Product Based Workflow Support: A Recommendation Service for Dynamic Workflow Execution

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

Technische Universiteit Eindhoven,
Department of Technology Management,
PO Box 513, 5600 MB Eindhoven, The Netherlands
{i.t.p.vanderfeesten, h.a.reijers, w.m.p.v.d.aalst}@tue.nl

Abstract. Product Based Workflow Design (PBWD) is a revolutionary and successful new approach to workflow process support. A description of the product, the Product Data Model (PDM), is central in this approach. While other research so far has focused on deriving a process model from the PDM, this paper presents a way to directly execute the PDM, leading to a more dynamic and flexible support for the workflow process. Based on the information available for a case the next step to be performed is determined using a strategy of e.g. lowest cost or shortest processing time. A prototype implementing these execution recommendations is presented.

Keywords: Workflow Management, Product Data Models, Process Modeling, Process Execution Strategies.

1 Introduction

Product Based Workflow Design (PBWD) [1,10,11] is a radical, innovative approach to workflow process design. The approach can be considered as *revolutionary* because it takes a blank sheet of paper to start a design from scratch, as opposed to *evolutionary* approaches that try to improve existing situations in a more gradual style. In the past decade, PBWD has been applied in several industrial cases to redesign and improve business processes and has gained quite some successes. For example, PBWD was used to redesign the process of awarding unemployment benefits in the Netherlands. This redesign was conducted within the UWV agency (formerly known as the GAK). As a result, the average throughput time of the process was reduced with 73% and 10% of all cases now require no human intervention at all [10,11]. More recently, the annual reporting process of a large Dutch bank was redesigned with PBWD, leading to a reduction of throughput time of 50%, a fall of handovers of 60%, and a decrease of control steps of 18%. These results clearly show the potential of PBWD for the improvement of processes.

The product-based approach to process design has several advantages [14,15]. The most important advantage is its *rationality*. In the first place, because a product specification is taken as the basis for a workflow design, each recognized data element and each production rule can be justified and verified with this specification. As a result, no useless steps are executed in the process, multiple registrations of the same information will no longer happen, and it becomes clear what data manipulations can be automated. Secondly, the ordering of (tasks with) production rules themselves is completely driven by the performance targets of the design effort. This allows for process execution that is not determined by more or less arbitrary updates to the process in the past, but by the drivers that are important to an organization today.

By now, various efforts have been undertaken to specify how workflow models can be derived from product structures, either for supporting the design of a process model that is to be executed by humans [4,15] or by elaborating the automatic generation of process models [1]. Currently, the most important direction for our research is to develop automatic support for PBWD, in the form of software tools that can be used to define product specifications, to specify performance goals for workflow processes, and to generate workflow models that comply with a given product specification on the one hand and deliver the desired performance on the other [15].

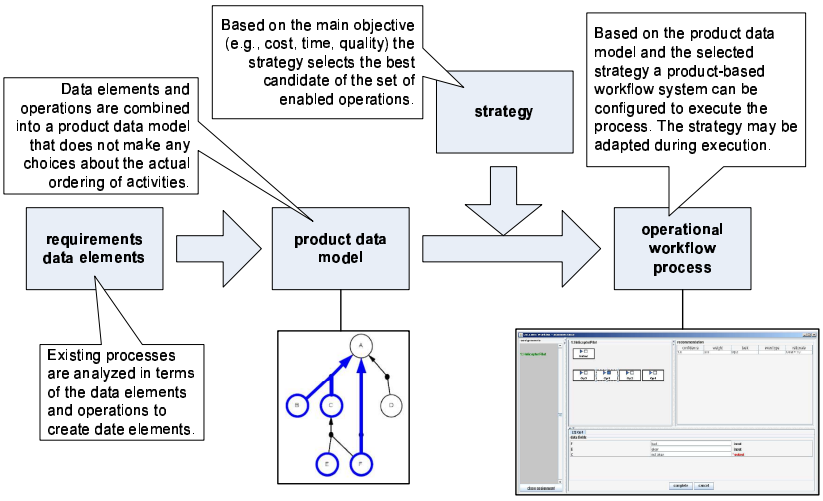


Fig. 1. Product Based Workflow Support (PBWS).

Experiences with the automatic generation of process models based on product structures, revealed many problems. First of all, it is not easy to construct a suitable process model based on a product data model. Second, a lot of the flexibility is lost in the translation process. This triggered the idea to provide flexible

and dynamic support for process execution *directly* on the basis of the product structure, i.e. without first deriving a process model that describes the desirable flow of work. This is the focus of this paper. The basic idea is that dynamically, during each step of the process execution, all data elements are determined that are available for a case. At run time, it can then be decided what would be the most proper next step in the execution of the process, considering the information available for the specific case, the underlying product specification, and the desired performance. We will refer to this concept as *Product Based Workflow Support* (PBWS).

Figure 1 shows an overview of the approach used in this paper. It is important to note that the construction of the product data model is an analytic activity, i.e., based on a detailed analysis of data elements and operations a model is created that reveals dependencies. These dependencies are based on functional requirements and not driven by performance considerations. Based on the selected strategy, the product data model is traversed in a particular way. The appropriate strategy is selected based on the performance goals (e.g., minimize costs of flow time). If the goals of an organization change, another strategy is selected without changing the product data model. By combining the product data model and the strategy, it is possible to automatically generate a workflow solution as will be shown in the remainder. In this paper we will present a prototype of our *PDM recommendation service* implemented in ProM [2], which interacts with the user via a worklist in DECLARE [8]. This worklist contains a list of recommended steps for the current case in execution.

The structure of this paper is as follows. In the next section, some background on PBWD is given, including a more detailed explanation of the execution of such a product structure. Section 3 deals with different strategies that can be used to determine the next step in the execution of a workflow product structure. After having explained the technical infrastructure in Section 4, a prototype that is based on ProM and DECLARE implementing our approach to dynamic workflow support is introduced in the fifth section. Finally, Section 6 provides some conclusions and a look into further research on this topic.

2 Product Based Workflow Support

Product Based Workflow Support differs from the more common, activity-oriented workflow support because it does not depend on a process model that is fed into a workflow management system, but by giving dynamic runtime support by computing and recommending the possible next steps considering the information that is available at a certain point in the execution of a case. The fundament for this kind of support is a description of the product that is produced by the process under consideration. Such a structure is called a Product Data Model (PDM). This section introduces the PDM concepts and contains some illustrations of its syntax and semantics.

2.1 The Product Data Model

The product of a workflow process is an informational product, for example: an insurance claim, a mortgage request, or a social benefits grant. Similar to a Bill-of-Material (BOM) from the manufacturing area [5], a product description for many informational products can be made. However, the building blocks are then not the physical parts that have to be assembled, but data elements (e.g. name, birth date, amount of salary, type of insurance and the amount of months that one is unemployed) that have to be processed to achieve new data. Such a product description, displayed as a network structure, is called a PDM.

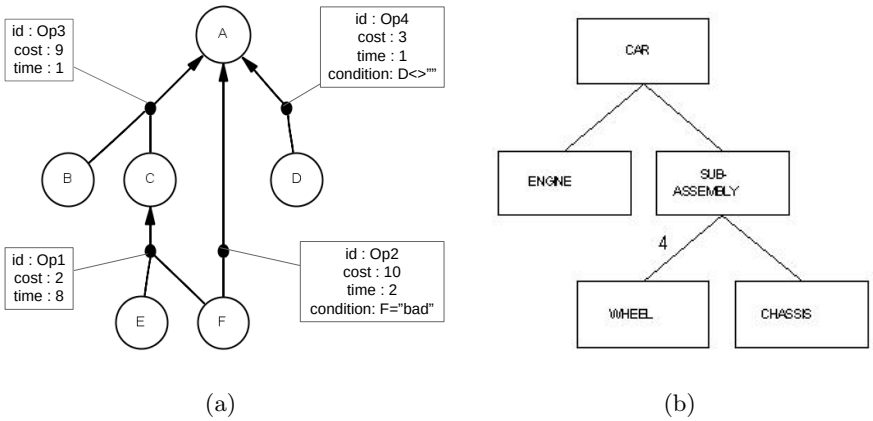


Fig. 2. (a) The product data model which represents the decision on the suitability to become a helicopter pilot. The meaning of the elements is as follows: (A) decision for suitability to become a helicopter pilot, (B) psychological fitness, (C) physical fitness, (D) latest result of suitability test in the previous two years, (E) quality of reflexes, (F) quality of eye-sight. Note that the operations have some attributes attached, such as an id and processing time. These attributes are important in selecting a suitable next step in the execution of the PDM for a specific case. (b) The Bill of Material (BOM) of a car with four wheels, a chassis and an engine.

Figure 2(a) contains a small and simple example of a PDM, comparable in complexity to the simple BOM of a car in Figure 2(b). It describes the decision whether an applicant is allowed to enter a training program to become a helicopter pilot (see also [10]). Persons that want to become a helicopter pilot should meet some requirements: they should be healthy, their eye-sight should be good enough, they should pass a psychological assessment, and they should not have been rejected in the previous two years. The figure shows that the final decision whether a person can become a helicopter pilot (data element A) is dependent either on the data elements B and C, or on F, or on D. In reality, these different combinations reflect the different conditions under which certain operations

can be executed. In case there is a result of a recent suitability test (D), this information directly determines the outcome (A). Also, in case the value for the quality of eye-sight of the applicant is extremely bad (F) this might lead to a direct rejection (A). In the other cases, the results of both a psychological (B) and a physical test (C) are needed. One level lower, the physical test (C) consists of the results for the quality of reflexes (E) and for the quality of eye-sight (F).

The *data elements* of the PDM are depicted as circles. The *operations* on these data elements are represented by arcs. The arcs are 'knotted' together when the data elements are all needed to execute the particular operation. Compare, for instance, the arcs from B and C leading to A on the one hand, to the arc from D leading to A on the other in Figure 2(a). In the latter case only one data element is needed to determine the outcome of the process (A), while in the case of B and C both elements are needed to produce A.

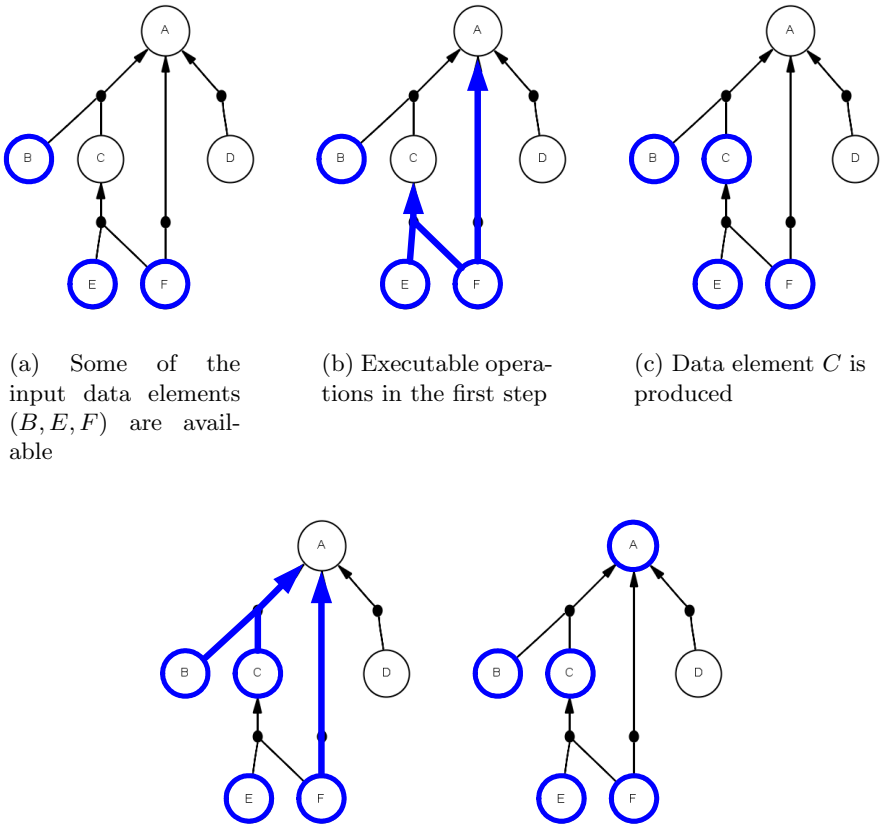
The helicopter pilot example, which we discussed here, is very small. Typically, in industry the PDMs are much larger; typically containing hundreds of data elements (see for instance the case studies described in [10]). What is important to stress here is that PDMs typically allow for a wide variety of *alternative* ways to generate the desirable end product. This is in contrast to its manufacturing antipode, where the production process has fewer alternatives because of physical constraints.

The idea to use a product data model was introduced in [1] and further detailed in [10,11,14]. Moreover, Müller et al. [4] have worked on data-driven process structures. Our approach is most related to the work of Kress, Melcher and Seese [3], who introduced so-called "Executable Product Models". In their approach agents are used to execute the steps in the workflow process. The distribution of the work items over the various agents is done by negotiations among the agents. Their goal is to optimize the workload for each agent and for the process as a whole. In contrast to this study, we are focusing in this paper on the optimal choices for the case (i.e. regarding throughput time, cost, etcetera). This idea was earlier introduced in [14] using a CPN Tools simulation model that illustrates the impact of a certain selection strategy.

2.2 Runtime execution of a PDM

Figure 3 illustrates how the runtime (step-by-step) execution of such a PDM works. Suppose that at the start of the process input data elements B, E, and F (i.e. psychological fitness, quality of reflexes, and quality of eye-sight) are available (see Figure 3(a)). The operations that are now enabled for execution are $Op1$ and $Op2$, since all of their input elements are available (Figure 3(b))[1]. Operation $Op3$ is not executable because data element C is not available yet and $Op4$ is not executable since D is not present. Now, we have to choose which of the two executable operations ($Op1$, $Op2$) we select. Suppose we take $Op1$. Then, data element C is produced (Figure 3(c)). The executable operations are

[1] For reasons of simplicity we abstract here from the execution conditions on the operations.



(a) Some of the input data elements (B, E, F) are available

(b) Executable operations in the first step

(c) Data element C is produced

(d) Executable operations in step two

(e) The end product (A) is determined

Fig. 3. The step-by-step execution of a product data model. Circles that are bold represent available data elements for the case under consideration; bold arrows indicate executable operations.

calculated again ($Op2$ and $Op3$) and one of those operations is selected. Suppose we select $Op3$. Then, the end product A is determined and the process ends.

In many situations more than one operation is executable, like in all steps of this example. Now the question can be raised how to select the best operation from the set of executable operations to proceed. It is essential to see that *functional requirements and performance considerations are mixed in conventional approaches*. A process model is typically based on both functional requirements

("this information is needed to make this decisions") and performance goals ("to minimize flow time it is better to do things in parallel"). Hence, if the goal changes, the model needs to be revised. The next section addresses how strategies can be combined with PDMs.

3 Execution strategies

As explained in the previous section, it is common that during the runtime execution of a PDM, a number of alternative operations is available for execution. Only one of them can be chosen as the next step to be performed. Choosing the best candidate depends on the performance goal(s) of the process. For instance, when it is important to produce the end product at low cost or rather at great speed one should choose the operation with the lowest cost or shortest processing time respectively.

We have identified several selection strategies to find the best candidate from the set of enabled operations. For that we were inspired by *sequencing and scheduling rules* from the field of logistics and production planning [6,12]. Short-range production planning is a method to decide, beforehand, which resource is going to perform which jobs in which order for production systems with a shop structure (i.e. flow shop, job shop er open shop) [12]. Typically, the solution to this problem is a dispatch list (also called *sequence* or *schedule*) for each resource, containing the order in which the jobs should be handled. The most favorable schedule is determined based on the objectives of the process. Well-known strategies are for instance First Come First Served (FCFS) or Earliest Due Date (EDD) [12].

These production planning problems are usually too big to be solved analytically and researchers therefore have developed pragmatic ways (e.g. heuristics) that approximate an ideal solution to a scheduling problem. Many different strategies or rules exist to schedule the sequence of jobs that have to be performed by a machine [6,12]. The following can also be translated to our selection problem:

- Random - The best candidate is randomly selected (cf. Random [6]).
- Lowest cost - The best candidate is the operation with the lowest cost.
- Shortest processing time - The operation with the shortest duration is chosen (cf. SPT [6]).
- Distance to root element - 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 [6]).
- Shortest remaining processing time - 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 (shortest remaining processing time) [6]).

Note that, using these rules, the selection of the best candidate is only optimized locally (i.e. within the set of executable operations); the effect of the selected

operation on future steps is not taken into account. A more general optimization approach is discussed in our section on future work (see Section 6).

4 Technical infrastructure

A prototype workflow system has been developed that supports the approach presented in this paper. The system is fully functional and serves as a proof of concept for Product Based Workflow Support. The prototype is based on the DECLARE [7,8] and ProM [2] frameworks. DECLARE is used to specify operations and to offer workitems to end users. However, the LTL-based engine of DECLARE is not used at all. ProM has been extended to support PDMs. Moreover, the engine of the prototype workflow system is realized in ProM. Since both DECLARE and ProM are essential components of our system, they are explained below. In the next section, we describe the new functionality that has been added.

4.1 DECLARE

DECLARE is a prototype of a workflow management system [7,8].[2] The novelty in DECLARE is that it is based on a truly declarative approach to business process modeling and execution. Unlike conventional systems, which use graph-like and activity-oriented modeling languages (e.g., Petri nets), DECLARE uses a constraint-based language. While most of the existing languages have a tendency to "over-specify" behavior (i.e., restrict people even in cases where human flexibility is essential), DECLARE tends to under-specify behavior assuming that people are not machines and are able to behave responsibly. This allows for more freedom in the execution of the model. The DECLARE system includes three tools:

- *Designer* is used to design the organizational structure, define possible relations between activities, and develop process models.
- *Framework* is the engine of the system. It executes process models.
- *Worklist* is the interface between the user and the framework.

Together these tools provide build-time and run-time support for a declarative approach to workflow management. In our approach we will not use the DECLARE process modeling language, i.e., the actual control-flow is based on the PDM and the selected strategy.

4.2 ProM

ProM is an extensible framework that supports a wide variety of process mining techniques and other process analysis methods in the form of plug-ins.[3] Currently, there are more than 190 import, export, analysis, conversion and mining

[2] DECLARE can be downloaded from http://is.tm.tue.nl/staff/mpesic/declare.htm.
[3] ProM can be downloaded from www.processmining.org.

plug-ins available [2]. Besides the import of log files and the discovery of process models from these logs, ProM also supports the import, export and conversion of models designed with several well-known modeling languages, such as (Colored) Petri Nets, EPC's, and Workflow Nets, and the verification and analysis of process models. Altogether, the ProM Framework is a powerful tool for process discovery, analysis and verification.

ProM also incorporates an analysis plug-in that provides execution recommendations based on the event log of a process [16]. In order to support a runtime process, an event log of the process is loaded into ProM. The event log contains the order in which activities for previously handled cases were executed. Based on the event log the *ProM log recommendation service* determines which activities are suitable as next steps in the running (partially executed) case. Each recommendation is given a *confidence* and a *weight* to indicate it's relative importance to other recommendations.

5 PDM recommendation tool

The prototype of the PDM recommendation tool uses the technical infrastructure described in the previous section and the selection strategies of Section 3. To deliver Product Based Workflow Support, we used the log based recommendation service in a different way than described in the previous section. Instead of using a log to decide on the best next steps we use a PDM and a selection strategy. The basic idea is shown in Figure 4.

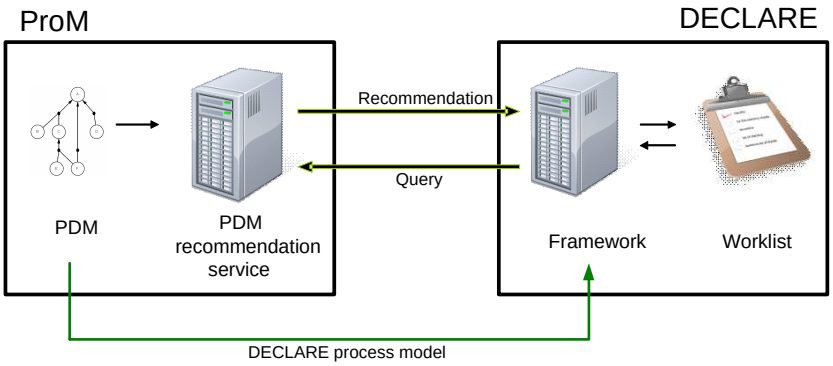


Fig. 4. An overview of the framework for PDM recommendations.

After a user has exported the PDM to a DECLARE process model containing all operations as activities (see Figure 5 for the exported DECLARE model of our example), the ProM PDM recommendation service can be started and a strategy can be selected (see the upper part of Figure 6). Then the process

model can be loaded into the DECLARE framework and actual cases can be handled.

To illustrate the functionality of PDM recommendation tool, we closely follow the steps in Figure 3. When the execution of a specific case is started, first of all the input data elements have to be filled in (cf. Figure 5). The resulting state corresponds to Figure 3(a). When this 'initial' task is completed the recommendation service calculates the recommendations (see Figure 6) and sends them to the DECLARE worklist. In the worklist the executable operations are displayed as a list and the recommended operation(s) for the immediate next step have been given a weight of '1', whereas the other operations have a weight of '0'. In this case we have selected a lowest cost strategy. Operation $Op1$ and $Op2$ are executable. The former has the lowest cost ($2 < 10$) and is therefore recommended. $Op2$ is still executable and can be chosen by the user, but since it is not recommended it is given weight 0 (Figure 7).

Then, the user selects the operation in the process model that s/he wants to perform for this case. Since $Op1$ was recommended, it is also selected. The value of the input elements is already filled in the respective fields and the user can add the value for the output element (Figure 8). Again the *ProM PDM recommendation service* calculates the enabled operations and the recommended next step, which in this case is $Op3$ because the cost of $Op3$ are lower than the cost of the other enabled operation $Op2$ (Figure 9). Finally, the user selects $Op3$ in the DECLARE worklist and fills in a value for the end product A. Since a value for the end product is produced now the process stops, although there might still be some operations enabled (e.g. $Op2$).

Using a lowest cost strategy we now have executed a case with total cost of 11: 2 for $Op1$ plus 9 for $Op2$.

In [14] a quantitative evaluation of these selection strategies is made by implementing these different strategies in a simulation model in CPN Tools. The simulation model is generic, since the PDM for a specific example is added to the model as a parameter: the PDM is represented by a list of operations and a list of available data elements. After choosing a selection strategy the model executes the given PDM step-by-step, similar to the explanation in Figure 3, but also considering a possible failure of an operation. For one specific case from industry (taken from [10]) we have executed simulation runs for all strategies. The outcome of these simulations makes it possible to compare the different selection strategies on all dimensions (i.e. cost and duration in this case). For this specific case study we can conclude for instance that a low cost strategy naturally gives lowest costs. More surprisingly, the selection of the operation with shortest processing time results in higher total costs than a random selection strategy.

6 Conclusion

This paper presents Product Based Workflow Support: a dynamic approach to workflow execution on the basis of a product data model. In contrast to conventional workflow management support, there is no need for a process model

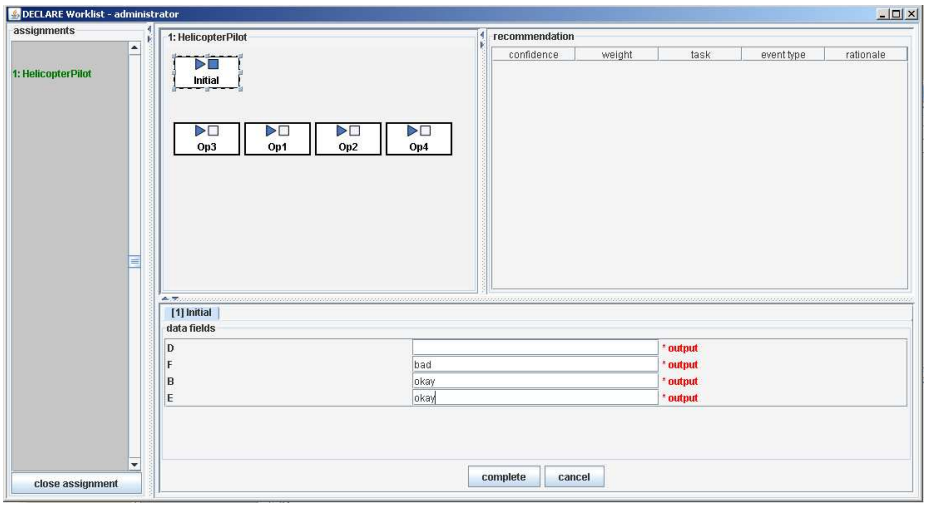


Fig. 5. As a first step a value for the input data elements has to be provided. In this example we have used the helicopter case again. A previous test result is not available ('D' is empty) and the other data elements are filled in.

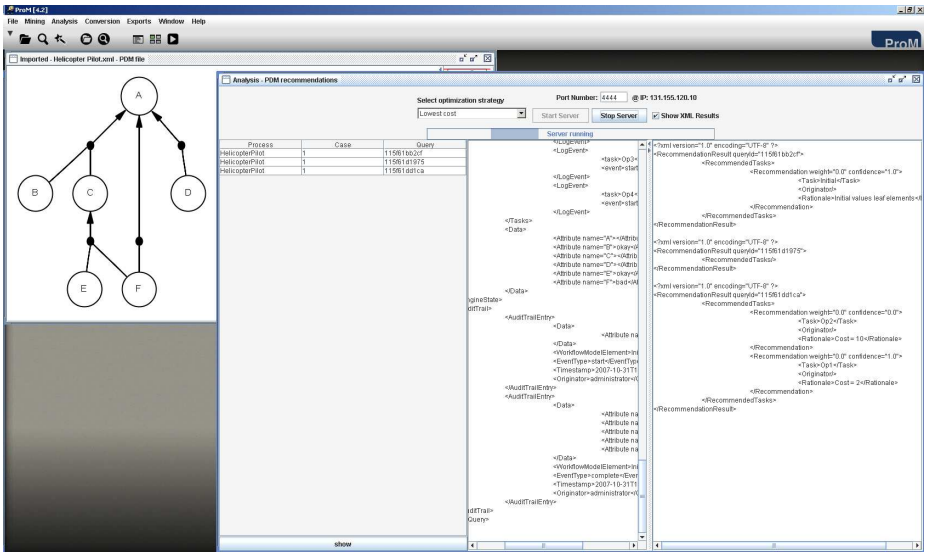


Fig. 6. The recommendation service calculates which operations are executable based on the availability of input elements. In this case only $Op1$ and $Op2$ are executable. since we have selected a lowest cost strategy $Op1$ is preferred over $Op2$ since the cost for the first one (2) are lower than for the second one (10).

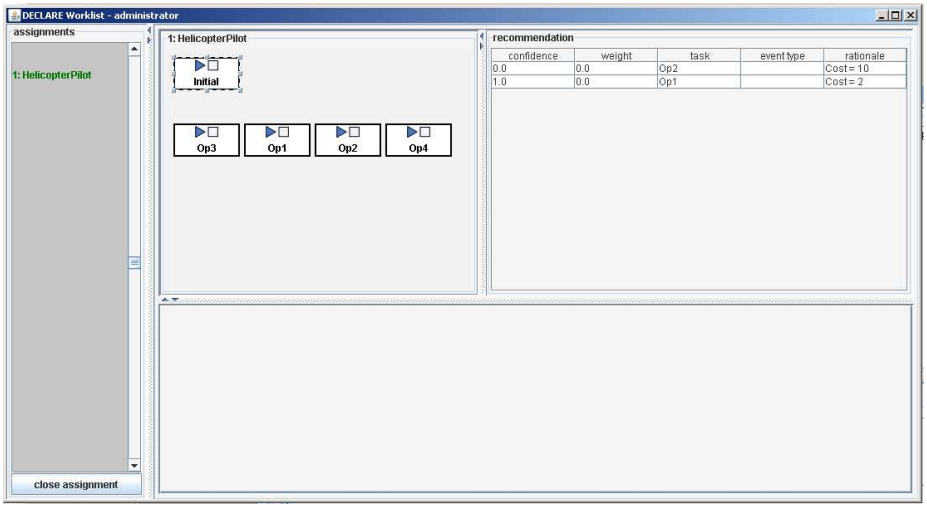


Fig. 7. Based on the data provided in the first step two operations are executable: $Op1$ and $Op2$. Under a lowest cost strategy only $Op1$ is recommended (see upper right corner).

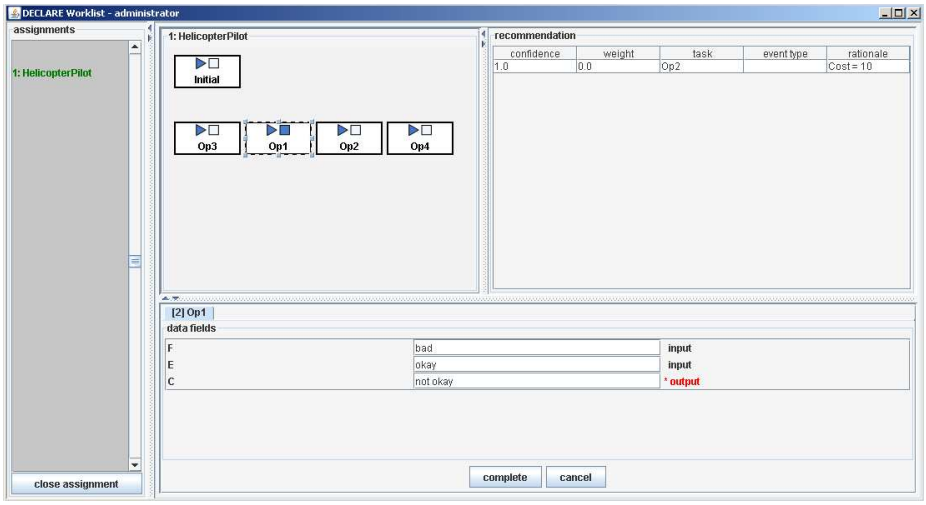


Fig. 8. The selected operation $Op1$ is executed by the user and the value for 'C' is added ('C'= 'not okay').

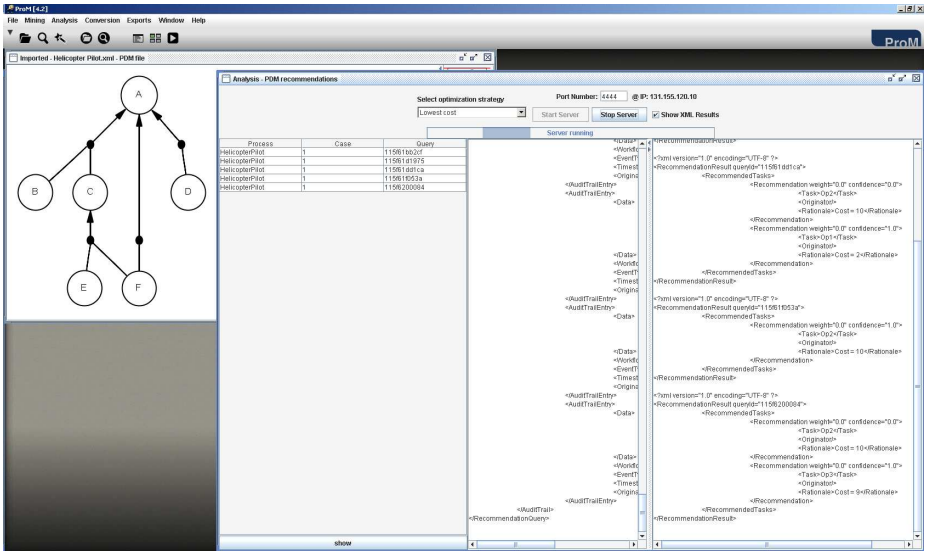


Fig. 9. Again the executable operations are calculated: $Op2$ and $Op3$. $Op3$ has the lowest cost and is recommended to the user.

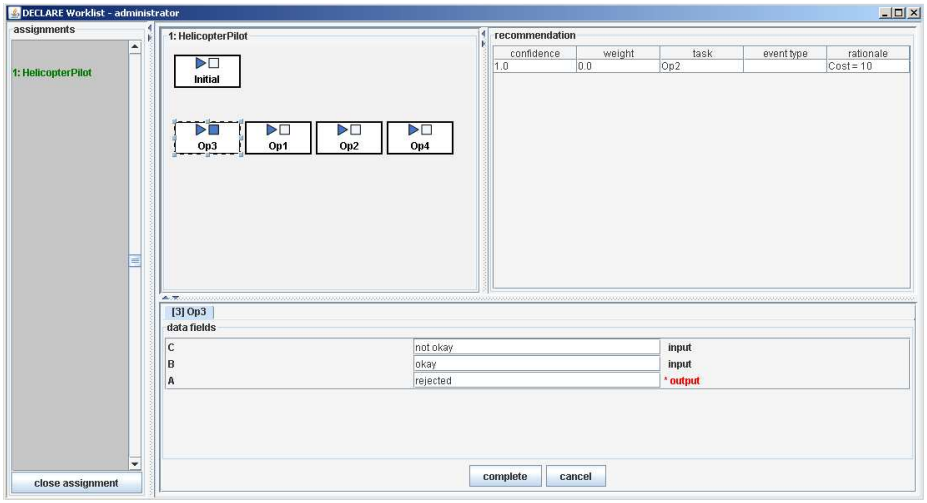


Fig. 10. The user executes $Op3$ and the end product A is produced. The process completes.

that guides the execution. Therefore, a more dynamic and flexible support is possible. Based on the data elements readily available for a specific case on the one hand and a selected strategy (i.e. lowest cost, shortest processing time, etc.) on the other this approach recommends the next step that should be performed for the case. *In contrast to conventional languages there is a clear separation of concerns: the product data model is based on functional requirements while the selected strategy focuses on performance* (e.g., minimize costs).

This paper presents a fully operational prototype workflow system that supports the approach presented in this paper. For user interaction the *Declare worklist* was used, while the core engine, the so-called *PDM recommendation service*, was implemented in the *ProM framework*.

An extension to this prototype could be the implementation of more strategies. One could, for instance, think of selecting the operation with the lowest failure probability or the highest chance of being a knock-out to the process. Also, a combination of two selection strategies is possible. This might particularly be helpful if the operations in the set of operations do not differ too much in one dimension. For instance, when there are several operations with the same minimum processing time (under a shortest processing time strategy), it might be a good idea to select the operation with lowest cost or perhaps the smallest failure probability.

As we have noted in Section 3, one of the limitations of the current approach is that the presented strategies for selection of the best candidate only perform a *local* optimization, which not necessarily leads to the best overall path to the end product. For instance, in the example, which we used to explain the syntax and semantics of a PDM in Section 2.2, it is overall wiser to select $Op2$ in the first step, considering the costs. By doing so, the overall costs will amount to 10 instead of 11. To overcome the problem of local optimization, the use of the theory of Markov Decision Processes (MDP) is a promising direction [9,13]. With this analytical method, it is possible to completely compute the optimal strategy.

Finally, we are also working together with our industrial partners in this research program to see how recommendation services can be incorporated in more conventional approaches to workflow management support. Our first demonstration of the prototype in the fall of 2007 has led to the commitment to develop a joint prototype in 2008. We hope that this can be a stepping stone towards the industrial spread of product-based approaches to workflow.

Acknowledgement

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. We would like to thank Maja Pesic for her work on DECLARE. Moreover, we thank the many people working on ProM. Their work enabled us to realize the workflow system presented in this paper.

References

1. W.M.P. van der Aalst. On the automatic generation of workflow processes based on product structures. *Computers in Industry*, 39:97–111, 1999.
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, and A.J.M.M. Weijters. ProM 4.0: Comprehensive Support for Real Process Analysis. In J. Kleijn and A. Yakovlev, editors, *Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007)*, volume 4546 of *Lecture Notes in Computer Science*, pages 484–494. Springer-Verlag, Berlin, 2007.
3. M. Kress, J. Melcher, and D. Seese. Introducing executable product models for the service industry. In *Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS '07)*, page 46. IEEE Computer Society, 2007.
4. D. Müller, M. Reichert, and J. Herbst. Flexibility of data-driven process structures. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops*, volume 4103 of *Lecture Notes in Computer Science*, pages 181–192. Springer, 2006.
5. J.A. Orlicky. Structuring the bill of materials for mrp. *Production and Inventory Management*, pages 19–42, Dec 1972.
6. S.S. Panwalkar and W. Iskander. A survey of scheduling rules. *Operations Research*, 25:45–61, 1977.
7. M. Pesic and W.M.P. van der Aalst. A declarative approach for flexible business processes. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops, Workshop on Dynamic Process Management (DPM 2006)*, 2006.
8. M. Pesic, H. Schonenberg, and W.M.P. van der Aalst. DECLARE: Full Support for Loosely-Structured Processes. In M. Spies and M.B. Blake, editors, *Proceedings of the Eleventh IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, pages 287–298. IEEE Computer Society, 2007.
9. M.L. Puterman. *Markov Decision Processes*. Wiley, 1994.
10. H.A. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*, volume 2617 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2003.
11. H.A. Reijers, S. Limam Mansar, and W.M.P. van der Aalst. Product-based Workflow Design. *Journal of Management Information systems*, 20(1):229–262, 2003.
12. E.A. Silver, D.F. Pyke, and R. Peterson. *Inventory Mangement and Production Planning and Scheduling*. John Wiley and Sons, Hoboken, NJ, 1998.
13. H.C. Tijms. *A First Course in Stochastic Models*. Wiley, 2003.
14. I. Vanderfeesten, W.M.P. van der Aalst, and H.A. Reijers. Modelling a Product Based Workflow System in CPN tools. In K. Jensen, editor, *Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2005)*, volume 576 of *DAIMI*, pages 99–118, Aarhus, Denmark, October 2005. University of Aarhus.
15. I. Vanderfeesten, H.A. Reijers, and W.M.P. van der Aalst. An evaluation of case handling systems for product based workflow design. In Vitor Pedrosa, editor, *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS 2007)*, pages 39–46, Funchal-Madeira, Portugal, June 2007. INSTICC.
16. B. Weber, B.F. van Dongen, M. Pesic, C.W. Guenther, and W.M.P. van der Aalst. Supporting Flexible Processes Through Recommendations Based on History. BETA Working Paper Series, WP 212, Eindhoven University of Technology, Eindhoven, 2007.