Making Work Flow: On the Application of Petri nets to Business Process Management

W.M.P. van der Aalst*

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

Abstract. 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 Staffware, enterprise resource planning systems such as SAP and Baan, but also include many domain specific systems. 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, a more fundamental approach has been missing. Only since the nineties, researchers started to work on the foundations of business process management systems. This paper addresses some of the scientific challenges in business process management. In the spirit of Hilbert's problems[1], 10 interesting problems for people working on Petri-net theory are posed.

1 Introduction

The goal of this paper is to show the relevance, architecture, and Achilles heel of business process management systems. This way we hope to interest Petri-net researchers in some of the scientific challenges in this domain. The definition of a business process management system used throughout this paper is: *a generic software system that is driven by explicit process designs to enact and manage operational business processes*. The system should be process-aware and generic in the sense that it is possible to modify the processes it supports. The process designs are often graphical and the focus is on structured processes that need to handle many cases.

In the remainder of this paper, we will first put business process management and related technology in its historical context. Then, we will discuss models for process design. Since business process management systems are driven by explicit models, it is important to use the right techniques. Next, we will 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 we will focus on systems for process

* Part of paper is taken from my inaugural lecture "Making Work Flow: On the Design, Analysis, and Enactment of Business Processes" [6].

[1] Note that by no means we are suggesting that the problems in this paper are of the same stature as the 23 problems raised by David Hilbert in 1900.

enactment, i.e., systems that actually make the "work flow" based on a model of the processes and organizations involved. Finally, we will pose 10 interesting problems in the spirit of Hilbert's problems [31].

2 Business process management from a historical perspective

Only the wisest and stupidest of men never change.
Confucius

To show the relevance of business process management systems, it is interesting to put them in a historical perspective. Consider Figure 1, which shows some of the ongoing trends in information systems. This figure shows that today's information systems consist of a number of 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 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 sixties the second and third layer were missing. Information systems were built on top of a small operating system with limited functionality. Since 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 which used to be in tailor-made applications. As a result of this trend, the emphasis shifted from programming to assembling of complex software systems. The challenge no longer is the coding of individual modules but orchestrating and gluing together pieces of software from each of the four layers.

Another trend is the shift from data to processes. The seventies and eighties 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. Due to the omnipresence of the Internet and its standards, information systems change on-the-fly. As a result, fewer systems are built from scratch. In many cases existing applications are partly used in the new system. Although component-based software development still has it problems, the goal is clear and it is easy to see that software development has become more dynamic.

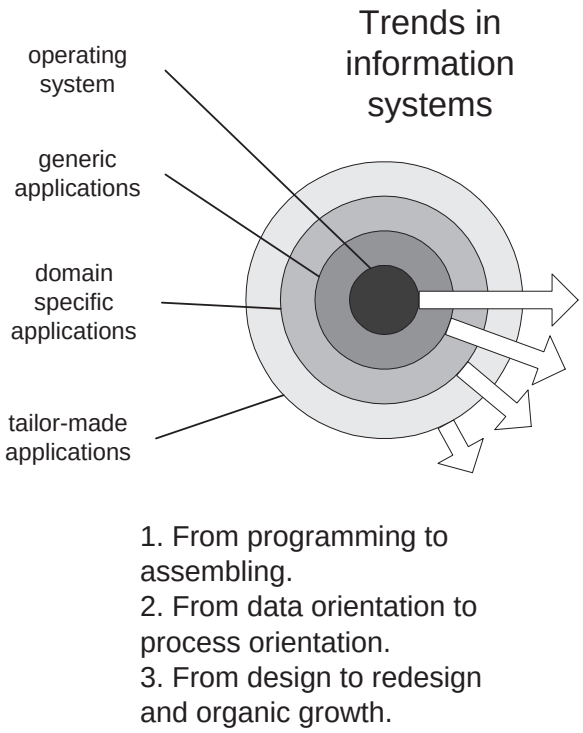

Fig. 1. Trends relevant for business process management.

The trends shown in Figure 1 provide a historical context for business process management systems. Business process management systems 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 business process management systems residing in the second layer are workflow management systems [33, 36] such as Staffware, MQSeries, and COSA, and case handling systems such as FLOWer. Note that leading enterprise resource planning 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 as integrated business process management systems. The idea to isolate the management of business processes in a separate component is consistent with the three trends identified. Business process management systems 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 support process change by merely changing the workflow diagram. Give these observations, we hope to have demonstrated the practical relevance of business process management systems. In the remainder of this paper 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 workflow management systems.

An interesting starting point from a scientific perspective is the early work on office information systems. In the seventies, people like Skip Ellis [24], Anatol Holt [32], and Michael Zisman [46] 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 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 again was a huge interest in these systems. The number of workflow management systems developed in the past decade and the many papers on workflow technology illustrate the revival of office information systems. Today workflow management systems are readily available [36]. However, their application is still limited to specific industries such as banking and insurance. As was indicated by Skip Ellis it is important to learn from these ups and downs [26]. The failures in the eighties can be explained by both technical and conceptual problems. In the eighties, networks were slow or not present at all, there were no suitable graphical interfaces, and proper development software was missing. However, there were also more fundamental problems: 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 workflow management systems enforce unnecessary constrains on the process logic (e.g., processes are made more sequential).

WHIRLWIND (1953)

Architecture: 32 bit word length, duplex CPU, 75kips single address, no interrupts, 4 index registers, real time clock
Memory: magnetic core (4Kx64word) 6 microseconds cycle time; magnetic drum (150K word); 4 IBM Model 729 tape drives (~100K word each); parity checking
I/O: CRT display, keyboard, light gun, real time serial data (teletype 1300bps modem), voice line
Size: 60,000 vacuum tubes, 175,000 diodes, 13,000 transistors; CPU space 50x150 feet each; CPU weight 500,000 lbs; power consumption: 3 megawatts

Fig. 2. The Whirlwind - photo from the Timeline of Events on Computer History (©2001 IEEE).

One of the great challenges of business process management systems is to offer both support and flexibility [9, 14, 35]. 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 processes outside a computer. Figure 2 illustrates the typical mind-frame of people developing business process management systems. This photograph shows the Whirlwind computer, which was the first computer system to have magnetic core memory (1953). It is interesting to mention that Whirlwind was developed by Jay Forrester who also developed the well-known Systems Dynamics approach [27]. Software engineers are typically trained in the architecture and systems software of computers like the Whirlwind and its successors. As a result, these engineers think in terms of control systems rather than support systems. This explains that few of the existing workflow management systems allow for the so-called implicit choice, i.e., a choice resolved by the environment rather than the system [13]. To solve these problems, the typical mind-frame should be changed such that the business process management system is viewed as a reactive system rather than merely a control system.

To summarize we state that, although the relevance of business process management systems is undisputed, many fundamental problems remain to be solved. In the remainder of this paper we will try to shed light on some of these problems.

3 Models for process design

A camel is a horse designed by committee.
Sir Alec Issigonis

Business process management systems are driven by models of processes and organizations. 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 business process management systems [33]. 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 talk.

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 since 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 [1]. Workflow nets are based on the classical Petri-net model

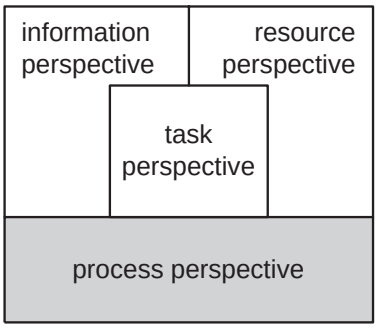

Fig. 3. Perspectives of models driving business process management systems.

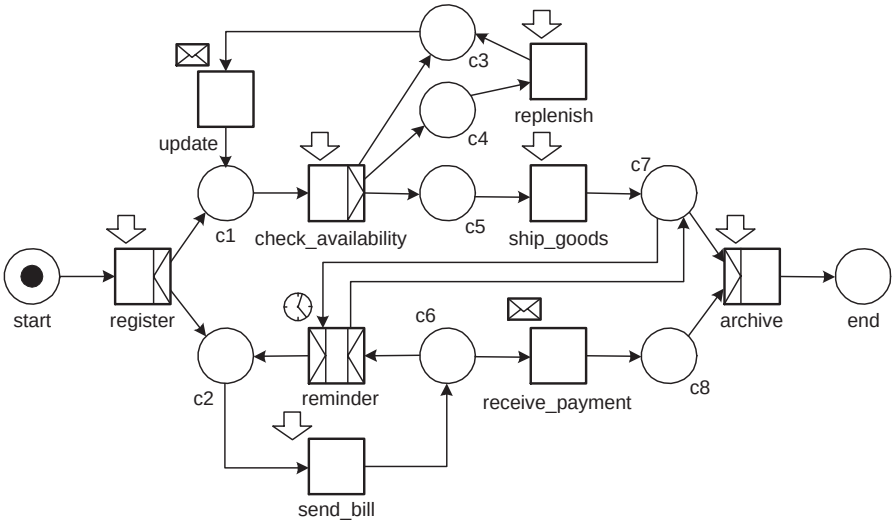

Fig. 4. WF-net.

invented by Carl Adam Petri in the early sixties [39]. 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 classical Petri net, the squares are named transitions and the circles places. A workflow net models the life-cycle 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 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 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 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*. This is a so-called *implicit choice* since 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*. The reason that we point out both constructs is that many systems have problems supporting these rather fundamental process patterns [13].

Workflow nets have clear semantics. The fact that we are able to 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 workflow nets there are many other formal techniques. Examples are the many variants of process algebra [17] and statecharts [29]. The reason we prefer to use a variant of Petri nets is threefold [1]:

- 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. This is rather surprising since in many administrative processes the actual processing time is measured in minutes and the flow time is measured in days. This means that most of

the time cases are in-between two subsequent tasks. Therefore, it is vital to model these states explicitly.

In recent years, the Unified Modeling Language (UML, [20]) 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. UML combines both good and bad ideas and can be considered semi-formal. Many colleagues are trying to provide solid semantics for UML. In my opinion, it would have been better to start with a solid foundation.

4 Techniques for process analysis

From the errors of others, a wise man corrects his own.
Syrus

Business process management systems allow organizations to change their processes by merely changing the models. The models are typically graphical and can be changed quite 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, queueing 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 [2, 42]. 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 while 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 return to the workflow net shown in Figure 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 is to connect task *receive_payment* with *replenish* using an additional place *c9* as shown in Figure 5. Although this extension seems to be correct at first glance, the resulting workflow net has 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 are logical errors that 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* [1]. A workflow net is sound if and only if the

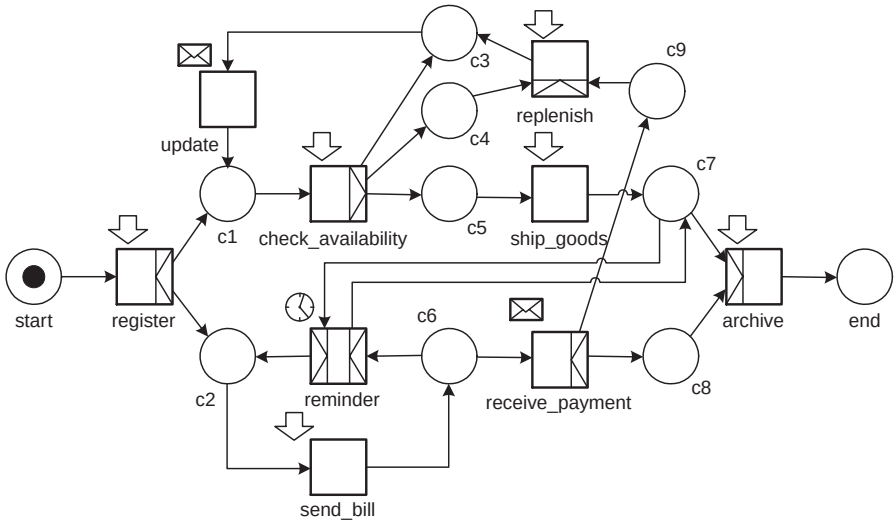

Fig. 5. An incorrect WF-net.

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 (i.e., place *end*) 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 [1]. The latter properties have been studied extensively [41, 23]. 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 towards the error.

Soundness does not guarantee that the workflow net behaves as intended. Consider for example, the workflow net shown in Figure 6. Compared to the original model, the shipment of goods is skipped if some of the goods are not available. Again this may seem to be a good idea 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 or not. Few analysis techniques exist to automatically support this kind of validation. The only means of validation offered by today's workflow management systems 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 [19]. 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 the superclass, the subclass

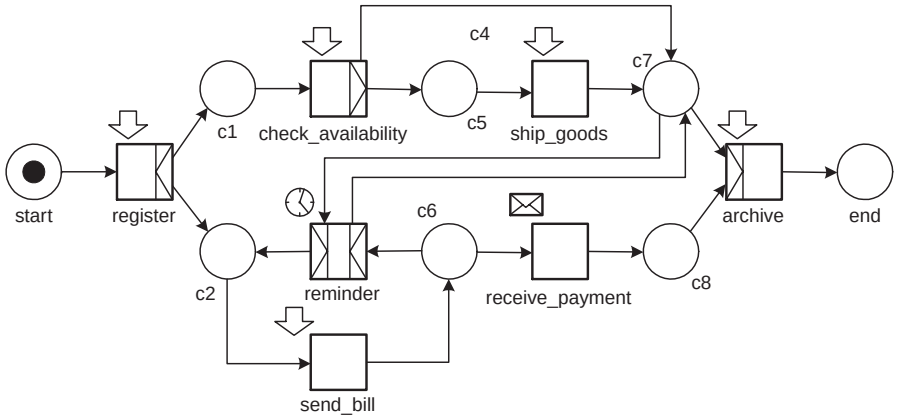

Fig. 6. A sound but incorrect WF-net.

inherits the dynamics of the superclass.[2] 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 Figure 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 Figure 4 is a subclass of this superclass. Therefore, the minimal requirements are satisfied. However, the order handling process shown in Figure 6 is not a subclass. The fact that task *ship_goods* can be skipped demonstrates that not all properties are preserved.

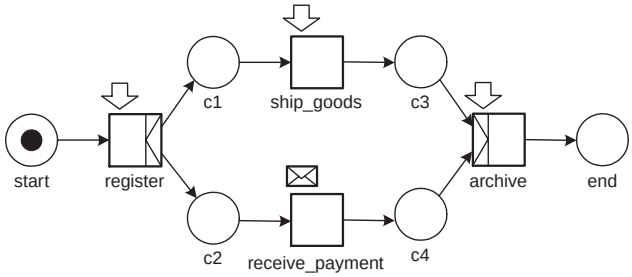

Fig. 7. A superclass WF-net.

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

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 [8]. 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 [4]. 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.

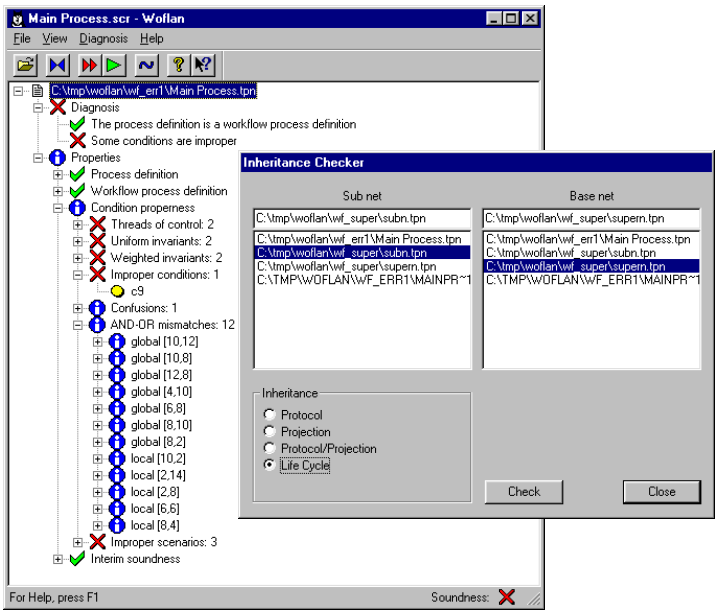

Fig. 8. A screenshot showing the verification and validation capabilities of Woflan.

Few tools aiming at the verification of workflow processes exist. Woflan [43] and Flowmake [42] are two notable exceptions. We have been working on Woflan since 1997. Figure 8 shows a screenshot of Woflan. Woflan combines state-of-the-art scientific results with practical applications [3, 11, 43, 45]. Woflan can interface with leading workflow management systems such as Staffware and COSA. It can also interface with BPR-tools such as Protos. 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 or not. 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 is able to decide whether one is a subclass of the other. Tools such as Woflan illustrate the benefits of a more fundamental approach.

Large scale experiments with experienced students show that workflow designers frequently make errors and that these design errors can be detected using Woflan [43].

5 Systems for process enactment

> *If the automobile had followed the same development cycle as the computer, a Rolls-Royce would today cost $100, get one million miles to the gallon, and explode once a year, killing everyone inside.*
> **Robert Cringely**

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. [3] 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. It is estimated that for each individual, i.e., child, man, and woman, 250 megabytes of data are gathered each year [37]. 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. Typically, software contains errors and people need to work around the system to get things done. These observations justify the use of solid models and analysis techniques, as discussed before.

Thus far, the focus of this paper has been on the design and analysis of work processes. Now it is time to focus on the systems to enact these work processes. Figure 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. This is the ultimate promise of business process management systems.

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

[3] 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.

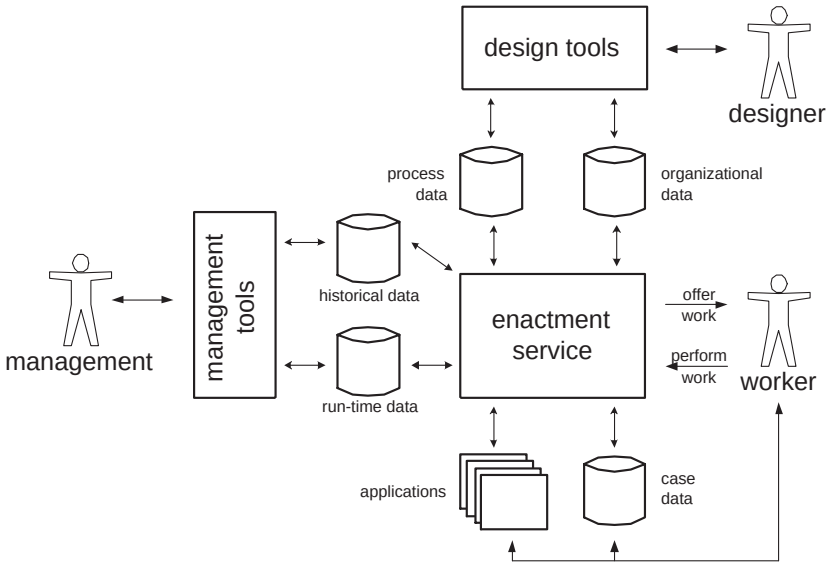

Fig. 9. The architecture of a business process management system.

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 enterprise resource planning systems, one will find the architecture shown in Figure 9 embedded in a larger architecture.

Despite the acceptance of process-aware information systems, the current generation of products leaves much to be desired. In the next section we will highlight some of the challenging problems in this domain.

6 Challenging problems

> *A scientific truth does not triumph by convincing its opponents and making them see the light, but rather because its opponents eventually die and a new generation grows up that is familiar with it.*
>
> **Maxwell Planck**

In 1900 David Hilbert gave a lecture before the International Congress of Mathematicians in Paris. In this lecture he gave gave a massive homework assignment to all the mathematicians of the world in the form of 23 problems [31]. These problems became the backbone for mathematical research in the 20th century. Most of the problems have been partially solved; some have been restated and the new interpretations have been solved. By merely formulating a number of problems, Hilbert directed research efforts towards the problems he thought to be relevant. In this paper, we are trying to direct the attention of researchers to 10 Petri-net-related problems relevant for business process management.

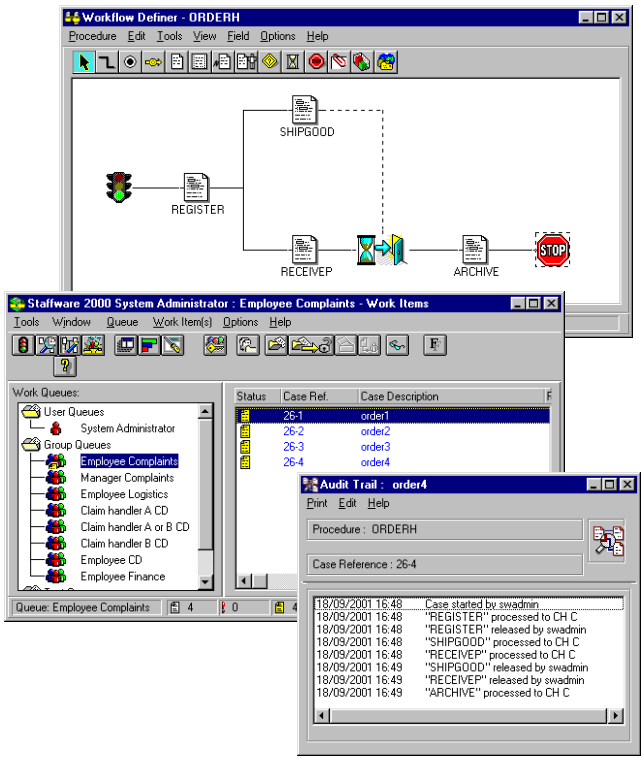

Fig. 10. The Graphical Workflow Definer, Work Queue, and Audit Trail of Staffware.

1. *What is the complexity of deciding soundness?*
   Given a workflow net it is possible to decide soundness by constructing and inspecting the reachability graph. It is also shown that soundness coincides with liveness and boundedness of the short-circuited net [1]. For subclasses such as free-choice workflow nets and workflow nets without PT- and TP-handles soundness can be analyzed in polynomial time [5]. However, for arbitrary workflow nets, the best known algorithm is non-primitive recursive space [43]. An open question is whether the structure of a workflow net can be exploited to improve this algorithm.
2. *How to calculate differences/commonalities of processes?*
   Inheritance is well-defined for static structures such as classes, e.g., a class is a subclass of another class if it has the attributes/methods of the superclass. It is more difficult to compare processes, cf. [19] for a discussion on this topic. The absence of a well-established notion of inheritance makes it difficult to compare processes let alone reason about differences/commonalities of processes. For example, one can have two processes sharing the same set of tasks. This does not imply that one can be substituted by the other. Similarly, it is difficult to define what two processes have in common if they do not agree on the ordering of tasks. A first attempt to reason about differences/commonalities of processes was given in [7]. Unfortu-

nately, existing notions are unsatisfactory since they provide only partial solutions and their computational complexity is unknown. Many practical questions are related to these issues. For example, when harmonizing procedures of different enterprises/govenments it is interesting to calculate what they have in common and list the differences. Therefore, the quest for good notions and algorithms to calculate differences/commonalities of processes is highly relevant.

3. *Which workflow process patterns are the essential ones?*
   The Workflow Management Coalition (WFMC) has standardized basic building blocks such as AND/XOR-splits/joins [36]. Using these building blocks one can model sequential, parallel, conditional, and iterative processes. However, these basic control-flow patterns are unable to directly cope with more advanced patterns. Some of these more advanced patterns are described in [12, 13]. Examples are patterns dealing with multiple instances, e.g., an insurance claim with a variable number of witness statements, and advanced synchronization patterns, e.g., the n-out-of-m join and the OR-join (only synchronize when needed). Some patterns can be modeled quite easily in terms of Petri nets. Other patterns can only be modeled after introducing color sets and spaghetti-like diagrams. After four decades of Petri-net modeling, a mature and complete set of process patterns is still missing and designers are re-inventing the wheel every day. Clearly, most patterns are domain dependent. The set of patterns reported in [12, 13] is specific for workflow management. The question is whether this set is complete. Another question is how frequently these patterns are used/needed in practice.
4. *How to evaluate and compare the expressive power of workflow management systems?*
   Over the last decade, more than 200 different commercial workflow management systems became available. Surprisingly, almost every system uses a propriety language for modeling workflow processes (despite the efforts of the WFMC [36]). Only a handful of systems is based on formal methods such as Petri nets (e.g., COSA by Thiel AG and Income by Promatis). Many systems use a vendor-specific variant of flow-charts extended with parallelism or Petri-net like diagrams such as the event-driven process chains (e.g., ARIS and SAP). Some systems also use very different principles such as false-token propagation (e.g., InConcert by TIBCO and MQSeries Workflow by IBM) and data-driven controls (e.g., FLOWer by Pallas Athena). This makes it very difficult to compare systems. Nevertheless, it is important to have insights in the expressive power of these systems. Note that the term "expressive power" is not used in the traditional sense. The term is used to refer to the effort needed to construct models that reflect the process logic in a direct manner. Note that the patterns mentioned before can be used as a stepping stone for developing a method for evaluating and comparing process-enabled systems such as workflow management systems.
5. *Which class of workflow processes can be rediscovered?*
   Contemporary workflow management systems are driven by explicit process models, i.e., a completely specified workflow design is required in order to enact a given workflow process. Creating a workflow design is a complicated time-consuming process and typically there are discrepancies between the actual workflow processes and the processes as perceived by the management, workers, and ultimately

the workflow designer. Therefore, it is interesting to develop techniques for (re)discovering workflow models based on observations of the real workflow. Starting point for such techniques are so-called "workflow logs" containing information about the workflow process as it is actually being executed (cf. Figure 11). Unfortunately, it is difficult to (re)discover an arbitrary workflow process. All known process mining techniques fail to properly detect non-free-choice structures [22, 44]. An open question is which class of workflow processes (i.e., a subclass of workflow nets) can be rediscovered assuming "complete" logs.

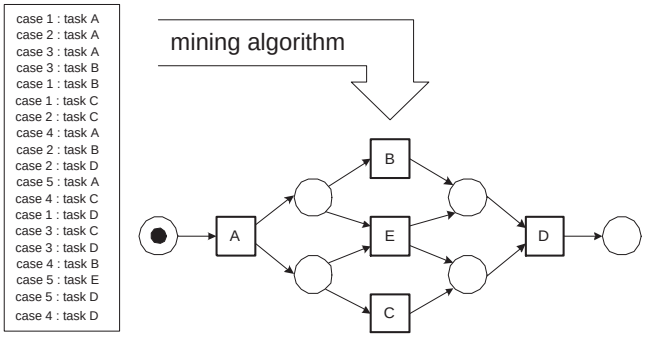

Fig. 11. Using mining techniques it is possible to distill a workflow net (right) from a workflow log (left).

6. *Which are the minimal requirements needed to ensure the correct cooperation between workflow processes?*
   The rise of E-business has increased the number of workflow processes crossing organizational boundaries. As a result, there is a need to connect autonomous workflow processes in such a way that some overall objective is satisfied. A minimal correctness criterion for a set of interoperating workflow nets is soundness. Suppose that each of the local workflow nets is sound. Which requirements should hold in order to guarantee soundness of the overall workflow? In [34] some results are given using scenarios specified in terms of sequence diagrams. Another approach based on inheritance was presented in [15]. Nevertheless, the overall problem remains unsolved.
7. *How to successfully migrate instances while dynamically changing the workflow process?*
   Workflow processes may change on the fly as a result of changing laws, reengineering efforts, etc. This often leads to the situation where cases (workflow instances) have to migrate from the old workflow process to the new workflow process. As was first demonstrated in [25] this may lead to all kind of anomalies (e.g., deadlocks, livelocks, etc.). Consider a case that is moved from a process with A and B in parallel to a process with A and B in sequence. If the case is in a state after executing B but before executing A, there is no corresponding state in the sequential process. This phenomenon is known as the "dynamic change bug" [25]. In [8] it is

demonstrated that these anomalies can be avoided if the old and the new process are in a subclass/superclass relation. In [16] the problem is tackled for workflow processes without loops. Despite these partial solutions, the more general problem remains unsolved.

8. *How to incorporate structural, historical, and actual data to predict the future performance of a process?*
   When it comes to performance analysis, most approaches focus on the steady-state of a process using assumptions about the arrival process (e.g., a Poisson arrival process) and the processing times (e.g., a Gamma distribution). The problem is that such an analysis requires additional modeling, does not exploit the information available, and cannot be used for short-term decisions. Business process management systems contain information about the structure of the process (e.g., a workflow process or business rules), historical data (e.g., transaction logs), and actual data (e.g., the state of each case). In principle, this information can be used to automatically generate performance models to predict throughput times and utilization rates in the near future. A workflow net augmented with predictions about the arrival of new cases, the processing times of tasks, and routing probabilities (note that such settings can be derived from historical data) can be used to generate a simulation model. If this simulation model is initialized using the current marking of the net, the near future can be analyzed. Such an approach requires a mixture of structural, historical, and actual data [40]. One of the problems is that these different kinds of data may overlap or may even be conflicting. Another problem is that most tools are designed for steady-state analysis rather than the analysis of the transient behavior. These and other problems need to be tackled in order to exploit the information in today's business process management systems.
9. *How to calculate the throughput time of concurrent workflow processes?*
   When it comes to the analysis of processes with parallelism, little has changed since the landmark paper by Baskett et al. [18] in 1975. Most of the fundamental problems remain. Queueing networks are particularly suitable for sequential processes [21] and stochastic nets [38] are typically restricted to phase-type distributions and suffer from the state-explosion problem. An alternative approach was presented in [10]. This approach allows for arbitrary discrete-time distributions provided that the workflow net can be constructed using sequential, parallel, conditional, and/or iterative routing. The problem with this approach is that it becomes intractable in case of iteration and long-tailed distributions. Approaches based on simulation are typically time-consuming and do not give exact results. The problem remains that there is no analytical technique which can cope with parallelism and non-phase-type distributions.
10. *How to allocate a fixed set of resources to tasks to optimize performance?*
    Most techniques for performance analysis are of type "What if?" and assume a given distribution of resources over tasks. Clearly, an optimal distribution of resources can be achieved by a "What if?" analysis of all possible resource allocations. Unfortunately, such an approach is intractable for large numbers of resources and tasks. An interesting alternative approach was given by Goldratt [28]. He proposes to first allocate the minimal number resources to each task. This can be determined by simply taking the product of the frequency of a task and the average

processing time. The additional resources are distributed one-by-one using the following mechanism: Determine the bottleneck and allocate one free resource to this bottleneck. Then again determine the bottleneck after adding this resource and allocate the next free resource. This procedure is repeated until all free resources have been allocated. Clearly, this procedure is much more efficient than analyzing all possible resource allocations. Unfortunately, as shown in [30], this procedure does not lead to an optimal distribution. In [30] it is shown that for various interpretations of the term bottleneck (e.g., average queue time and resource utilization) there exist counterexamples for Goldratt's algorithm. Van Hee and Reijers also propose a so-called marginal allocation strategy. This strategy is optimal for state-machine workflow nets with Poisson arrivals and negative exponential service times [30]. Unfortunately, all known strategies fail in the presence of parallelism and/or non-exponential service times.

As indicated before, we do not claim that this list of problems is of the same stature as Hilbert's problems. In fact, the list is heavily biased by personal experiences. Some problems are very concrete (e.g., the complexity of deciding soundness) while others are more general, and therefore, less concrete (e.g., how to allocate a set of resources).

7 Conclusion

In this paper, the application of Petri nets to business process management was discussed. First, business process management was put in its historical perspective. Then, the topics of process design, process analysis, and process enactment were discussed. These topics have been illustrated using Petri nets. To conclude, in the spirit of Hilbert's problems, 10 interesting problems for people working on Petri-net theory have been posed. We hope that the challenges mentioned in this paper will stimulate new and exciting research.

References

1. 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.
2. W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.
3. W.M.P. van der Aalst. Woflan: A Petri-net-based Workflow Analyzer. *Systems Analysis - Modelling - Simulation*, 35(3):345–357, 1999.
4. 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.
5. 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.
6. W.M.P. van der Aalst. *Making Work Flow: On the Design, Analysis and Enactment of Business Processes (inaugural lecture given at 30 November 2001)*. Eindhoven University of Technology, Eindhoven, The Netherlands, 2001.

7. 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.
8. W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.
9. 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.
10. W.M.P. van der Aalst, K.M. van Hee, and H.A. Reijers. Analysis of Discrete-time Stochastic Petri Nets. *Statistica Neerlandica*, 54(2):237–255, 2000.
11. 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.
12. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns Home Page. http://www.tm.tue.nl/it/research/patterns/.
13. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Advanced Workflow Patterns. In O. Etzion and P. Scheuermann, editors, *7th International Conference on Cooperative Information Systems (CoopIS 2000)*, volume 1901 of *Lecture Notes in Computer Science*, pages 18–29. Springer-Verlag, Berlin, 2000.
14. W.M.P. van der Aalst and S. Jablonski, editors. *Flexible Workflow Technology Driving the Networked Economy*, Special Issue of the International Journal of Computer Systems, Science, and Engineering, volume 15, number 5. CRL Publishing Ltd, 2000.
15. W.M.P. van der Aalst and M. Weske. The P2P approach to Interorganizational Workflows. In K.R. Dittrich, A. Geppert, and M.C. Norrie, editors, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume 2068 of *Lecture Notes in Computer Science*, pages 140–156. Springer-Verlag, Berlin, 2001.
16. A. Agostini and G. De Michelis. Improving Flexibility of Workflow Management Systems. 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 218–234. Springer-Verlag, Berlin, 2000.
17. J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, Cambridge, 1990.
18. F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open, Closed and Mixed Networks of Queues with Different Classes of Customers. *Journal of the Association of Computing Machinery*, 22(2):248–260, April 1975.
19. T. Basten and W.M.P. van der Aalst. Inheritance of Behavior. *Journal of Logic and Algebraic Programming*, 47(2):47–145, 2001.
20. G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley, Reading, MA, USA, 1998.
21. J.A. Buzacott. Commonalities in Reengineerd Business Processes: Models and Issues. *Management Science*, 42(5):768–782, 1996.
22. 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.
23. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
24. 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.
25. 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.
26. 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.
27. J.W. Forrester. *Industrial Dynamics*. MIT Press, Cambridge, MA, 1968.
28. E.M. Goldratt and J. Cox. *The Goal*. Gower, Aldershot, UK, 1984.
29. D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.
30. K.M. van Hee, H.A. Reijers, H.M.W. Verbeek, and L. Zerguini. On the Optimal Allocation of Resources in Stochastic Workflow Nets. In K. Djemame and M. Kara, editors, *Proceedings of the Seventeenth UK Performance Engineering Workshop*, pages 23–34. University of Leeds, Leeds, UK, 2001.
31. D. Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8:437–479, 1902.
32. 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.
33. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
34. 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.
35. M. Klein, C. Dellarocas, and A. Bernstein, editors. *Adaptive Workflow Systems*, Special Issue of Computer Supported Cooperative Work, 2000.
36. P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.
37. P. Lyman and H. Varian. How Much Information. http://www.sims.berkeley.edu/how-much-info.
38. M. Ajmone Marsan, G. Balbo, and G. Conte et al. *Modelling with Generalized Stochastic Petri Nets*. Wiley series in parallel computing. Wiley, New York, 1995.
39. C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.
40. H.A. Reijers and W.M.P. van der Aalst. Short-Term Simulation: Bridging the Gap between Operational Control and Strategic Decision Making. In M.H. Hamza, editor, *Proceedings of the IASTED International Conference on Modelling and Simulation*, pages 417–421. IASTED/Acta Press, Anaheim, USA, 1999.
41. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
42. W. Sadiq and M.E. Orlowska. Analyzing Process Models using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.
43. 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.
44. T. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data. In V. Hoste and G. de Pauw, editors, *Proceedings of the 11th Dutch-Belgian Conference on Machine Learning (Benelearn 2001)*, pages 93–100, 2001.
45. Woflan Home Page. http://www.tm.tue.nl/it/woflan.
46. M.D. Zisman. *Representation, Specification and Automation of Office Procedures*. PhD thesis, University of Pennsylvania, Warton School of Business, 1977.