions. Classic data mining techniques such as decision trees can be leveraged for this purpose. Similarly, the process model can be extended with timing information (e.g., bottleneck analysis).

At this point in time there are mature tools such as the ProM framework (23), featuring an extensive set of analysis techniques that can be applied to real-life logs while supporting the whole spectrum depicted in Fig. 12.

Although flexibility requirements may form an inhibitor for the application of PAISs, process mining techniques may in fact increase the value of a PAIS. The structured analysis of the event logs of PAISs provides an added value over information systems that are not aware of the processes these support.

Process mining is strongly related to classic data mining approaches (40). However, the focus is not on data but on process-related information (e.g., the ordering of activities). Process mining is also related to monitoring and business intelligence (41).

CONCLUSION

Process-aware information systems (PAISs) follow a characteristic *lifecycle*. Figure 13 shows the four phases of such a lifecycle (7). In the *design phase*, the processes are (re)designed. In the *configuration phase*, designs are implemented by configuring a PAIS (e.g., a WFMS). After configuration, the *enactment phase* starts where the operational business processes are executed using the system configured. In the *diagnosis phase*, the operational processes are analyzed to identify problems and to find

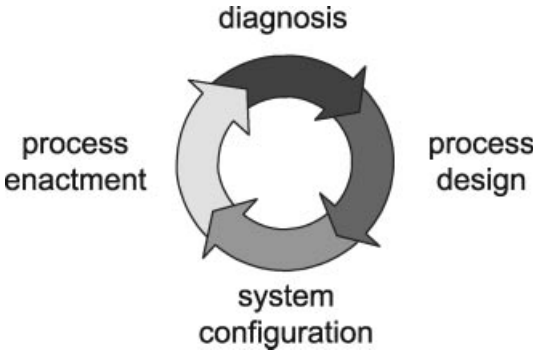

Figure 13. PAIS lifecycle. (This figure is available in full color at http://www.interscience.wiley.com/reference/ecse.)

things that can be improved. The focus of traditional workflow management (systems) is on the lower half of the lifecycle. As a result, there is little support for the diagnosis phase. Moreover, support in the design phase is limited to providing a graphical editor while analysis and real design support are missing.

In this article, we showed that PAISs support operational business processes by combining advances in information technology with recent insights from management science. We started by reviewing the history of such systems and then focused on process design. From the many diagramming techniques available, we chose one particular technique (Petri nets) to show the basics. We also emphasized the relevance of process analysis, for example, by pointing out that 20% of the more than 600 process models in the SAP reference model are flawed (24). We also discussed the systems that enact such process designs (e.g., the workflow engines embedded in various systems) and concluded by elaborating on two challenges: flexibility and process mining. More flexibility is needed to widen the scope of PAISs. Today's systems tend to restrict people in their actions, even if this is not desired. Process mining is concerned with extracting knowledge from event logs. This is relatively easy in the context of PAISs and offers many opportunities to improve the performance of the underlying business processes. Moreover, process mining is an essential factor in closing the PAIS lifecycle shown in Fig. 13.

BIBLIOGRAPHY

1. M. Dumas, W. M. P. van derAalst, and A. H. M. ter Hofstede, *Process-Aware Information Systems: Bridging People and Software through, Process Technology*, New York: Wiley & Sons, 2005.
2. 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. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.
3. S. A. White et al., Business Process Modeling Notation Specification (Version 1.0, OMG Final Adopted Specification), 2006.
4. C. Ouyang, M. Dumas, A. H. M. terHofstede, and W. M. P. van der Aalst, Pattern-Based Translation of BPMN Process Models to BPEL Web Services, *Int. J. Web Serv. Res.*, **5**(1): 42–62, 2007.
5. W. Reisig and G. Rozenberg, ed., *Lectures on Petri Nets I: Basic Models,* volume 1491 *of Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 1998.
6. W. M. P. van der Aalst, Making, work flow: On the application of Petri nets to business process management, in J. Esparza and C. Lakos (eds.), *Application and Theory of Petri Nets 2002*, Vol. 2360 *of Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 2002, pp. 1–22.
7. W. M. P. van der Aalst and K. M. van Hee, *Workflow Management: Models, Methods, and Systems*, Cambridge, MA: MIT Press, 2004.
8. D. Georgakopoulos, M. Hornick, and A. Sheth, An overview of workflow management: From process modeling to workflow automation infrastructure, *Dist. Parallel Databases*, **3**: 119–153, 1995.
9. S. Jablonski, and C. Bussler, *Workflow Management: Modeling Concepts, Architecture, and Implementation*, London: International Thomson Computer Press, 1996.
10. C. A. Ellis, Information control nets: A mathematical model of office information flow, *Proc. Conference on Simulation, Measurement and Modeling of Computer Systems*, ACM Press: Boulder, Colorado, 1979, pp. 225–240.
11. A. W. Holt, Coordination technology and Petri nets, in G. Rozenberg (ed.), *Advances in Petri Nets 1985*, Vol. 222 *of Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 1985, pp. 278–296.
12. M. D. Zisman, *Representation, Specification and Automation of Office Procedures*. PhD thesis, University of Pennsylvania, Warton School of Business, 1977.
13. W. M. P. van der Aalst, The application of Petri nets to workflow management, *J. Circuits, Sys. Comp.*, **8**(1): 21–66, 1998.
14. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, Workflow patterns, *Dist. Parallel Databases*, **14**(1): 5–51, 2003.
15. Object Management Group, *OMG Unified Modeling Language 2.0*. OMG, Available: http://www.omg.com/uml/, 2005.
16. E. Kindler. On the semantics of EPCs: A framework for resolving the vicious circle, *Data Know. Eng.*, **56**(1): 23–40, 2006.
17. J. Desel and J. Esparza, *Free Choice Petri Nets*, Vol. 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge UK: Cambridge University Press, 1995.
18. T. Basten and W. M. P. van der Aalst, Inheritance of behavior, *J. Logic Algebraic Programming*, **47**(2): 47–145, 2001.
19. W. M. P. van der Aalst and T. Basten, Inheritance of workflows: An approach to tackling problems related to change, *Theor. Comp. Sci.*, **270**(1–2): 125–203, 2002.
20. 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 (eds.), *Informal Proc. of the 4th International Workshop on Web Services and Formal Methods (WS-FM 2007)*; QUT, Brisbane Australia, 2007, 119–134.
21. N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg, Analyzing interacting BPEL processes, in S. Dustdar, J. L. Faideiro, and A. Sheth (eds.), *International Conference on Business Process Management (BPM 2006),* Vol. 4102 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 2006, 17–32.
22. H. M. W. Verbeek, T. Basten, and W. M. P. van der Aalst, Diagnosing workflow, processes using woflan, *Comp. J.*, **44**(4): 246–279, 2001.

23. W. M. P. van der Aalst, B. F. van Dongen, C. W. Gnther, R. S. Mans, A. K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H. M. W. Verbeek, and A. J. M. M. Weijters. ProM 4.0: Comprehensive support for real process analysis, in J. Kleijn and A. Yakovlev (eds.), *Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007)*, Vol. 4546 *of Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 2007, pp. 484–494.
24. J. Mendling, G. Neumann, and W. M. P. van der Aalst. Understanding the occurrence of errors in process models based on metrics, in F. Curbera, F. Leymann, and M. Weske (eds.), *Proc. OTM Conference on Cooperative information Systems (CoopIS 2007),* Vol. 4803 of *Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 2007, pp. 113–130.
25. P. Lyman and H. Varian, How Much Information. Available: http://www.sims.berkeley.edu/how-much-info.
26. W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, N. Russell, H. M. W. Verbeek, and P. Wohed, Life after BPEL?, in M. Bravetti, L. Kloul, and G. Zavattaro (eds.), *WS-FM 2005,* Vol. 3670 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 2005, 35–50.
27. G. Weikum and G. Vossen, *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*, San Francisco, CA: Morgan Kaufmann Publishers, 2002.
28. W. M. P. van der Aalst and S. Jablonski, Dealing with workflow change: identification of issues and solutions, *Int. J. Comp. Sys., Sci., Eng.*, **15**(5): 267–276, 2000.
29. W. M. P. van der Aalst, M. Weske, and D. Grnbauer. Case handling: A new paradigm for business process support. *Data Knowl. Eng.*, **53**(2): 129–162, 2005.
30. 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 (eds.), *Proc. OTM Conference on Cooperative information Systems (CoopIS 2007),* Vol. 4803 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 2007, 95–112.
31. 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 (eds.), *Proc. Conference on Organizational Computing Systems*; pages 10–21, Milpitas, California, 1995.
32. M. Reichert and P. Dadam, ADEPTflex: Supporting dynamic changes of workflow without loosing control, *J. Intell. Inf. Sys.*, **10**(2): 93–129, 1998.
33. 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 (eds.), *Proc. OTM Conference on Cooperative information Systems (CoopIS 2007),* Vol. 4803 of *Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 2007, pp. 77–94.
34. S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems: A survey, *Data Know. Eng.*, **50**(1): 9–34, 2004.
35. S. Sadiq, W. Sadiq, and M. Orlowska, Pockets of flexibility in workflow specification, in *Proc. 20th International Conference on Conceptual Modeling (ER 2001),* Vol. 2224 *of Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 2001, pp. 513–526.
36. M. Weske, Formal foundation and conceptual design of dynamic adaptations in a workflow management system, in R. Sprague (ed.), *Proc. Thirty-Fourth Annual Hawaii International Conference on System Science (HICSS-34)*. Los Alamitos, CA: IEEE Computer Society Press, 2001.
37. W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster, Workflow mining: Discovering process models from event logs, *IEEE Trans. Knowl. Data Eng.*, **16**(9): 1128–1142, 2004.
38. W. M. P. van der Aalst, H. A. Reijers, A. J. M. M. Weijters, B. F. vanDongen, A. K. Alves de Medeiros, M. Song, and H. M. W. Verbeek, Business process mining: An industrial application, *Inf. Sys.*, **32**(5): 713–732, 2007.
39. A. Rozinat and W. M. P. van der Aalst, *Conformance testing: Measuring the fit and appropriateness of event logs and process models*, in C. Bussler et al. (ed.), *BPM 2005 Workshops (Workshop on Business Process Intelligence),* volume 3812 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 2006, 163–176.
40. I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques* (2nd edition). San Francisco, CA: Morgan Kaufmann, 2005.
41. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M. C. Shan, Business process intelligence, *Comp. Ind.*, **53**(3): 321–343, 2004.

WIL M.P. VAN DER AALST
Eindhoven University of Technology
Eindhoven, The Netherlands