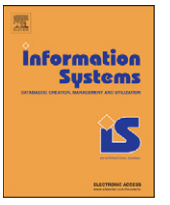
Information Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys



Research Highlights

Product-based workflow support ☆

Irene Vanderfeesten *, Hajo A. Reijers, Wil M.P. van der Aalst

Eindhoven University of Technology, Department of Industrial Engineering and Innovation Sciences, PO Box 513, NL-5600 MB Eindhoven, The Netherlands

► Product-Based Workflow Support is a new and scientifically grounded approach that provides flexible and dynamic support for workflow processes. ► Product-Based Workflow Support takes a product-oriented view on process optimization using the notion of the product Product Data Model. ► The feasibility of Product-Based Workflow Support is demonstrated by a workflow system that is realized using ProM and DECLARE. ► To support end-users in making decisions for executing the workflow process, the system provides them with recommendations for the best step to be taken next.

0306-4379/$ - see front matter
doi:10.1016/j.is.2010.09.008

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

Information Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Product-based workflow support ☆

Irene Vanderfeesten *, Hajo A. Reijers, Wil M.P. van der Aalst

Eindhoven University of Technology, Department of Industrial Engineering and Innovation Sciences, PO Box 513, NL-5600 MB Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 16 June 2010
Received in revised form
27 September 2010
Accepted 28 September 2010
Recommended by: D. Shasha

Keywords:
Business process modeling
Workflow management
Product data model

ABSTRACT

Despite the industrial need for the improvement of information-intensive business processes, few scientifically grounded approaches exist to support such initiatives. In this paper, we propose a new approach that builds on concepts that are part of a product-oriented view on process optimization. Essentially, this approach allows end users to flexibly decide on the best possible way to create an informational product within the limits that are imposed by regulations and logical dependencies. We argue that this provides various benefits in comparison to earlier work. To support end users in making sensible decisions, we describe two alternative approaches to provide them with recommendations to this end. We formalize these alternatives and discuss their relative strengths and weaknesses. The feasibility of the overall approach, which we refer to as Product-Based Workflow Support, is demonstrated by a workflow system that is realized using ProM and DECLARE.

1. Introduction

Contemporary management concepts such as "operational excellence," "lean management," and "business process redesign" all stress the importance of smoothly running business processes. It seems a natural angle from which to consider *processes* – complete chains of operations that are needed to produce certain products or services – to make organizations perform better. Unsurprisingly, market analyses consistently identify the improvement of business processes as the top business priority for CIO's [22–24].

Given the importance of business processes and their tight relation to organizational performance, it may come as a surprise that few scientific approaches are available that address the issue of how to actually *design* a process or, since in many contexts the processes are already in place, how to *redesign* one. The best-known references are situated in the domain of the popular management literature, e.g., [11,14,17]. Understandably, it is often said that process design is "more art than science" [44,45].

One of the notable exceptions is Product-Based Workflow Design (PBWD) [37]. PBWD has been developed in close cooperation between academic and industrial parties to arrive at a method for process redesign that is repeatable, objective, and effective. Its focus is on the design of processes that deliver informational products, the so-called *workflow processes*. Since its conception, PBWD has been adopted by consultancy and service companies to improve the performance of various business processes in the services domain [35,36].

Highly characteristic for PBWD is that it aims first and foremost at developing a deep understanding of the characteristics of the *informational product* that is to be delivered, e.g., a particular type of decision, proposal, permit, etc. which is laid down in a *product data model*. This is subsequently used by the designer to determine the best process structure to create and deliver that product. Given that there are generally alternative ways to produce an informational product, PBWD discloses all the opportunities to produce a product.

At this stage, considerable experience has been gained with the application of PBWD in practice. Aside from the

☆ This research is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs.

* Corresponding author. Tel.: +31 40 247 4366; fax: +31 40 243 2612.
E-mail address: i.t.p.vanderfeesten@tue.nl (I. Vanderfeesten).

0306-4379/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2010.09.008

tangible business benefits that PBWD has delivered, it has become apparent that the "product" notion is an extremely viable concept to reason about workflow processes. The shift of attention to what is the desired *outcome* of a workflow process without directly discussing *how* this is achieved leads to an interaction with stakeholders that quickly converges. This sharply contrasts with the problems that are often associated with process improvement projects, such as the confusion about what actions in the current process are really necessary and which ones are merely motivated by tradition [14,41]. Interestingly, IBM's recent artifact-centric approach takes a similar indirect route by first considering the objects that are manipulated in a process before the focus moves to the actual process design [6].

At the same time, it must be acknowledged that the translation of the product data model to a favorable workflow is a critical step. In the first applications of PBWD this derivation was done manually [36, pp. 256–273]. Since this is time-consuming and error-prone, we have been developing IT tools to support the administration of a product data model, as well as algorithms that automatically generate workflow designs on the basis of a product data model [46]. Still, business users find it difficult to consider and compare all the options that are available for the final workflow design. The reasons for this are that, in general, there are many of these that may differ in subtle ways: While some options may work well for some cases, they may not do so for others.

This paper presents an entirely new outlook on the use of the product data model. Instead of aiming at the derivation of a workflow design that is generally the best possible way to generate an informational product, the product data model *itself* is proposed as the vehicle to steer a workflow's execution. In other words, the need to translate a product data model into a workflow design disappears. Instead, a business user determines on a case-by-case basis the best possible way to create an informational product in accordance with the relevant product data model. This approach addresses the difficulty for business users to compare many alternative workflow designs, while it still relies on the product data model with its attractive properties. In addition, this approach allows for a highly dynamic and case-specific execution of workflows, as will be illustrated in the remainder of this paper.

The proposed approach builds on two pillars. First, we exploit the wide industrial proliferation of "process-aware" or "process-oriented" information systems [13,26]. We will assume the existence of such type of system to support the proposed approach and, along the way, show the feasibility of this idea in the form of a prototype workflow system. Second, the offered solution rests on the idea that it is easier for a business user to determine the best possible action in the context of processing a *single* case versus a *general* case. To guide the business user in this respect, we present two alternative approaches to provide her with recommendations: one that is optimal in relation to a dominant performance criterion but rather computing-intensive, and another that is computationally lightweight but based on heuristics.

Our contribution can be summarized as follows. We present a rigorous approach for business process improvement, which addresses the need for guidance in this respect from practice. The innovative aspect is that we do not aim at the design of an underlying generic process; instead, we provide a business user with direct support for delivering an informational product in a performative way. To do so, we build on the successful notions from PBWD, in particular the product data model, in order to arrive at a method that we coin "Product-Based Workflow Support" (PBWS). In this approach, the product data model specifies the elements to assemble a particular product, while a process-aware information system suggests how a business user should use these to deliver the product in the best possible way.

The structure of this paper is as follows. Section 2 contains background information, a running example and the motivation for PBWS. Next, Sections 3 and 4 present the two alternative realizations of the envisioned support. A comparison of these two approaches is given in Section 5, followed by a description of a workflow system (based on ProM and DECLARE) to support the overall approach provided in Section 6. The paper ends with related work and conclusions.

2. Background and motivation

This section provides information that is essential as background for the remainder of the paper. In particular, the product data model is explained and illustrated with an example. We also provide motivation for the idea of PBWS.

2.1. Workflow products

The product of a workflow process is an *informational* product, e.g., a decision on an insurance claim, the allocation of a subsidy, or the approval of a loan. Based on the input data provided by the client or retrieved from other systems, the process constructs the end product step-by-step. In each step new information is produced based on the specific data present for the case.

In this paper, we use a financial workflow process as a running example. The workflow process deals with the calculation of the maximum amount of mortgage a bank is willing to loan to a client. The bank has three alternative ways to decide on the maximum mortgage. First of all, if the client has a negative registration in the central register for credits (e.g., the client has a history of non-payment), the bank may directly deny this person a mortgage (leading to a maximum amount of zero). The central credit register keeps track of all loans a person has and helps providers of loans in their assessment of the creditworthiness of such a person.

Second, if the client has previously requested a mortgage offer and the term of the validity of this offer is not yet expired, this may determine the amount of the mortgage. Typically, the percentage of interest changes over time, and a mortgage offer is valid for some months. In case the interest has increased since the previous offer,

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

the valid offer might be better than a new one which is based on the higher interest percentage.

Finally, if the credit register shows a positive credit history, the bank needs more information on the client's situation (e.g., gross income, type of mortgage) in order to decide on the maximum mortgage. To a certain extent, a bank defines its own internal rules and policies to decide on how high a risk a mortgage applicant is to their business if they approve a mortgage. Therefore, each bank uses a percentage of the gross income of the client to calculate how much money the client is allowed to spend on the house. With this rule, a bank ensures that the client can afford the costs of food and recurring expenses, and that the probability is high that he will meet the monthly payment liabilities with the bank. However, this percentage is not fixed and can vary, based on the bank's current situation as well as that of the client's.

In this example the maximum amount of mortgage is the end product of the workflow process. In other words, this is the piece of information that is "produced." A graphical representation of the structure of this workflow product is given in Fig. 1 and is explained below.

2.2. The product data model

The structure of a workflow product is captured by a *Product Data Model* (*PDM*). The PDM is described by a tree-like structure similar to a Bill-of-Materials [27]. The information that is processed in the workflow process is described by the so-called *data elements* of the PDM. For each specific case a data element can have a different *value*. Data elements are depicted as circles in the PDM, as can be seen in the example of Fig. 1.

The actions that can be taken on the data element values are called *operations* and are represented by (hyper)arcs. Each operation may have zero or more *input data elements* and produce exactly one *output data element*. The arcs are "knotted" together when values for a set of data elements are needed to execute the particular operation. Fig. 1 shows a comparison of the arcs from *B*, *C*, and *D* leading to *A*, on the one hand, and the arc leading from *E* to *A*, on the other hand. In the latter case, only one data element value is needed to determine the outcome of the process, while in the case of *B*, *C*, and *D*, all three data element values are needed to produce *A*. An operation is said to be *executable* for a case when values relating to that case are available for all of the operation's input elements.

Several operations can have the *same* output element while having a different set of input elements. Such a situation represents *alternative ways* to produce a value for that output element. For example, a value for the end product *A* in Fig. 1, can be determined in three alternative ways: (i) based on a value for *E*, (ii) based on a value for *H*, and (iii) based on values for *B*, *C*, and *D*.

The top element of the PDM, i.e., the end product, is called the *root* of the PDM. The *leaf elements* are the elements that are provided as inputs to the process. They are produced by operations with no input elements (e.g., the operations with output elements *B*, *D*, *E*, *F*, *G*, and *H*). The operations producing values for the leaf elements are denoted as *leaf operations* or input operations. An operation can be identified by a tuple consisting of the output element and a set of input elements, e.g., (*A*, {*B*, *C*, *D*}) for the operation producing a value for *A* based on data elements *B*, *C*, and *D*. Throughout this paper, operations are also referred to by identifiers, such as *Op*01.

The construction of a PDM is a manual task. However, the information needed to construct a PDM can be obtained from e.g., rules and regulations, work instructions, forms, textual product descriptions, jurisprudence, information systems and the knowledge on the workflow product that is present with stakeholders.

Fig. 1 expresses the PDM for the mortgage example. It shows that the maximum mortgage (element *A* in Fig. 1) is dependent either on a previous mortgage offer (*E*), or on the registration in the central credit register (*H*), or on

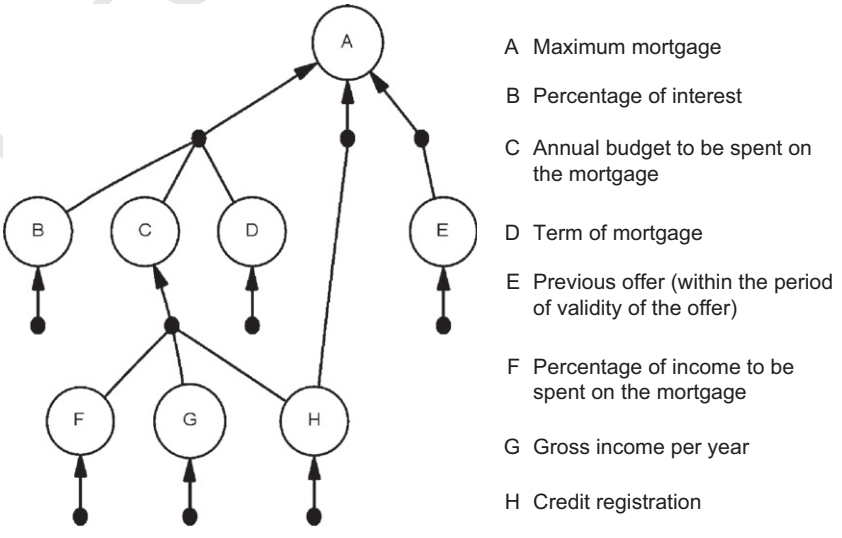


Fig. 1. The PDM of the maximum mortgage calculation.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

the combination of the percentage of interest (B), the annual budget to be spent on the mortgage (C), and the term of the mortgage (D). The annual budget (C) is determined by the gross income of the client per year (G), the credit registration (H), and the percentage of the income the client is allowed to spend on paying the mortgage (F).

2.3. Motivation

According to the PBWD method, the PDM is used as the basis for designing a process that, on the one hand, respects the dependencies between the various data elements, and, on the other hand, provides a performative "walk through" along the various operations that need to take place [37]. It is important to stress that with PBWD such a process is proposed as a *generally* attractive way to deal with all thinkable instances of the product that the PDM relates to.

Both for the derivation of a general process (in accordance with PBWD) and for Product-Based Workflow Support (as proposed in this paper) the availability of various alternatives to achieve a particular outcome is exploited. Operations that create new values for data elements can take on different forms (e.g., an automatic calculation, an assessment by a human, or a rule-based decision), but all will consume time and money. In general, an operation can have a number of attributes associated with it that describe the characteristics of the operation in more detail:

- *Execution cost*: The cost associated with executing the operation (given by a probability distribution and its parameters).
- *Processing time*: The time that is needed to complete the operation (given by a probability distribution and its parameters).
- *Execution conditions*: Conditions on the value of the input data elements that restrict the execution of the operation. If the condition is not satisfied, the operation is not executable, even if a value for all of its input data elements is available.
- *Failure probability*: The probability that the operation is not performed successfully, i.e., the probability that the output data element is not produced.
- *Resource class*: The resource class or role that is required to perform the operation.

If we again consider our example, it could occur, for instance, that the cost of executing alternative operations differs. The involved bank has to pay for receiving a copy of the client's registration in the credit register. Thus, the cost for this operation is higher than, for example, the cost of determining the gross income, since the information on income is provided by the client. The same holds true for the processing time of these operations: Since an external party is consulted, it may take more time to retrieve the credit registration than to ask the client for her gross income.

To illustrate the notion of PBWS, we present with Fig. 2 a step-by-step execution of the operations that occur in a PDM.[1] Suppose that the values for the leaf elements B, D, F, G, and H (i.e., the interest percentage, term of mortgage, percentage of income to be paid in rent) are available at the start of the process *for one particular case* (see Fig. 2(b)).[2] The operations that are now enabled for execution for this specific case are $Op02$ and $Op03$, since a value for all of their input elements is available (Fig. 2(c)). Operation $Op01$ is not executable because a value for data element C is not available yet, and $Op04$ is not executable, since there is no value for E present. We now have to choose between the two executable operations ($Op02$, $Op03$). Suppose we select $Op02$. Then, a value for data element C is produced (Fig. 2(d)). The executable operations are calculated again, i.e., $Op01$ and $Op03$, and one of these two operations is to be selected next. Suppose we select $Op01$ (Fig. 2(f)). Then, the value for the end product A is determined and the process ends.

The example illustrates that more than one operation may be executable at a certain point in time. For example, in the first step of this example we could have chosen for $Op03$ instead of $Op02$. This would have led to the end product (A) immediately, but it could also have generally resulted in a different execution of the process with e.g., different total cost, throughput time, etc. For example, suppose we take the execution cost and processing time for the operations of the mortgage example, as given in Table 1. If we focus on the cost of execution, it seems best to choose $Op02$ since the execution cost for $Op02$ is lower than the execution cost for $Op03$ (5.0 vs. 9.0). However, if we consider the processing times for the operations, selecting $Op03$ as a next step would perhaps be a better decision (4.0 for $Op02$ vs. 3.0 for $Op03$). It would be extremely valuable if at any point during the execution of operations these insights were to be available.

The idea of Product-Based Workflow Support can now be described as follows. *An end user is supported on a case-by-case basis with recommendations on the most suitable way to carry out the available operations in a PDM.* A recommendation takes into account what type of performance is pursued for each case and how the various alternative executions differ from each other with respect to that performance criterion. Note that the general process model (most suitable to deal with the average case) as prescribed by the PBWD method, becomes superfluous when using PBWS. As an alternative, the end user is guided through the operations of the PDM in a way that is both flexible and performative.

The remainder of the paper deals with the question of how to determine proper recommendations, i.e., how to select in each possible situation the best operation to proceed with from a set of executable operations. It

[1] For reasons of simplicity, we abstract here from the execution conditions on the operations. Moreover, in Fig. 2 we assume that all leaf operations have already been executed in the initial state, i.e., a value is available for the output data elements of all successfully executed leaf operations.

[2] Note that for this case not all leaf operations have been executed successfully. Because the execution of $Op07$ has failed, there is no value for data element E available, i.e., there is no valid previous offer available.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

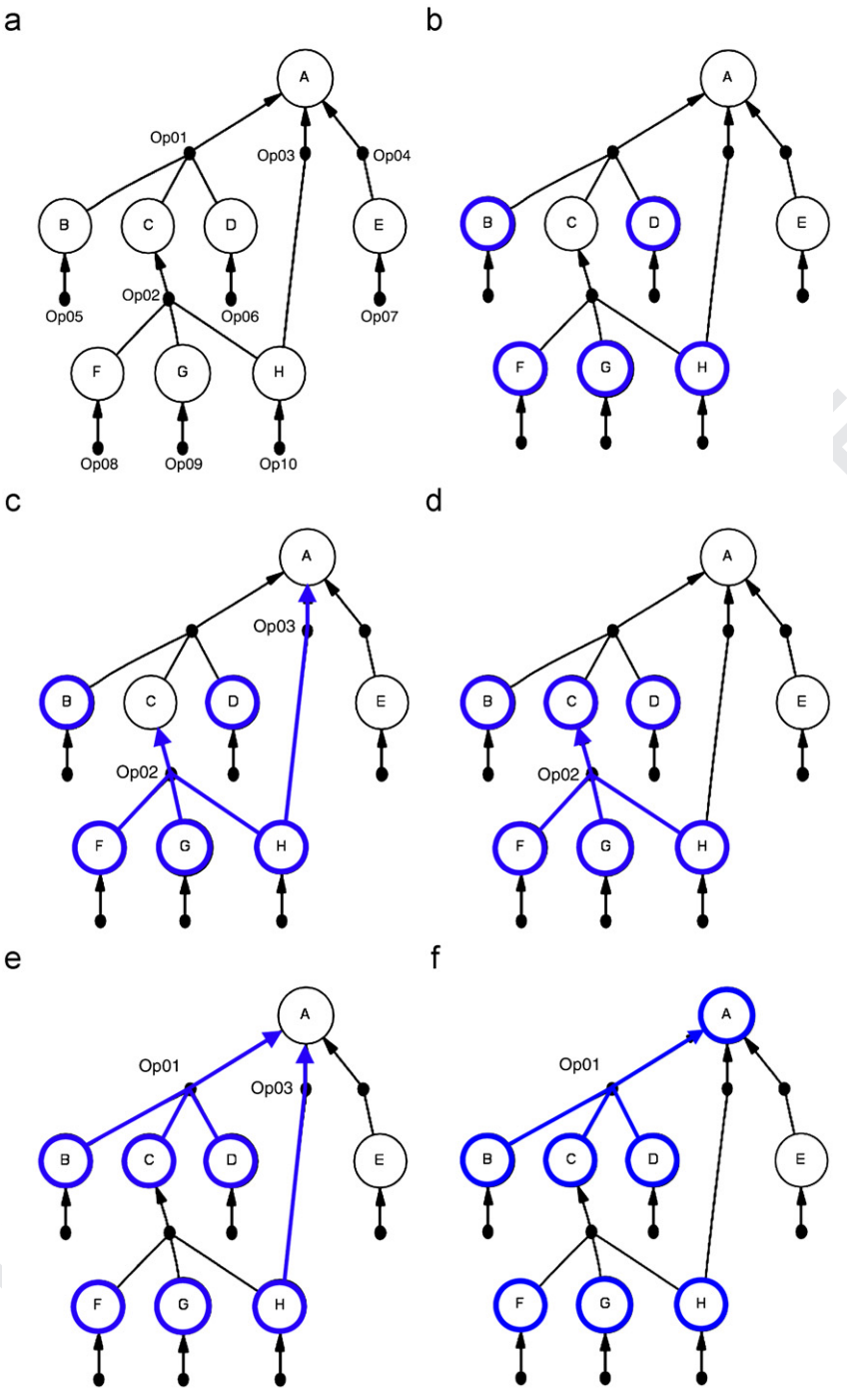


Fig. 2. The step-by-step execution of the PDM for the mortgage example. Bold circles represent available data element values for the case under consideration; bold arrows indicate executable operations. (a) The PDM for the mortgage example. (b) The values for some of the leaf data elements (*B*, *D*, *F*, *G*, *H*) are available (indicated by bold circles). Throughout this paper, we refer to this situation as the *initial state*. (c) Executable operations in the first step: *Op*02 and *Op*03. (d) The value for data element *C* is produced by operation *Op*02. (e) Executable operations in step two: *Op*01 and *Op*03. (f) The value for the end product (*A*) is determined by executing operation *Op*01.

should be stressed that we define "best" in the context of a single case, i.e., the performance goal (e.g., total cost, total processing time) of the case in isolation is optimized. Two main solution approaches for selecting candidate operations are presented in the next two sections.

3. Global decision strategies

The first solution approach aims at creating guaranteed optimal walkthroughs of the PDM by using global decision strategies. A global decision strategy takes into account the

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

Table 1
Operations and their attributes for the mortgage example.

	Output	Input	Cost	Time	Prob.	Conditions
*Op*01	*A*	*B, C, D*	5.0	1.0	0.05	–
*Op*02	*C*	*F, G, H*	5.0	4.0	0.05	–
*Op*03	*A*	*H*	9.0	3.0	0.05	Value(*H*) = "negative"
*Op*04	*A*	*E*	2.0	2.0	0.00	–
*Op*05	*B*	–	0.0	0.0	0.00	–
*Op*06	*D*	–	0.0	0.0	0.00	–
*Op*07	*E*	–	1.0	1.0	0.50	–
*Op*08	*F*	–	0.0	0.0	0.00	–
*Op*09	*G*	–	0.0	2.0	0.00	–
*Op*10	*H*	–	3.0	10.0	0.15	–

effect of the current decision on future decisions, i.e., the overall performance of the case. For example, in the first step of the mortgage case, choosing *Op*02 enables *Op*01 in the next step and determines an alternative path to the path containing only *Op*03 in order to produce the end product *A*. This effect is considered when determining a recommendation for the current situation. A global decision strategy takes the complete, alternative path to the end product into account to optimize the overall performance of the case.

The technique we use to determine a walkthrough of the PDM using a global perspective is based on the theory of *Markov Decision Processes* (MDPs) [33,43]. A *Markov Decision Process* (*MDP*) extends the notion of a Markov chain with decisions. Markov chains are mathematical models of stochastic processes, i.e., systems that evolve over time [43]. They are widely used in, e.g., operations research, biology, and computer science. MDPs provide a means for modeling decision making in situations that are partly random and partly under the control of a decision maker. In an MDP, several decisions can be taken in each state of the Markov chain. The state transitions in the Markov chain are dependent on these decisions, and are given by the transition matrix. Each decision has associated costs which depend on the state of the system. The goal of an MDP is to find a *strategy* that specifies which decision to take in each state in order to minimize the overall cost [43]. Note that an important characteristic of a Markov chain is that it is *memoryless*, i.e., the probability of being in a certain state at time $n+1$, is only dependent on the state at time n and not on earlier states ($<n$). Also note that the formalism of a Markov chain is highly similar to the concept of a (probabilistic) Markov state machine [18,21]. Markov state machines may in fact be used to represent Markov chains. This means that they have the same expressive power as Markov chains. For our application, however, the specification of probabilities on the state transitions in the Markov chain or Markov state machine is not sufficient. We also would like to model the decisions that can be made by the user as special cases of the state transitions. Therefore, we use the MDP formalism.

3.1. Formulation of the Markov decision process

The execution of a PDM can be described as a memoryless stochastic process with decisions. We now present the formulation of an MDP based on the PDM to elaborate on this mapping. In general, an MDP is defined by: (i) the state space, (ii) the time space, (iii) the decision space, (iv) the transition function, and (v) the cost functions. A definition of these components is given in Definition 3.1.

Definition 3.1 (*Markov decision process*). An MDP is represented by a tuple (S, T, A, P, c, q) where

- S is the finite state space ($S=\{0,1,2,\ldots,M\}$).
- T is the set of discrete time points with a finite horizon ($T=\{0,1,2,\ldots,N\}$).
- A is the finite decision space. In each state, $i \in S$, the set of decisions that can be taken (A_i) is a subset of the decision space (i.e. $A_i \subseteq A$).
- P is the transition function: $p_{ij}(a)$ is the probability that decision a in state i leads to state j at the next time point.
- c is the immediate cost function such that $c^a(i)$ is the immediate cost received in state i when decision a is taken:

$$c^a(i) := s_i + \sum_{j \in S} p_{ij}(a) \cdot cst^a(i,j),$$

  where s_i denotes the sojourn cost of being in state i for one time step and $cst^a(i,j)$ represents the cost whenever the process is in state i, decision a is taken and the process moves to state j.
- q is the final cost function. $q(i)$ denotes the final cost when the process finishes in state i at the time horizon $t=N$.

At each time point ($n \in T$) the process is in one of the states ($i \in S$). Then, a decision from the set A_i has to be taken and the process moves to another state ($j \in S$). We are interested in choosing the decisions that should be taken in a specific state at a specific point in time, such that the expected total cost over a finite period is minimized. A prescription for taking decisions is called a *strategy*.

To show that the decision problem in the execution of a PDM can be translated into an MDP, we describe for each component of the MDP the corresponding part in a PDM. Furthermore, a simple example, based on the mortgage process, is used to clarify this mapping.

State space: The state space (S) describes the states the process can be in. The states of the execution of a PDM can be described by the operations that have been executed (either successfully or unsuccessfully) together with the data elements for which a value is available. A state in the state space is therefore represented by a tuple consisting of three sets: (i) the successfully executed operations, (ii) the unsuccessfully executed operations, and (iii) the data elements for which a value is available.[3] In Fig. 3(b),

[3] Note that the set of available data elements in a state actually gives redundant information, since the set of available data elements can also be determined based on the set of successfully executed operations. However, this information is added for reasons of clarity and understandability of the examples.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

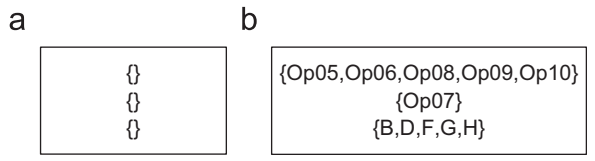


Fig. 3. A state in the state space is described by three sets: (i) the operations that have been executed successfully so far, (ii) the operations that have been executed unsuccessfully, and (iii) the data elements for which a value is available. (a) The state in which no data element values are available yet, cf. the initial state shown in Fig. 2(b). (b) The state in which the values for data elements *B*, *D*, *F*, *G*, and *H* are available, cf. Fig. 2(b).

for example, the state in the mortgage example in which a value for each of the leaf elements *B*, *D*, *F*, *G*, and *H* is available (cf. Fig. 2(b)) is denoted by ({*Op*05, *Op*06, *Op*08, *Op*09, *Op*10}, {*Op*07}, {*B*, *D*, *F*, *G*, *H*}). Thus, $S \subseteq \mathcal{P}(O) \times \mathcal{P}(O) \times \mathcal{P}(D)$. The state space of the MDP is finite since the number of operations is finite, and we assume that each operation can only be executed once, either successfully or unsuccessfully.

Time space: The time space describes the time points at which a decision is taken and at which a state transition occurs (also called *decision epochs*). The times used in our MDP are discrete and can be represented by the number of executed operations ($T = \{0,1,2,3, \ldots, |O|\}$), i.e., time is indicated by $t \in T$ indicating the number of executed operations. The times are not necessarily equidistant, but since there is a finite number of operations in a product data model and we assume no concurrency, the time space is bounded by this number, i.e., the MDP problem has a finite time horizon.

Decision space: In each state a number of decisions can be made. For the execution of a PDM, these decisions are described by the set of operations that are executable in the current state of execution (i.e., those operations of which the input elements are available and that have not yet been executed).[4] Moreover, if there are no executable operations for a certain state, there is only one decision possible, i.e., to stop. Thus, the decision space *A* is equal to the set of operations plus the decision to stop, i.e., $A = (O \cup \{stop\})$. Furthermore, the decision space in a particular state A_i is a subset of $(O \cup \{stop\})$. Note that the decision space in a certain state is time-independent, i.e., given a state, at any point in time, the same decisions can be made.

Transition probabilities: The transition probabilities are given by a matrix *P* that describes the probabilities that the system moves from the current state to any of the other states in the system. These transition probabilities are dependent on the decision that was made. For our application, a decision *a* in state *i* can lead to two new states: j_1 (for a successful execution of the operation given in decision *a*) and j_2 (for a failed execution of the operation given in decision *a*), each with their own probabilities. The two transition probabilities under decision *a* always add up to 1. For example, recall the execution of the mortgage example in Fig. 2. If we start in the state with available values for data elements *B*, *D*, *F*, *G*, and *H*, there are two decisions which can be taken, i.e., $\{Op02, Op03\} \subseteq O$. Each of these two operations can either fail or be successfully executed. This leads to four new states (Fig. 4). The transition probabilities correspond to the probabilities of failure or successful execution of an operation (see Table 1):

$$p_{1,3}(Op02) = 0.95, \quad p_{1,2}(Op03) = 0.95,$$
$$p_{1,5}(Op02) = 0.05, \quad p_{1,4}(Op03) = 0.05.$$

Thus, the probability of moving from state 1 to 3 under decision *Op*02 is 0.95.

Immediate and final cost: The immediate cost of a transition from states *i* to *j* under decision *a* is the cost of executing operation *a*. In our model there are no costs associated with residing in a state, i.e., costs are associated with decisions. Depending on the performance objective of the process, e.g., minimization of cost or processing time, the immediate costs are defined as the cost or the processing time of the operation. Suppose we want to minimize the execution cost of the process, then the immediate costs in state 1 are (see Table 1):

$$c^{Op02}(1) = 5.0, \quad c^{Op03}(1) = 9.0.$$

The final cost incurred at time $|O|$, when no decisions can be made anymore, in state *i* is zero, i.e., $q(i)$=0.

By using the mapping presented above, the process of executing a PDM can be translated to a time-homogenous MDP with a finite horizon. Note that the optimal strategy for the MDP results in a Markov chain with some special properties. First, the state space is *finite*, i.e., there is a finite number of states. Second, each operation can be executed at most once. Thus, each decision can also be taken at most once during any execution. Moreover, at each point in time an operation is selected until there are no executable operations anymore. The only decision that can be made then is to stop, and the system stays in the same state. These end states are *absorbing* states. The Markov chain is a *transient* chain since each closed class has exactly one absorbing state. Apart from the absorbing end states, there are *no cycles* in the Markov chain, i.e., it is not possible to return to a state that was previously visited.

3.2. The mortgage example as an MDP

We have used a small part of the mortgage example to illustrate the mapping from a PDM to an MDP (see Fig. 4). Below, this example is used to demonstrate how a stationary Markov decision strategy (*f*) is determined for the complete mortgage example.

The initial situation (see Fig. 2(b)) in which there is already a value for data elements *B*, *D*, *F*, *G*, and *H* is used again because in this case the decision space is rather small. Therefore, the state space for this MDP problem stays small and readable. The state space is depicted in Fig. 4 and describes all possible execution steps from the initial state. The decision space per state can also be derived from the figure, e.g., in state 3 two alternative decisions exist: *Op*01

[4] Throughout this paper we abstract from concrete values of data elements and conditions based on these values. Incorporating data are possible if the number of possible values is small. If many values are possible, the state space will be too large to allow for any form of analysis.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

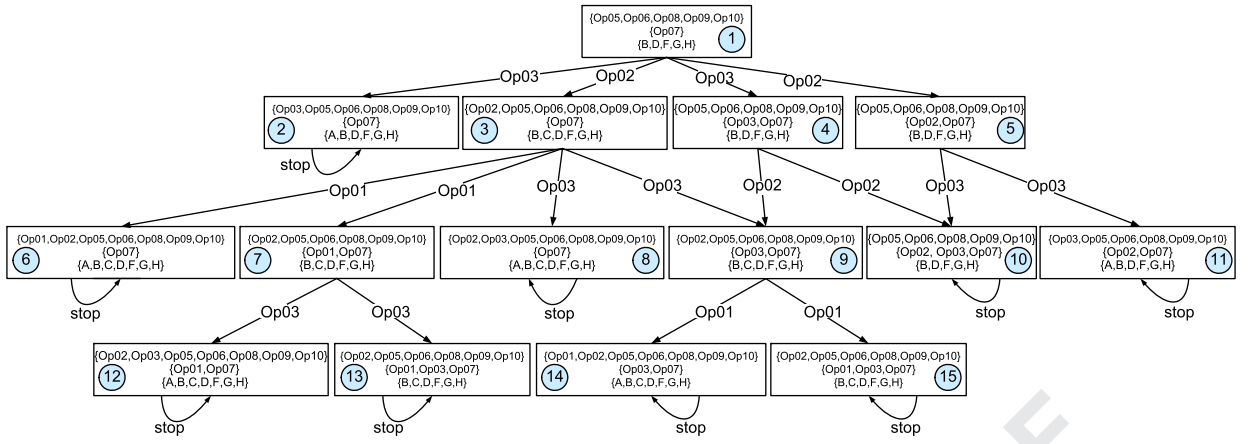


Fig. 4. The state space of the mortgage example. Note that there are ~~eight~~ 9 end states: {2, 6, 8, 10, 11, 12, 13, 14, 15}.

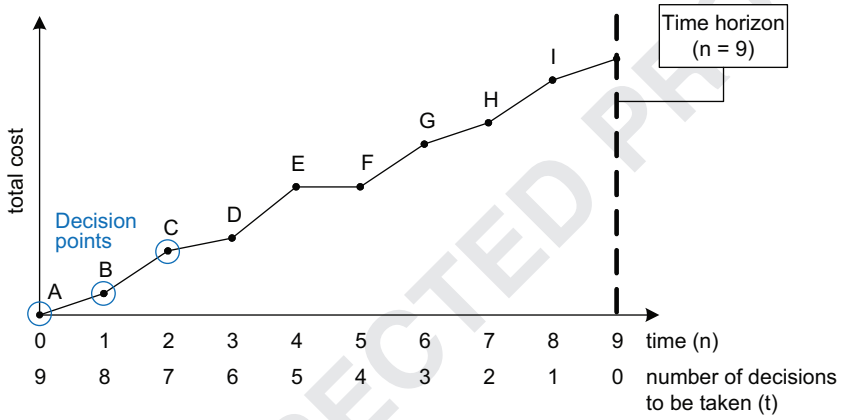


Fig. 5. This figure describes a time line of an MDP with a finite time horizon. The process ends at time n=9. This point is either represented by the time, i.e. n=9, or by the number of decision epochs to the end of the process, i.e. t=0.

and *Op*03. Because the execution of both operations can either be successful or unsuccessful, state 3 has four outgoing arcs to four new states. If operation *Op*01 is executed successfully, the system moves from state 3 to 6. Confronted with an unsuccessful execution of *Op*01, the system moves to state 7. Similarly, upon execution of *Op*03, the system moves to state 8 or 9. For this example we focus on minimizing the total cost for a case. The cost for executing an operation can be found in Table 1.

The optimal decision strategy for this MDP can be calculated using a value iteration algorithm [43]. The algorithm is defined as follows:

Definition 3.2. Let $V_0(i) := q(i)$, and $V_t(i)$, t=1,2,… be recursively given by

$$V_t(i) := \min_{a \in A_i} \left[c^a(i) + \sum_{j \in S} p_{i,j}(a) \cdot V_{t-1}(j) \right].$$

Then, any decision rule f_t determined by

$$f_t(i) = \arg\min_{a \in A_i} \left[c^a(i) + \sum_{j \in S} p_{i,j}(a) \cdot V_{t-1}(j) \right]$$

gives the optimal decision for each state in the state space and achieves the minimal cost over t periods to the time horizon.

The value iteration algorithm uses backward induction to derive the decision rules step-by-step. First, the cost at the time horizon (see Fig. 5) is determined. The time horizon is the point in time at which the process ends and no decisions can be made anymore. Then, a decision rule (f_1) is calculated for the last decision step in the process, i.e., the situation in which one decision can be made (t=1). This calculation is based on the expected cost at t=0. Subsequently, the decision rules for t = 2,3,… can be determined. Below, the first steps of this algorithm are elaborated upon for our MDP. $V_t(i)$ denotes the total expected cost at decision epoch t and state i. The cost at decision epoch t=0 is equal to the final cost, i.e.

$$V_0(i) = 0, \quad \forall i \in S.$$

Next, the values for t=1 can be calculated based on all $V_0(i)$ values. For instance, two decisions can be made in state 3: *Op*01 and *Op*03. The expected cost for choosing *Op*01 is dependent on the immediate cost for choosing *Op*01 in state 3, and the expected cost and transition

probabilities for the states the process can move to under decision *Op*01:

$$c^{Op01} + p_{3,6}(Op01) \cdot V_0(6) + p_{3,7}(Op01) \cdot V_0(7) = 5.0 + 0.95 \cdot 0.0 + 0.05 \cdot 0.0 = 5.0.$$

Similarly, the expected cost in state 3 for decision *Op*03 can be calculated:

$$c^{Op03} + p_{1,2}(Op03) \cdot V_0(2) + p_{1,4}(Op03) \cdot V_0(4) = 9.0 + 0.95 \cdot 0.0 + 0.05 \cdot 0.0 = 9.0.$$

Since decision *Op*01 has the lowest expected total cost, it is the best decision. Thus, if the process is in state 3 and there is only one decision to be taken until the end of the process, the best decision is *Op*01, i.e., $f_1(3)$ = *Op*01, with expected total cost of 5.0, i.e., $V_1(3)$ = 5.0. In the same way, the expected cost and best decisions for all other states at time t=1 can be calculated. Tables 2 and 3 show that if there is only one time period left, it is best to choose *Op*02 in state 1 since $V_1(1)$ has the minimal value of 5.0 (vs. 9.0) for its argument *Op*02. Also, *Op*02 should be chosen in state 4 ($V_1(4)$=5.0 for *Op*02), etc. In states 2, 6, 8, 10, 11, 12, 13, 14, and 15 the only decision that can be made is to stop since these states are end states in the state space. Thus, the decision rule f_1 (for time t=1) is

$f_1 = \{(1,Op02), (2,\text{stop}), (3,Op01), (4,Op02), (5,Op03), (6,\text{stop}), (7,Op03), (8,\text{stop}), (9,Op01), (10,\text{stop}), (11,\text{stop}), (12,\text{stop}), (13,\text{stop}), (14,\text{stop}), (15,\text{stop})\}$.

In a similar way, the values and decision rules for all time points can be calculated. The details can be found in Tables 2 and 3. These tables show that the decision rules become stationary for sufficiently large t (i.e., $t \geq 3$ for this case), since the Markov chain does not contain any cycles other than the absorbing end states. This results in the following global decision strategy:

$f = \{(1,Op03), (2,\text{stop}), (3,Op01), (4,Op02), (5,Op03), (6,\text{stop}), (7,Op03), (8,\text{stop}), (9,Op01), (10,\text{stop}), (11,\text{stop}), (12,\text{stop}), (13,\text{stop}), (14,\text{stop}), (15,\text{stop})\}$.

Note that the best decision for the first state (describing the initial situation) has changed from *Op*02 at t=1 to *Op*03 at $t \geq 2$. Thus, by looking at the complete state space and decision space, the algorithm calculates a global strategy that gives the best decision for each state in the MDP.

3.3. Computability of the decision strategy

In the previous section we have elaborated on the computation of an optimal decision strategy for an example with a small state space. Such a computation is done in two steps: (i) building the state space, and (ii) computing the decision strategy. In this section, we elaborate on our experiences with the computability for larger, realistic PDMs, since both steps make a substantial contribution to the overall time complexity of this approach.

Table 2
The minimal cost values obtained by the value iteration algorithm, i.e. $V_t(i)$.

t	i														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	5.0	0.0	5.0	5.0	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
2	9.25	0.0	9.25	9.75	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
3	9.49	0.0	9.25	9.75	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
4	9.49	0.0	9.25	9.75	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
5	9.49	0.0	9.25	9.75	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
6	9.49	0.0	9.25	9.75	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
7	9.49	0.0	9.25	9.75	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
8	9.49	0.0	9.25	9.75	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
9	9.49	0.0	9.25	9.75	9.0	0.0	9.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0

Note that the values become stable, i.e. they do not change anymore, for $t \geq 3$.

Table 3
The rows in this table denote the decision rule (f_t) for each point in time (t).

t	i														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	**Op02**	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
2	**Op03**	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
3	*Op*03	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
4	*Op*03	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
5	*Op*03	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
6	*Op*03	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
7	*Op*03	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
8	*Op*03	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
9	*Op*03	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–
10	*Op*03	–	*Op*01	*Op*02	*Op*03	–	*Op*03	–	*Op*01	–	–	–	–	–	–

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

As mentioned earlier, the state space of our MDP problem is finite. Nevertheless, the state space can become extremely large. For instance, if we use an initial situation in the mortgage example in which none of the operations have been executed yet and no value is available for any of the data elements, i.e., the initial state with ({}, {}, {}) of Fig. 3(a), the state space contains 2218 states and, as such, is much larger than in our example of Fig. 4 with 15 states. This is because the six leaf operations can be executed in an arbitrary order. All possible combinations are then explicitly represented in the state space.

Generally, an upper bound for the size of the state space can be given in terms of the number of operations of the PDM. Each operation can be either (i) successfully executed, (ii) unsuccessfully executed (failed), or (iii) not yet executed. Thus, there are $3^{|O|}$ possible combinations of operations, which is an upper bound for the size of the state space. However, most state spaces will be smaller since not all combinations of executed operations are possible. For instance, the state space for the mortgage example will never contain the state ({*Op*02,*Op*07}{*Op* 09}{*C*,*G*}), since *Op*02 can never be executed before both *Op*08 and *Op*10 are executed.

We have experimented calculating the state space for one of the industry cases in which PBWD was applied. The PDM of the UWV-case (described in [36]) contains 45 data elements and 50 operations, which is significantly larger than our running example but still reasonably small for a practical case. During our experiments we experienced that a normal PC[5] was not able to process the complete state space of this case. Therefore, we tried to limit the amount of states in the state space based on a number of restrictions described in [46]. One of the main design objectives for the redesign project of the case was a minimum number of contacts with the client (see [36]). This requirement was realized in the design by requesting all input information from the client at once at the start of the case. Therefore, the restriction of using an initial state in which all leaf operations have been (successfully) executed is quite realistic. Calculating the first 300,000 states of the state space with only this restriction took more than 60 days, and we failed to gain the complete state space. Furthermore, it also seems realistic to assume that all operations are executed successfully, since the failure probabilities for the operations in the PDM are typically very low. With these two restrictions, the state space for the social benefits case was calculated in 2.5 days. The state space contained 66,581 states. Based on this partial state space, a decision strategy can be computed. However, this decision strategy may not be optimal because of the assumptions that were made to limit the state space.

For the second step in the computation of the optimal decision strategy, we have used a standard value iteration algorithm for the example in Section 3.2. Other algorithms such as policy iteration and dynamic programming [43] may be used as well and may decrease the computation time needed to find an optimal strategy given the state space, the time space, the decision space, the transition function, and the cost functions of the MDP. However, improving the time complexity of this step by using a faster algorithm will not have a huge impact on the total computation time, since building the state space (step 1) may still be the bottleneck.

The experiments, described in this section, show that it is theoretically possible to find globally optimal recommendations for executing the PDM, but for real-life applications it may be intractable to compute the optimal solution. Typically, PDMs may be even larger than the one for the UWV-case, e.g., containing hundreds of data elements. Therefore, the MDP may become unsolvable in practice. Note that the selection of the next operation needs to be done at real time and repeated after every step. Hence, a more pragmatic way of determining recommendations would be desirable. As a second solution approach, we now describe a number of *local decision strategies* that focus on the selection of the candidate operation for execution within the set of executable operations.

4. Local decision strategies

The issue of unsolvable MDPs due to large state spaces also exists in other application areas of MDP theory. In the field of production planning, simpler heuristics have been developed that perform well and approximate the optimal solution of the decision problem. These heuristics aim for a local instead of a global optimization. They do not consider the effects of a decision on the next steps in the execution, but merely look at the currently available set of possible decisions and take the best one out of it. For instance, in the initial situation of Fig. 2(b), two operations are executable. When deciding on the next operation to be executed, only those two operations are considered without looking at the further, possible effects of the choice for one of the two. Based on the characteristics of the two operations, the best performing one with respect to the selected criterion is selected.

Based on production planning heuristics, we have developed some *local decision strategies* for the direct execution of a PDM. We have drawn inspiration from *sequencing and scheduling rules* in the field of logistics and production planning [28,42]. Short-range production planning is a method to decide, a priori, which resource is going to perform which jobs in which order for production systems with a shop structure (i.e., flow shop, job shop, or open shop) [42]. Production planning problems are usually too large to be solved mathematically, and researchers therefore have developed pragmatic ways (e.g., heuristics) that approximate the desired solution to a scheduling problem. Many different strategies or rules exist to schedule jobs that have to be performed by a machine [28,42]. Well-known strategies are, for instance, First Come First Served (FCFS) or Earliest Due Date (EDD) [42]. The following decision strategies can be applied to our selection problem of finding the next operation in the execution of a PDM:

- *Random*: The operation is randomly selected from the set of executable operations (cf. [28]).

[5] A normal PC at the time of writing this paper is e.g. an Intel Pentium 4 with a 3.8 GHz processor and 3 GB of RAM.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

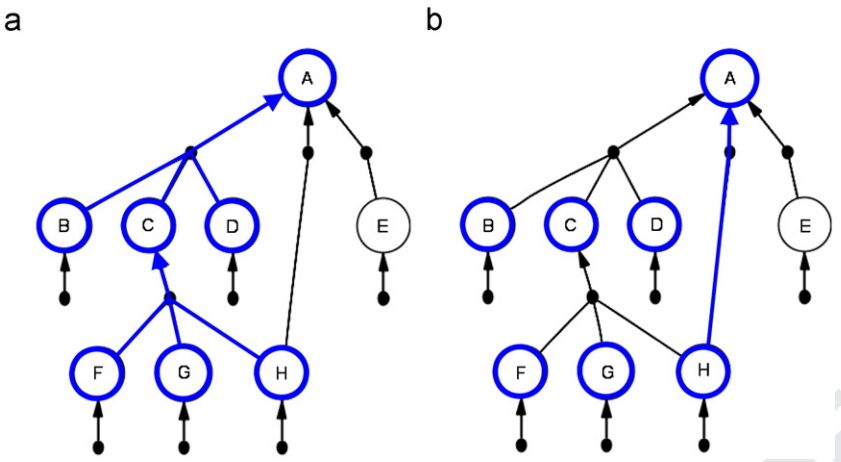


Fig. 6. Two different execution paths leading to the end product of the PDM. The cost of executing an operation is presented in Table 1. (a) Total cost: 5.0+5.0=10.0. (b) Total cost: 9.0.

- *Lowest cost*: The operation with the lowest cost is selected.
- *Shortest processing time*: The operation with the shortest duration is chosen (cf. SPT defined in [28]).
- *Lowest failure probability*: The operation with the lowest probability of not being performed successfully is chosen.
- *Shortest distance to root element*: The operation with the shortest distance to the root element (measured in the total number of operations) is selected. The distance of an operation to the root element is the "shortest path" from this operation to an operation that produces the root element. The distance to the root element can be measured as the total number of operations to the root element (cf. FOPNR defined in [28]).
- *Shortest remaining processing time*: The operation with the shortest remaining processing time is chosen. The shortest remaining processing time is another form of the distance to the root element. In this case, the processing times of the operations on the path to the root element are added up (cf. SR defined in [28]).

If, for example, a lowest cost strategy was used for decision making on the initial situation of the mortgage example in Fig. 2(b), then *Op*02 would have been selected, since the cost for *Op*02 is lower than that for *Op*03 (i.e., 5.0 vs. 9.0). If a shortest processing time strategy was used, *Op*03 would have been selected, and with a shortest distance to root element strategy *Op*03 would also have been chosen.

The above strategies present ways to select the next step based on a single strategy, i.e., only one performance goal is considered. A more advanced way to use these simple selection strategies is by combining them. With a single strategy there may be several candidates available for selection with the same value for the performance goal. To distinguish between the operations that all have an optimal value based on the first strategy, one can use another strategy to complement the first one. For instance, if the lowest cost strategy leads to three operations with the same minimal execution cost, another strategy (e.g., lowest failure probability) may be used to rank these three operations and find the best one among the three. Additionally, weighted criteria can be used. For instance, if the processing time and the execution cost are equally important for the selection of an operation, the selection strategy can be based on an equal weight (0.5 vs. 0.5) for both performance criteria.

Using the strategies presented above, the selection of the next candidate for execution is optimized locally (i.e., within the set of currently executable operations); the effect of the selected operation on future steps is not taken into account.[6] Such a local optimization may lead to a less desirable situation if we consider the overall performance of the case. Consider, for instance, the execution steps for the mortgage example with respect to the total execution cost. The first execution sequence contains *Op*02 followed by *Op*01. The total cost of this execution is: 5.0+5.0=10.0. The total cost of the alternative execution path containing only *Op*03 is 9.0. Thus, the cost of the second execution path is lower. Selecting the best candidate based on the lowest cost selection strategy in this case does not lead to the best overall decision for the case considering the total cost of execution (see Fig. 6). However, many of these pragmatic heuristics are proven to reach a near-optimal solution [28,34]. To assess our approaches, a comparison between some local strategies and the global decision strategy is made in the next section.

5. Evaluation

In this section, we evaluate the two proposed approaches based on the mortgage example. In order to assess the performance of the global and local decision strategies, we compare their results using two performance criteria. We consider a situation in which the goal

[6] Note that this is comparable to a situation in the MDP in which there is only one decision left to the end of the process, i.e. $t=1$.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

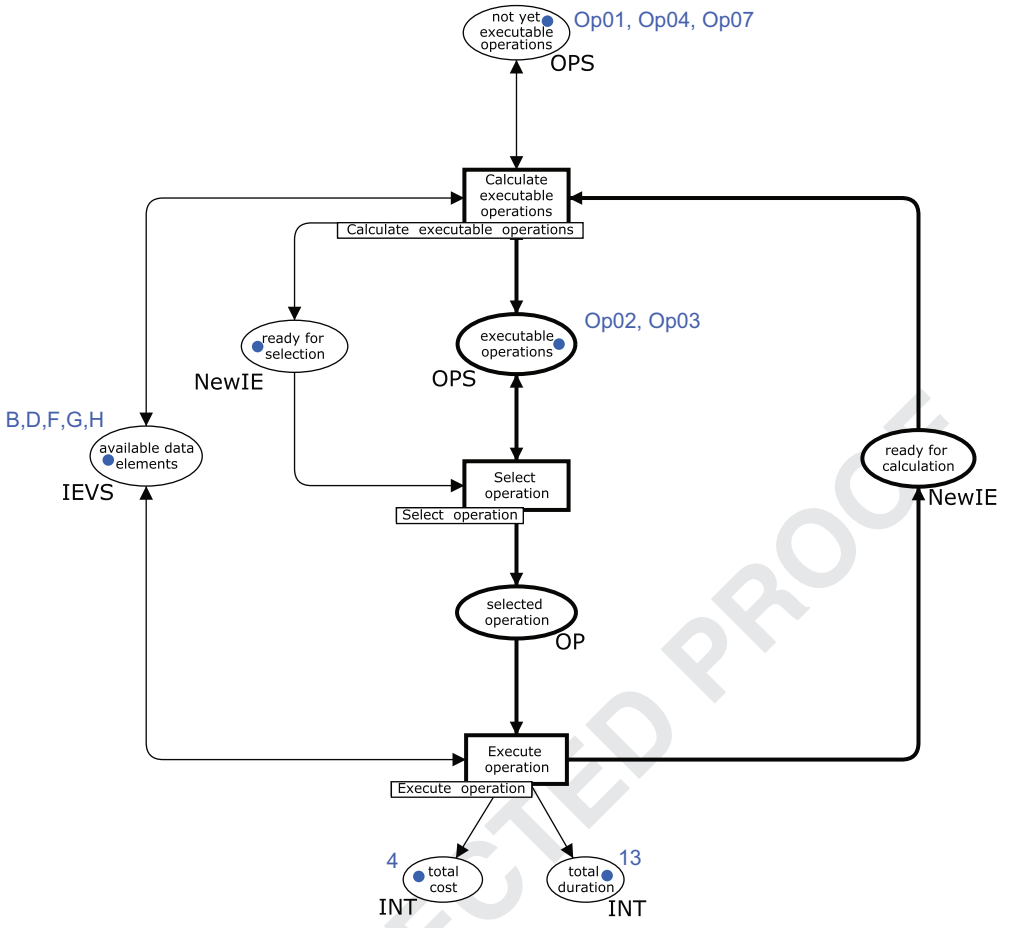


Fig. 7. The main level of the simulation model, represented by a Colored Petri Net (CPN). The main execution stream of the simulation model is indicated by thick lines. Also, the initial situation of Fig. 2(b) is depicted by tokens in the model.

is to minimize cost as well as a situation in which the processing time is minimized.

For the global decision strategy, an expected value of each of these performance criteria can be calculated based on the MDP (see also Definition 3.2). Since the mortgage example is small enough, we are able to compute a complete state space[7] and decision strategy for the MDP derived from the PDM. To do so, we used the prototype workflow system presented in Section 6.

The performance of the local decision strategies is measured through simulation. In this simulation study we have executed several simulation runs per local decision strategy. A simulation run corresponds to one complete execution of the PDM of the mortgage example. For each execution the total execution cost and total processing time were determined. Per strategy, we collected 300 samples (i.e., $n=300$) and constructed 90%-confidence intervals [25]. In the next section, the simulation model is explained, and then the simulation results are compared to the results of the global selection strategies.

[7] We take the initial situation in which no input data element values are available, i.e., ({}, {}, {}).

5.1. Simulation model

The simulation model for executing a PDM describes the functional design of a tool supporting the direct execution of a PDM based on a local decision strategy. The simulation model is represented by a Colored Petri Net (CPN, [16]) and has been developed using CPN Tools [10]. The model takes care of a *correct* step-by-step execution of the PDM as illustrated in Fig. 2. This means that the functional design ensures that only executable operations are executed, that no duplicate data element values are produced, and that only one operation at a time is performed. It does not deal with performance and the selection of the best operation for the next step. The structure of the model is generic since the model does not need to be changed for a different PDM or a different selection strategy. The PDM is added to the model as a variable and the selection strategy as a function. This section describes the CPN model on a high level.

The main level of the CPN model is depicted in Fig. 7. It contains eight places and three transitions. All three transitions are sub-processes in which the actions to be taken are described in detail. The main execution stream in the model is indicated by thick lines: First, the

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

executable operations are determined, and one of the executable operations is selected based on the local decision strategy that was chosen. The selected operation is then executed. If the execution of the operation is successful, the data element and its value are stored in the place *available data elements*, and the execution cost and execution time for this operation are added to the total cost and duration of the process. If the execution is not successful, only cost and duration are increased; no data element is added to the set of *available data elements*. Next, the executable operations can be executed again; this procedure is then repeated until the end product is produced or no executable operations remain.

5.2. Comparison of decision strategies

The results of the simulation study are shown in Fig. 8. The first diagram shows the results for the cost performance criterion. Confidence intervals of the total cost resulting from executions under a lowest cost and a random decision strategy are given. According to the simulation results, the lowest cost strategy has a mean of 8.11. The 90%-confidence interval has a lower bound of 7.55 and an upper bound of 8.67. The total cost under a *random* strategy has a mean of 10.71 (between 10.21 and 11.22 according to the 90%-confidence interval). Since these confidence intervals do not overlap, we may conclude that a lowest cost strategy leads to a lower total cost than does a random selection strategy. Apart from the comparison of the local strategies, we can also compare the outcome of the local strategies to the optimal solution resulting from the MDP as a benchmark. The total expected cost for executing the PDM of the mortgage example is 7.54 (see horizonal line in Fig. 8(a)). Fig. 8 shows that a lowest cost strategy closely approaches the optimal solution of the MDP.

The second diagram depicts the results for the processing time performance criterion. The confidence intervals for the total processing time under a shortest processing time strategy and under a random strategy are given. The total processing time under a shortest processing time strategy has a mean of 9.75, a lower limit of 9.13 and an upper limit of 10.37. The *random* strategy has a mean of 13.94, a lower limit of 13.48 and an upper limit of 14.40. Clearly, the shortest processing time strategy performs better than does a random strategy to minimize the total processing time. The results of the local decision

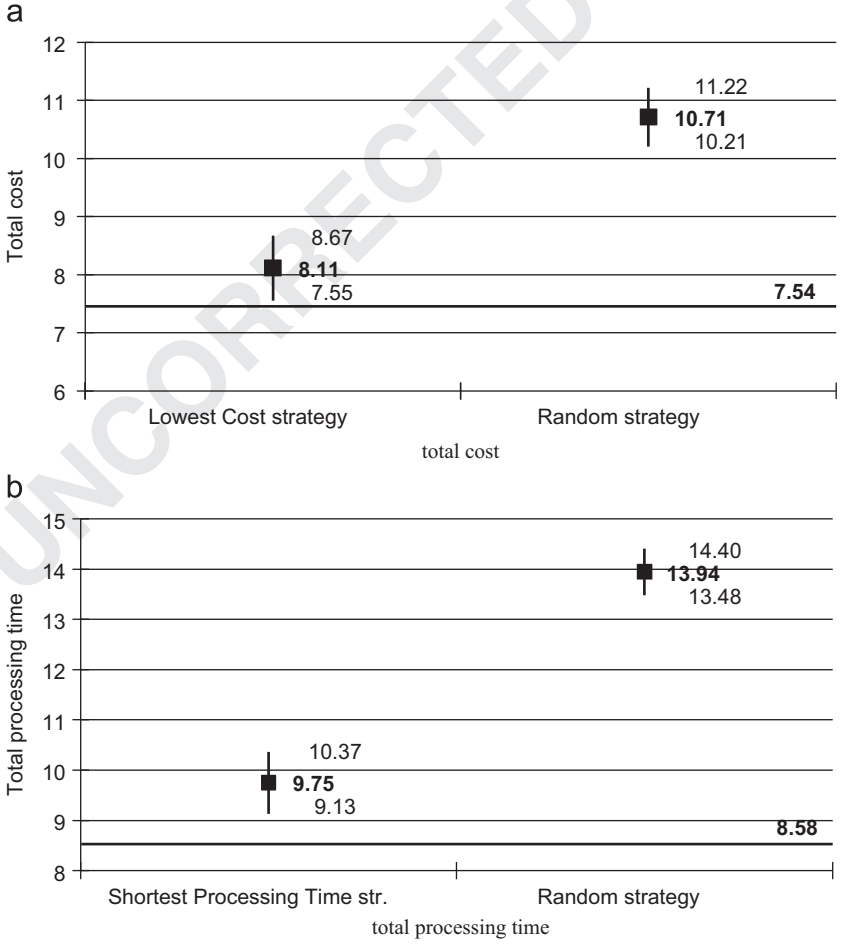


Fig. 8. Some results of the simulation study. (a) Total cost. (b) Total processing time.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

strategies can again be compared to the expected processing time achieved by the global decision strategy obtained by the MDP: 8.58 (see horizonal line in Fig. 8(b)). The shortest processing time strategy approaches the optimal solution compared to a minimal processing time decision strategy in the MDP.

We conclude from this simulation study that (i) using a heuristic (e.g., a lowest cost strategy to minimize cost) leads to better results than does random selection of operations, and (ii) the outcomes of the heuristics closely approximate the optimal solution calculated by an MDP.

6. System support

This section describes the workflow system that we have developed for Product-Based Workflow Support. It allows for the direct execution of a PDM by guiding the user through all steps to the end product. The system presents *execution recommendations* to the user based on the selected decision strategy and the current state of a case. Section 6.1 explains the architecture of our workflow system. Next follows an explanation of how the global and local strategies are supported by this system.

6.1. Architecture

Our approach builds on ProM and DECLARE.[8] ProM [2,12] was initiated as a framework for *process mining*, but in recent years it has evolved into a broad and powerful *process analysis* tool supporting all kinds of analyses related to business processes [12]. ProM has a plug-able architecture. ProM 5.2 has 286 plug-ins, each realizing a particular type of functionality, e.g., the α miner plug-in is able to discover a process model from event logs, and the conformance checker plug-in is able to measure the quality of a model in relation to an event log. DECLARE is a workflow management system [29–31]. DECLARE is based on a *declarative* approach to business process modeling and execution, and therefore provides more flexibility than do conventional workflow management systems. While the mainstream workflow management systems use a procedural approach where every activity needs to be triggered, DECLARE starts from the viewpoint that "anything is allowed unless explicitly forbidden."

To realize a workflow system for Product-Based Workflow Support, we use DECLARE as-is, and added several plug-ins to ProM. DECLARE is used to offer work to end users and could be replaced by another workflow system. However, we selected DECLARE because it is easy to specify workflow models that do not impose any control-flow constraints, i.e., from the viewpoint of the process, any sequence of activities is possible (unless explicitly forbidden). Another reason for selecting DECLARE is the connection to ProM which allows combining ProM's analysis and reasoning capabilities with the flexibility of DECLARE. DECLARE does not know about data and dependencies between data elements, i.e., the PDM is only known to ProM. ProM uses a so-called *recommendation service*, the PDM, and knowledge about the history of cases to inform DECLARE about recommended operations. Such a recommendation service (that is driven by a PDM and the availability of data) replaces the conventional control-flow oriented workflow engine. Two recommendation services have been realized using ProM's plug-able architecture: one using the global MDP approach and one using local decision strategies.

An overview of the architecture of our system is shown in Fig. 9. As a first step in the use of the PDM-based workflow system, a PDM is loaded into the ProM framework and exported to a DECLARE model. This DECLARE model solely contains the operations of the PDM as activities and their input and output data elements. Thus, no control flow is added to the DECLARE model, i.e., the DECLARE activities are not related to each other in any way. Next, the ProM PDM recommendation service is started and a strategy is selected by the user. Actual cases can be handled after the DECLARE model has been loaded into the DECLARE framework, and the DECLARE worklist has been started. The DECLARE framework communicates with the ProM recommendation service by sending *queries*. Such a query contains the current state of the case in terms of executed activities and available data element values. Based on this information, the PDM recommendation service calculates which operation is the best candidate to be executed for the case with respect to the selected decision strategy. The result of this calculation is sent back to the DECLARE framework as a *recommendation*. Finally, this recommendation is shown to the user in the worklist of DECLARE. The user may then follow up on the recommendation and execute the recommended operation. When the operation is executed, a new query is sent to the ProM recommendation service.

Two different implementations of the PDM recommendation services have been developed. The first one is based on the global decision strategies introduced in Section 3. The second implementation uses the local decision strategies, as described in Section 4, to generate recommendations for a query. Both implementations are described in greater detail in the next sections.

6.2. Global decision strategies

In case a global decision strategy based on an MDP is chosen for generating recommendations, the user first has to calculate the state space and the optimal decision strategy in ProM (see Figs. 10 and 11). Next, the recommendation service in ProM can be started. Also, the DECLARE framework and the DECLARE worklist are started, and the user fills out the values for the leaf data elements in DECLARE. Suppose we take the same initial situation for the mortgage example as before (i.e., values for *B*, *D*, *F*, *G*, and *H* are available and *E* is not available). A query containing the set of executable operations (i.e., *Op*02, *Op*03) and a set of available data elements and their values is then sent to ProM. ProM determines a recommendation based on the calculated global decision strategy (i.e., *Op*03 under a minimize cost strategy) and sends it to DECLARE. In the DECLARE worklist, the recommendations are

[8] ProM can be downloaded from www.processmining.org. DECLARE can be downloaded from http://www.win.tue.nl/declare/.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

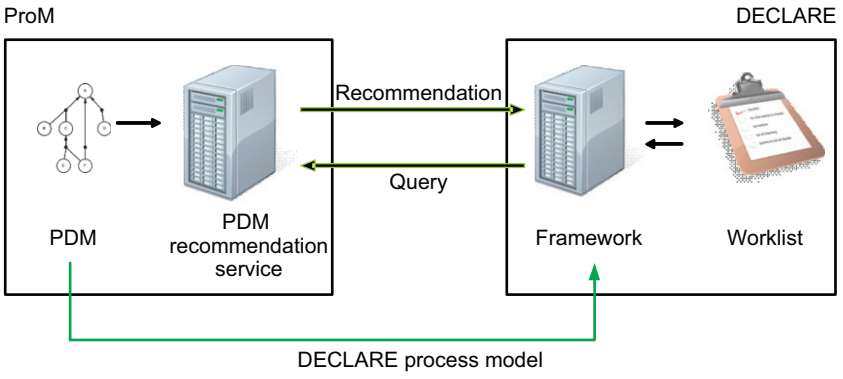


Fig. 9. An overview of the architecture of the workflow system based on ProM and DECLARE.

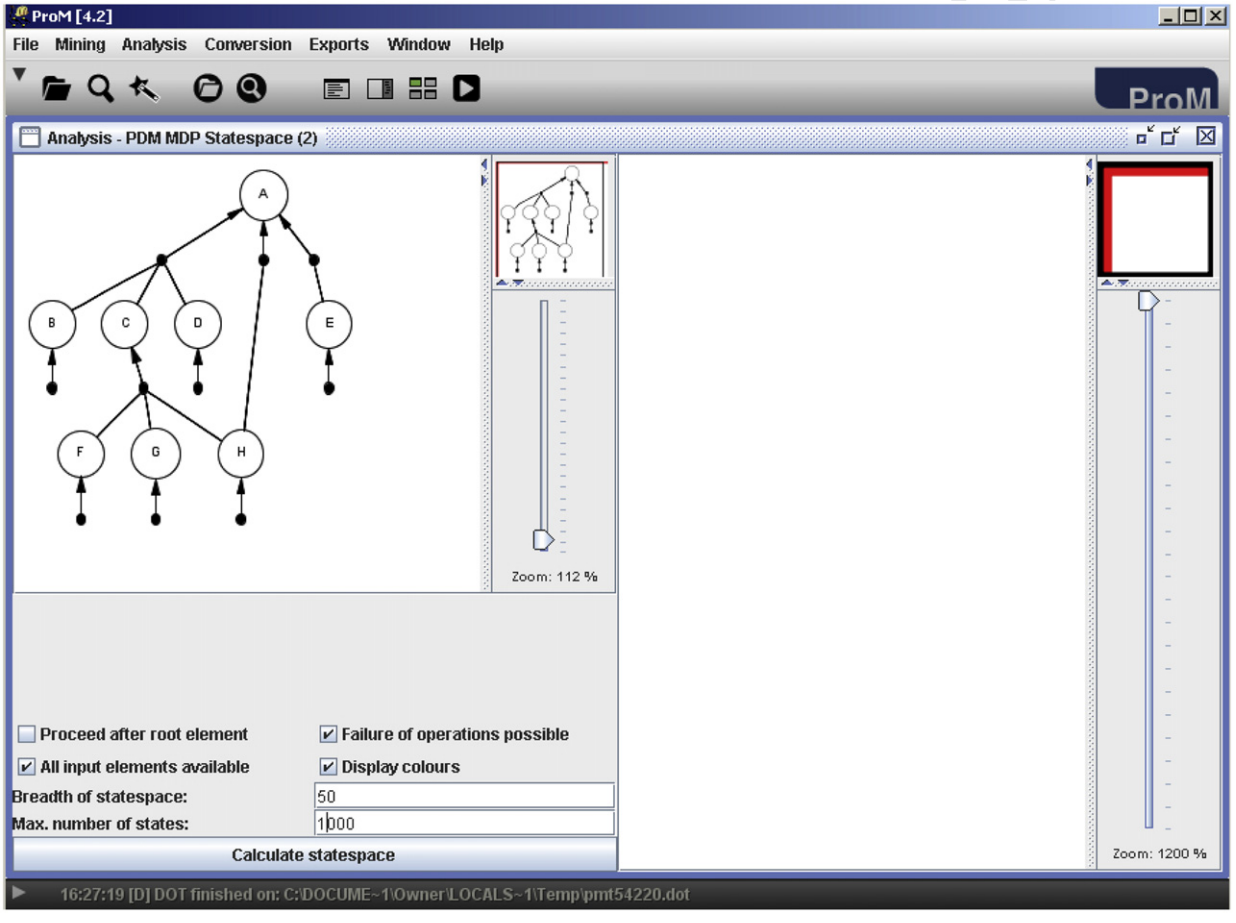


Fig. 10. This screenshot of ProM shows the user interface of the PDM recommendation service before computing the state space for the MDP. A number of options may be selected to restrict the state space as presented in Section 3.3. For example when the checkbox "all input elements available" is checked it reflects the situation in which all leaf data elements are available at the start of the process, or when "failure of operations possible" is checked the possibility of unsuccessful execution of operations (failure) is taken into account.

presented (see Fig. 12). The first one is the recommended activity according to the calculated strategy. The others are second best options. Each recommendation also has a field called "rationale" in which the total expected cost or processing time to finish the process is shown.

6.3. Local decision strategies

In the PDM recommendation service based on local decision strategies, the user can immediately select one of the strategies introduced in Section 4 via a drop-down

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

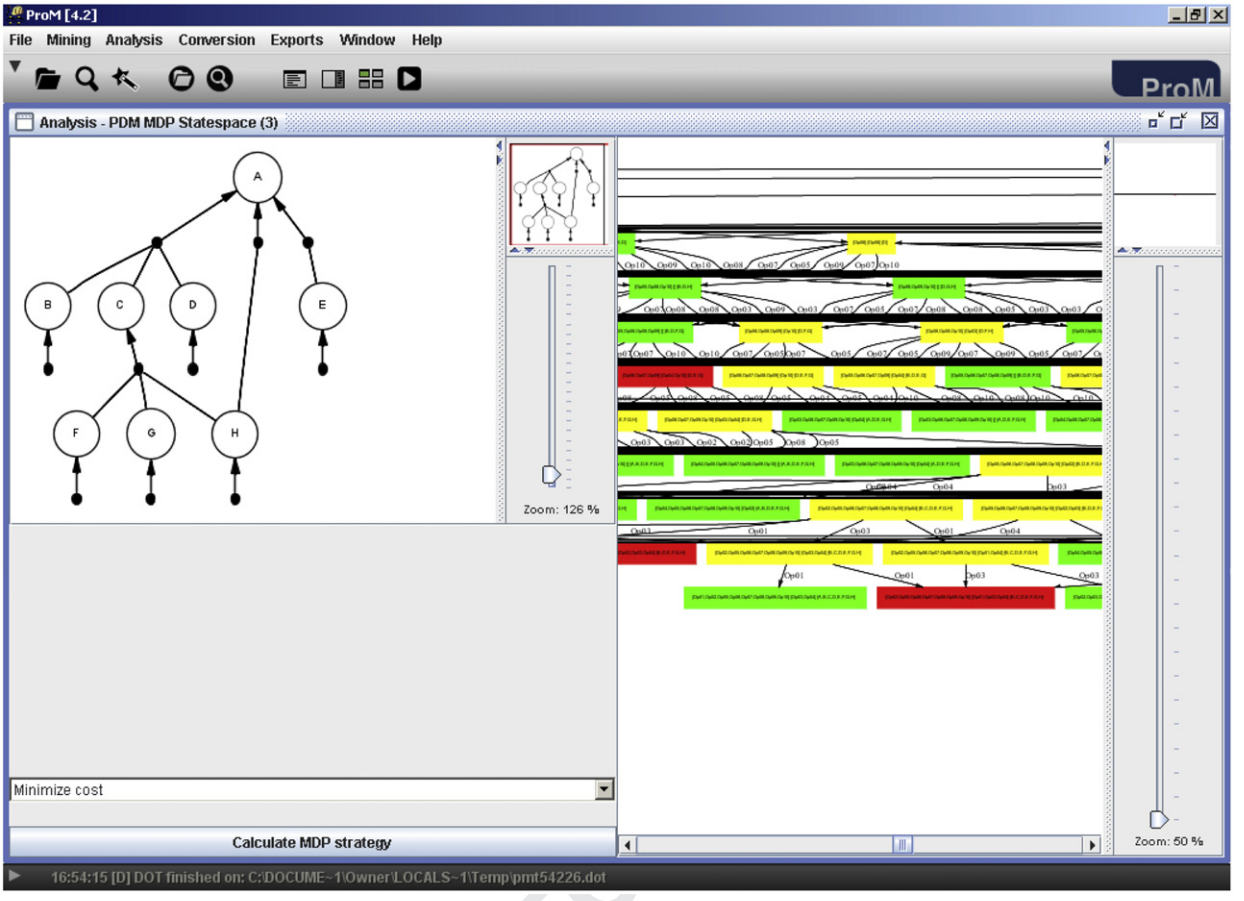


Fig. 11. On the right hand side, this screenshot of ProM shows the complete state space that was computed for the mortgage example. In the lower left corner the user can select the performance objective to be optimized (e.g. minimize cost, or minimize processing time) via a drop-down menu. Next, the decision strategy is calculated.

menu. Once the strategy has been selected, the recommendation service can be started. The execution of the process based on these recommendations is similar to the execution based on the global strategies described above. Given the state of the case, the recommendation service determines which operations are executable and which one of these is the best choice considering the selected local decision strategy. A list of all executable operations is presented with the best next step listed first. The user may decide whether or not to follow the recommendation and execute a next step. In any case, the recommendation service will again calculate a recommendation for the new state of the process until the end product is produced.

Note that it is possible to change a local decision strategy during the execution of a case. This is useful; for example, if a company decides to switch from a low cost strategy to a strategy focusing on a short throughput time to be able to deal with extra case arrivals as quickly as possible. In contrast to this, changing a global strategy during the execution of a case is not allowed, since this would force the system to recalculate the complete strategy. Recalculation may take some time and therefore is not acceptable at run-time.

7. Related work

Most of the workflow approaches described in the literature and supported by commercial systems and academic prototypes assume a procedural approach [3]. Although there are huge differences in the expressiveness and the suitability of procedural languages ranging from YAWL and BPMN to BPEL and ADEPT, the starting point of all these languages is the modeling of control-flow dependencies among tasks. Few alternative approaches have been proposed. For example, the DECLARE system supports a more declarative style of modeling grounded in LTL [29–31]. However, also DECLARE focuses on control-flow dependencies among tasks. This paper proposes starting from data and dependencies between data elements. The PDM has a similar role as the Bill-of-Materials (BOM) in production planning. This relation has been explored in the nineties [1,32]. However, no concrete workflow support has been provided thus far. In our earlier work [1,36,37,46], we showed that in some cases the PDM could be converted into a procedural model. However, as indicated in this paper, this has the drawback that logical dependencies and performance related concerns get mixed, and it becomes impossible to dynamically change the strategy.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

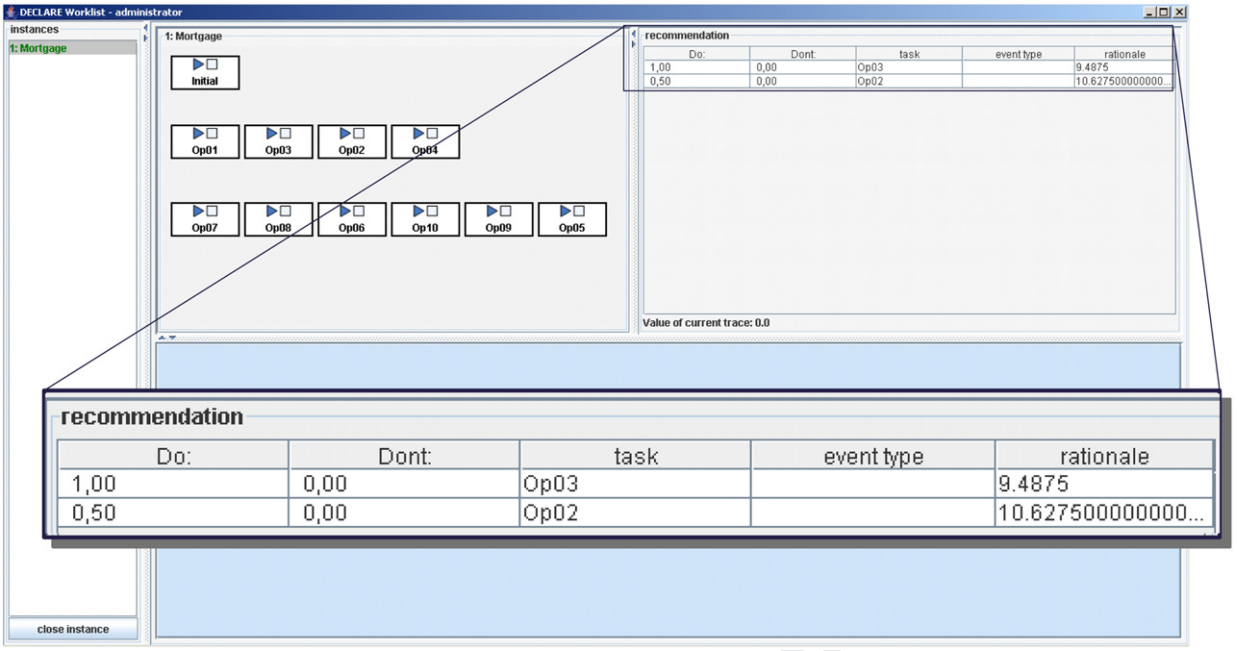


Fig. 12. Two recommended activities are shown for the process which is in the state where all leaf operations have been executed and the values for data elements *B*, *D*, *F*, *G*, and *H* are available. Based on the global decision strategy calculated by using the MDP, *Op*03 is recommended as the next operation to be executed (cf. $V_n(3)$ and $f_n(3)$, for $n \geq 2$, in Tables 2 and 3). Note that all executable operations are ranked with respect to the performance objective: the second best next step is *Op*02. Other operations are not executable.

Also related is the work on case handling, as it is supported by systems such as BPM|one [5]. In the case handling paradigm the availability of data also plays a dominant role. However, hard-coded control-flow dependencies need to be added to steer the user. The case handling paradigm is one of many approaches aimed at providing flexibility in workflows. Clearly, the direct execution of a PDM also offers flexibility. However, most of the literature focusing on workflow flexibility assumes a procedural language and changes of the underlying model. See for example, [39,47] for related work and typical flexibility mechanisms.

Recently, there has been a revival of approaches that balance data and process aspects, cf. the artifact-centric approach [7] and the proclets approach [4]. However, these approaches do not use the data dependencies as a way to replace control-flow.

Most related to our work is the work by Kress, Melcher and Seese on Executable Product Models [19,20]. This approach is based on our definition of a PDM [1,36,37] and the authors also propose to directly execute the PDM. Kress et al. combine this with multi-agent systems, reinforcement learning, and genetic algorithms. The core idea is to learn how to best execute the PDM to reach an optimal resource allocation with respect to a certain goal (e.g., the earliest completion time of a task). In contrast to this, our approach looks at the overall optimization of a case with respect to the total throughput time or processing cost. Also, Kress et al. assume that all executable operations in the PDM will be executed eventually (possibly leading to extra, unnecessary work), while our approach only selects those operations to be executed that really contribute to the end product.

The workflow solution presented in this paper combines DECLARE [30] and ProM [2,12]. The core idea is that a recommendation service in ProM provides guidance to users of the DECLARE system. Such a recommendation-based approach was proposed in [40], and our current system is using this idea. However, the recommendation approach presented in [40] is not using a PDM. Instead a model is learned and it is assumed that all logical dependencies and other constraints are handled outside of the recommendation service. Hence, this approach is not data driven and serves as an add-on rather than as a full-fledged workflow engine. The recommendation approach presented in [40] is closely linked to *recommender systems* [8,9,38]. Recommender systems are widely used in other domains such as information filtering. They are used to recommend items (films, television, video on demand, music, books, news, images, web pages, products, etc.) that are likely to be of interest to the user. These systems typically do not consider a process, i.e., the creation of a product using a sequence of operations. The broader area of Web mining, i.e., the application of data mining techniques to discover patterns by observing users of the Web [15], has the same characteristic. Therefore, these techniques cannot be applied to workflow management. Moreover, these approaches aim at analysis rather than execution support involving data and processes.

8. Conclusion

This paper has presented a new approach to workflow execution on the basis of a product description, named the Product Data Model. Instead of aiming at the derivation of

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

a workflow design that is in general the best possible way to generate an informational product, the Product Data Model *itself* is proposed as the vehicle to steer a workflow's execution. Based on data element values readily available for a specific case, on the one hand, and a selected strategy, on the other hand, this approach recommends the next step that should be performed for the case. There is a clear separation of concerns at work here: the product data model is based on functional requirements, while the selected strategy focuses on performance (e.g., minimizing costs or time). Therefore, the execution of one case can still be dynamic and flexible, i.e., the strategy can be changed during execution and will take the actual data values into consideration.

We have introduced two types of strategies to calculate the recommendations: (i) a global decision strategy, which takes into account the effect of the current decision that is made on future decisions, and (ii) a number of local decision strategies, which only look at the set of directly available, executable steps. The first approach is computationally demanding and may be infeasible for large decision problems, but generates the overall best decision for a case. The second approach is much faster and more flexible, but may result in sub-optimal solutions. We have assessed the performance of several strategies by a simulation study, showing that the local decision strategies indeed reach a near-optimal solution to the global decision strategy for a realistic case. Finally, the feasibility of the presented ideas is demonstrated by the description of a fully operational prototype that supports the approach presented in this paper.

A limitation to this work is that the different decision strategies presented in this paper all focus on improving performance while considering a case in isolation. The best decisions are defined in terms of the performance objective on the case level, e.g., the minimization of throughput time or cost for a single case. Further research may consider how optimization can take place on the process level with respect to e.g., the utilization of the process, or the optimal distribution of work among resources.

Also, the presented decision strategies are not yet able to deal with parallel execution of operations. Concurrency would lead to difficulties with different values being available for the same data element. In general, data updates are an issue for the use of PDMs and other data-driven approaches. Noteworthy is that current data-driven workflow management systems, e.g., the case-handling system FLOWer, prevent parallel executions for the same case by blocking the whole case as soon as someone is working on it. However, we can imagine less restrictive mechanisms that will still allow some level of parallelism to be incorporated.

Currently, we plan to cooperate with our industrial partners to incorporate the direct execution of a PDM and the decision strategies in a commercial tool. The availability of such a tool will certainly enhance the practical applicability of the ideas, which will be useful to further evaluate the approach in practice.

Acknowledgements

We would like to thank Maja Pesic for her work on the development of DECLARE. We would also thank the many people who have contributed to ProM. Their work enabled us to realize the workflow solution presented in this paper. We thank Jan van der Wal for sharing his expertise on Markov Decision Processes with us.

References

[1] W.M.P. van der Aalst, On the automatic generation of workflow processes based on product structures, Computers in Industry 39 (1999) 97–111.

[2] W.M.P. van der Aalst, B.F. van Dongen, C.W. Günther, R.S. Mans, A.K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, A.J.M.M. Weijters, ProM 4.0: comprehensive support for real process analysis, in: J. Kleijn, A. Yakovlev (Eds.), Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007), Lecture Notes in Computer Science, vol. 4546, Springer-Verlag, Berlin, 2007, pp. 484–494.

[3] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros, Workflow patterns, Distributed and Parallel Databases 14 (1) (2003) 5–51.

[4] W.M.P. van der Aalst, R.S. Mans, N.C. Russell, Workflow support using proclets: divide, interact, and conquer, IEEE Bulletin of the Technical Committee on Data Engineering 32 (3) (2009) 16–22.

[5] W.M.P. van der Aalst, M. Weske, D. Grünbauer, Case handling: a new paradigm for business process support, Data and Knowledge Engineering 53 (2) (2005) 129–162.

[6] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, J. Su, Towards formal analysis of artifact-centric business process models, in: Lecture Notes in Computer Science, vol. 4714, 2007, p. 288.

[7] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, J. Su, Towards formal analysis of artifact-centric business process models, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), International Conference on Business Process Management (BPM 2007), Lecture Notes in Computer Science, vol. 4714, Springer-Verlag, Berlin, 2007, pp. 288–304.

[8] R. Burke, The Wasabi personal shopper: a case-based recommender system, in: Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence, American Association for Artificial Intelligence, 1999, pp. 844–849.

[9] R. Burke, A case-based reasoning approach to collaborative filtering, in: E. Blanzieri, L. Portinale (Eds.), Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning (EWCBR 2000), Lecture Notes in Computer Science, vol. 1898, Springer-Verlag, Berlin, 2000, pp. 370–379.

[10] CPN Tools Home Page ⟨http://wiki.daimi.au.dk/cpntools/cpntools.wiki⟩.

[11] T.H. Davenport, Process Innovation: Reengineering Work through Information Technology, Harvard Business School Press, Boston, 1993.

[12] B.F. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, W.M.P. van der Aalst, The ProM framework: a new era in process mining tool support, in: G. Ciardo, P. Darondeau (Eds.), Application and Theory of Petri Nets 2005, Lecture Notes in Computer Science, vol. 3536, Springer-Verlag, Berlin, 2005, pp. 444–454.

[13] M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede, Process-Aware Information Systems: Bridging People and Software through Process Technology, Wiley & Sons, 2005.

[14] M. Hammer, J. Champy, Reengineering the Corporation, Nicolas Brealey Publishing, London, 1993.

[15] P. Hofgesang, Online mining of web usage data: an overview, in: Web Mining Applications in E-commerce and E-services, Studies in Computational Intelligence, vol. 172, Springer-Verlag, Berlin, 2009, pp. 1–24.

[16] K. Jensen, in: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts, Monographs in Theoretical Computer Science, Springer-Verlag, Berlin, 1997.

[17] H.J. Johansson, P. McHugh, A.J. Pendlebury, W.A. Wheeler, Business Process Reengineering: Breakpoint Strategies for Market Dominance, Wiley & Sons, 1993.

[18] M.T. Jones, Artificial Intelligence: A Systems Approach, Jones and Bartlett Learning, 2008.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008

[19] M. Kress, J. Melcher, D. Seese, Introducing executable product models for the service industry, in: Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS '07), IEEE Computer Society, 2007, pp. 1–10.
[20] M. Kress, D. Seese, Executable product models—the intelligent way, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (ISIC '07), 2007, pp. 1987–1992.
[21] G.F. Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Pearson Education, 2005.
[22] M. McDonald, M. Blosch, T. Jaffarian, L. Mok, S. Stevens, Growing IT's Contribution: the 2006 CIO Agenda, Gartner January, 2006.
[23] M. McDonald, T. Nunno, Creating Enterprise Leverage: the 2007 CIO Agenda, Gartner January, 2007.
[24] M. McDonald, T. Nunno, D. Aron, Making the Difference: the 2008 CIO Agenda, Gartner January, 2008.
[25] D.C. Montgomery, G.C. Runger, Applied Statistics and Probability for Engineers, John Wiley and Sons, New York, 1999.
[26] B. Mutschler, M. Reichert, J. Bumiller, Unleashing the effectiveness of process-oriented information systems: problem analysis, critical success factors and implications, IEEE Transactions on Systems, Man, and Cybernetics (Part C) 38 (3) (2008) 280–291.
[27] J.A. Orlicky, Structuring the bill of materials for MRP, Production and Inventory Management, December 1972, pp. 19–42.
[28] S.S. Panwalkar, W. Iskander, A survey of scheduling rules, Operations Research 25 (1977) 45–61.
[29] M. Pesic, W.M.P. van der Aalst, A declarative approach for flexible business processes, in: J. Eder, S. Dustdar (Eds.), Business Process Management Workshops, Workshop on Dynamic Process Management (DPM 2006), Lecture Notes in Computer Science, vol. 4103, Springer-Verlag, Berlin, 2006, pp. 169–180.
[30] M. Pesic, M.H. Schonenberg, W.M.P. van der Aalst, DECLARE: full support for loosely-structured processes, in: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), IEEE Computer Society, 2007, pp. 287–300.
[31] M. Pesic, M.H. Schonenberg, N. Sidorova, W.M.P. van der Aalst, Constraint-based workflow models: change made easy, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2007: OTM 2007 Confederated International Conferences, Part I, Lecture Notes in Computer Science, vol. 4803, 2007, pp. 77–94.
[32] E.A.H. Platier, A logistical view on business processes: BPR and WFM concepts, Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, 1996 (in Dutch).
[33] M.L. Puterman, Markov Decision Processes, Wiley, New York, 1994.
[34] R. Ramasesh, Dynamic job shop scheduling: a survey of simulation research, OMEGA International Journal of Management Science 18 (1) (1990) 43–57.
[35] H.A. Reijers, Product-based design of business processes applied within the financial services, Journal of Research and Practice in Information Technology 34 (2) (2002) 110–122.
[36] H.A. Reijers, Design and control of workflow processes: business process management for the service industry, Lecture Notes in Computer Science, vol. 2617, Springer-Verlag, Berlin, 2003.
[37] H.A. Reijers, S. Limam Mansar, W.M.P. van der Aalst, Product-based workflow design, Journal of Management Information Systems 20 (1) (2003) 229–262.
[38] P. Resnick, H.R. Varian, Recommender systems, Communications of the ACM 40 (3) (1997) 56–58.
[39] H. Schonenberg, R. Mans, N. Russell, N. Mulyar, W.M.P. van der Aalst, Process flexibility: a survey of contemporary approaches, in: J. Dietz, A. Albani, J. Barjis (Eds.), Advances in Enterprise Engineering I, Lecture Notes in Business Information Processing, vol. 10, Springer-Verlag, Berlin, 2008, pp. 16–30.
[40] H. Schonenberg, B. Weber, B.F. van Dongen, W.M.P. van der Aalst, Supporting flexible processes through log-based recommendations, in: M. Dumas, M. Reichert, M.-C. Shan (Eds.), Proceedings of the 6th International Conference on Business Process Management (BPM 2008), Lecture Notes in Computer Science, vol. 5240, Springer-Verlag, Berlin, 2008, pp. 51–66.
[41] A. Sharp, P. McDermott, Workflow Modeling: Tools for Process Improvement and Application Development, second ed., Artech House, 2009.
[42] E.A. Silver, D.F. Pyke, R. Peterson, Inventory Management and Production Planning and Scheduling, John Wiley and Sons, Hoboken, NJ, 1998.
[43] H.C. Tijms, A First Course in Stochastic Models, Wiley, Chichester, England, 2003.
[44] P. Trkman, The critical success factors of business process management, International Journal of Information Management (2009). Q1
[45] M. Vakola, Y. Rezgui, Critique of existing business process re-engineering methodologies, Business Process Management Journal 6 (3) (2000) 238–250.
[46] I. Vanderfeesten, Product-based design and support of workflow processes, Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2009.
[47] B. Weber, M. Reichert, S. Rinderle-Ma, Change patterns and change support features: enhancing flexibility in process-aware information systems, Data and Knowledge Engineering 66 (3) (2008) 438–466.

Please cite this article as: I. Vanderfeesten, et al., Product-based workflow support, Informat. Systems (2010), doi:10.1016/j.is.2010.09.008