Automatic support for product based workflow design : generation of process models from a product data model

Automatic Support for Product Based Workflow Design: Generation of Process Models from a Product Data Model

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

Technische Universiteit Eindhoven,
Department of Industrial Engineering and Innovation Sciences,
School of Industrial Engineering,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{i.t.p.vanderfeesten,h.a.reijers,w.m.p.v.d.aalst,j.j.c.l.vogelaar}@tue.nl

Abstract. Product Based Workflow Design (PBWD) is one of the few scientific methodologies for the (re)design of workflow processes. It is based on an analysis of the product that is produced in the workflow process and derives a process model from the product structure. Until now this derivation has been a manual task and is therefore a time-consuming and error-prone exercise. Automatic support would enhance the use of the PBWD methodology. In this paper we propose several algorithms to automatically generate process models from a product structure and we present a software tool (implemented in ProM) to support this. Finally, the properties of the resulting process models are analysed and discussed.

Keywords: workflow product, process model, Product Based Workflow Design.

1 Introduction

In the field of Business Process Management (BPM) only a few scientific approaches exist that address the issue of how to actually *design* a process or, as in many companies 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. [4,6]. Understandably, it is often said that process design is "more art than science".

One of the notable exceptions is Product-Based Workflow Design (PBWD) [15]. PBWD has been developed as a method for process redesign that is repeatable, objective, and effective. The focus of this method is on the design of processes that deliver informational products, the so-called *workflow processes*. The PBWD methodology takes the structure of the informational product, which is described in a Product Data Model (PDM), as a starting point to derive a process model.

PBWD has proven to be a valuable methodology used by various consultancy and service companies to improve the performance of various business

R. Meersman et al. (Eds.): OTM 2010 Workshops, LNCS 6428, pp. 665–674, 2010.

processes in the services domain, see e.g. [14]. However, the derivation of process models from a product structure has mainly been a manual task and therefore a time-consuming and error-prone exercise. Automatic support for this step would certainly enhance the use of the PBWD methodology.

Our contribution, presented in this paper, is focused on the automatic derivation of process models from a PDM. We propose a number of algorithms that generate a process model solely based on the structure of the product. These algorithms are implemented in ProM to show their feasibility and to provide automatic support. This paper shows that various process models may be derived from the same product structure. We analyse and discuss the differences between those process models based on their properties.

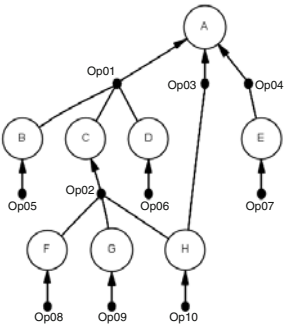
The paper is structured as follows. In Section 2, the notion of the PDM is explained and illustrated with an example. Next, the algorithms for the automatic generation of process models from a PDM are introduced and discussed in Section 3. Finally, related work is discussed in Section 4 and Section 5 concludes this paper.

2 The Product Data Model

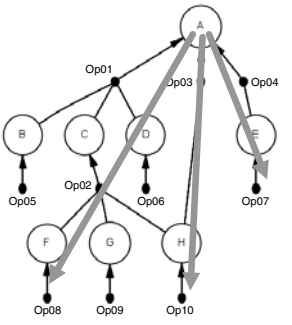
The PBWD methodology takes the structure of the workflow product as a starting point to derive a process model. The product of a workflow process is an informational product, for example: an insurance claim, a mortgage request, or a social benefits grant. Similar to a Bill-of-Materials (BOM) from the manufacturing area [13], a product description for many informational products can be made. However, the building blocks of this product structure are 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 Product Data Model (PDM).

Figure 1(a) contains a small and simple example of a PDM. It describes the calculation of the maximum amount of mortgage a bank is willing to loan to a client. The figure shows that the maximum mortgage (element A in Figure 1(a)) is dependent either on a previous mortgage offer (E), or on the registration in the central credit register (H), or on 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 from 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).

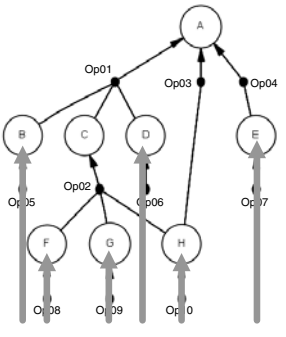
The *data elements* of the PDM are depicted as circles. The *operations* on these data elements are represented by (hyper)arcs: the arcs are 'knotted' together when the data elements are all needed to execute the particular operation. Compare, for instance, the arcs from B, C and D leading to A on the one hand, to the arc from H leading to A on the other in Figure 1(a). In the latter case only one data element is needed to determine the outcome of the process (A),



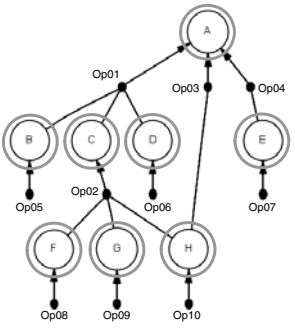
(a) The product data model (PDM) of the mortgage example.



(b) A top-down order of walking through the PDM: starting from the root element walking to the leafs.



(c) A bottom-up order of walking through the PDM: starting from the leaf elements walking to the root element.



(d) A middle-out order of walking through the PDM: data elements and operations are selected arbitrarily.

Fig. 1. An example of a PDM (a), and a visualization of the different orders (see (b), (c), and (d)) in which an algorithm can translate the PDM of (a) to a process model

while in the case of B, C and D all three elements are needed to produce A. An operation is *executable* when a value for all of its input elements is available.

The mortgage example, which we discussed here, is very small. In industry the PDMs are much larger; typically containing hundreds of data elements (see for instance the case studies described in [14]). 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.

3 Generation of Process Models

In the traditional PBWD methodology, the PDM is used as a starting point for designing process models. The design of a process model as a manual task is

time-consuming and error-prone which illustrates the need for automatic support. In this section, we introduce a number of algorithms to generate process models based on a PDM. These algorithms only take into account the structural properties of the PDM, i.e. the data dependencies described by the PDM. It will be shown that only looking at these properties may already lead to various different process models for the same PDM. In this section we will first discuss the different ways to compose the process models, after which we will discuss the implications for the resulting models in Section 3.2.

3.1 The Algorithms

The algorithms presented here all build a process model step-by-step by 'walking through' the PDM. We have used three different strategies to walk through the PDM to build a process model: (i) a top-down strategy, (ii) a bottom-up strategy, and (iii) a middle-out strategy. These strategies will be explained in more detail below, including a short description of the algorithms and an illustration of the resulting process models. Due to space limitations not all process models are presented. Only those process models are shown that are necessary to analyse and discuss the results with respect to their different properties. A full explanation of the algorithms may be found in [8].

Top-down. With a top-down strategy to build the process model, we start at the top of the PDM and walk towards the leaf elements by following the data element dependencies in the PDM. Figure 1(b) illustrates this top-down order. We use *backward chaining* to 'climb down the tree'. This leads to the determination of all minimal execution paths of the PDM which contain no superfluous operations.

We have developed two algorithms with a top-down strategy: algorithm Alpha and algorithm Bravo. The process model for the mortgage example of Figure 1(a) resulting from algorithm Alpha is displayed in Figure 2. The difference between algorithms Alpha and Bravo is in the way in which data elements and operations are translated to activities in the process model. The process model generated by algorithm Alpha has a focus on data elements, i.e. the activities in the process model all relate to data elements, except for a few routing constructs. In contrast to that, algorithm Bravo focuses on the operations in the PDM to be translated to activities in the process model. This leads to two process models, with a similar structure but differences in the details.

Middle-out. In a middle-out strategy, data elements and operations are translated to transitions in the process model in an arbitrary order, not following the dependencies between data elements in a top-down or bottom-up manner (see Figure 1(d)). The dependencies between data elements and operations are added to the model later by checking the in- and output elements of an operation.

We have developed three algorithms with a middle-out strategy: algorithms Charlie, Delta and Echo. Figures 3 and 4 show the process models for the mortgage example generated by algorithms Charlie and Echo. Algorithm Delta uses

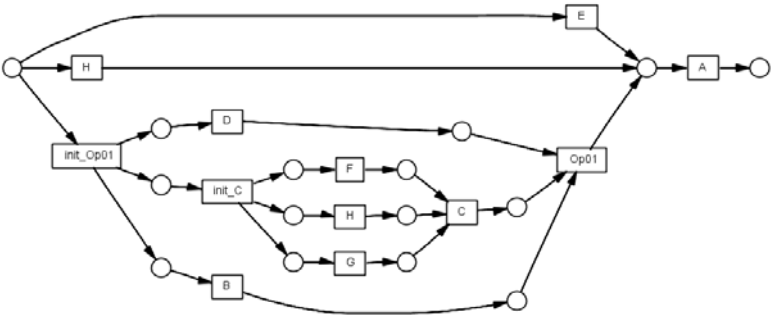


Fig. 2. The process model for the mortgage example generated by algorithm Alpha

the same basis as algorithm Charlie, but contains additional 'control places' and generates a YAWL [7] model instead of a classical Petri Net [12] to be able to use cancellation regions in the model. The difference between algorithms Charlie and Echo again is in the focus of the activities. Algorithm Charlie starts with an activity for each data element and links these activities when necessary through activities related to operations. In algorithm Echo, the focus is on translating operations to activities first. Next, these activities are linked through activities named after the data elements. This leads to two process models, with a similar structure but differences in the details (see Figures 3 and 4).

Bottom-up. Using a bottom-up strategy, we start with the leaf elements and walk towards the root element (see Figure 1(c)). Based on the data elements for which a value is available, it is determined which operations are executable as a next step in the process and which new data element values can be determined by these operations. Using this approach, we 'climb up the tree' from the input elements via the operations to their output elements. The determination of new steps is based on the availability of data element values. In general, the bottom-up order of walking through the PDM gives all possible execution paths of the PDM.

We have developed one algorithm with a bottom-up strategy: algorithm Foxtrot. A fragment of the resulting process model for the mortgage example of Figure 1(a) is displayed in Figure 5. Because the process model contains all possible execution paths, the model is very large.

3.2 Discussion of Results

Figures 2-5 all present a process model for the mortgage example. From these figures it may be clear that the process models are very different although they are generated from the same PDM. In this section, the main differences between the algorithms are discussed.

Concurrency. Concurrency means that parallel behavior during the execution of the process is allowed, e.g. several activities may be executed for the same case

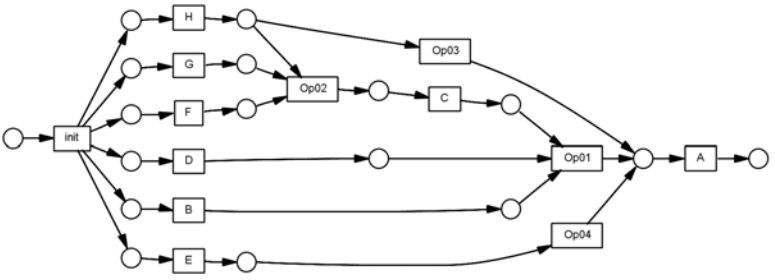


Fig. 3. The process model for the mortgage example generated by algorithm Charlie

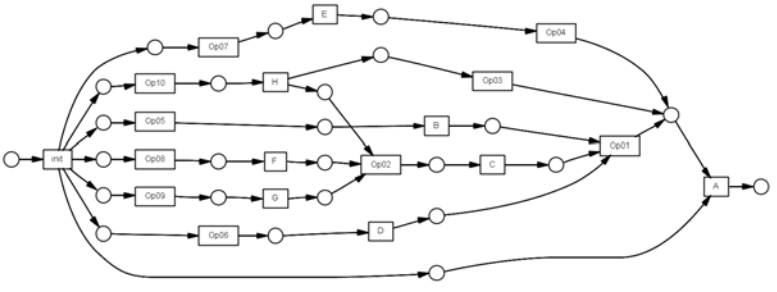


Fig. 4. The process model for the mortgage example generated by algorithm Echo. Note the differences in structure with the process model of algorithm Charlie: e.g. there are three different final transitions producing A ($Op03$, $Op04$, $Op01$ vs. A), and transition H has two output places instead of one.

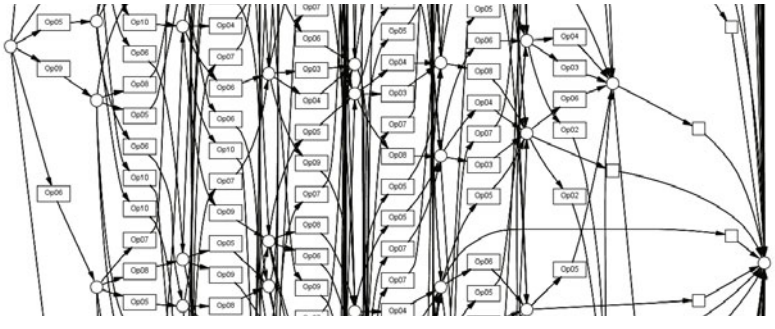


Fig. 5. A fragment of the process model for the mortgage example generated by algorithm Foxtrot. The complete process model contains 438 transitions.

at the same time. Algorithms Alpha, Bravo, Charlie, Delta and Echo all result in process models that allow for concurrency when the PDM contains an 'AND' construct such as the operation that produces a value for data element 'A' based on the values for data elements 'B', 'C' and 'D' in Figure 1(a). In contrast, the process model of algorithm Foxtrot only contains sequential behavior.

Moment of choice. With the moment of choice we indicate the point in time during execution at which a decision on alternative paths has to be made. When the PDM contains alternative paths to produce a value for a data element, this may be reflected in the process model by alternative branches ('XOR' constructs in the Petri Net) that have to be chosen 'early' or 'late' in the execution process. If the moment of choice in a process model is 'early', it has to be decided from the start which of the alternative branches is followed. If the moment of choice is 'late' this decision is deferred to a later point in time and the execution of the process is already started. Early choices in the process model are the case when using algorithm Alpha and Bravo. Figure 2 illustrates this by showing that the start place of the Petri net already is a choice for one of the branches to the end product. In the process models generated by algorithms Charlie, Delta, Echo and Foxtrot, choices are deferred, but this may also lead to the production of data element values that are not strictly needed.

Soundness. A process model is sound [1] if it can always terminate properly, i.e. it has a single token in the end place and all other places are empty. In addition, there must be no dead transition. Process models generated by algorithms Alpha, Bravo, Delta, and Foxtrot are sound per definition since this is enforced by the algorithm. Algorithms Charlie and Echo do not necessarily produce sound process models. Depending on the constructs in the PDM (e.g. duplicate use of a data element value, or alternative operations producing the same output element) the process models may contain possible execution traces in which deadlocks occur or tokens are left behind. In the process model in Figure 3 for example, tokens may be left behind after the end place is reached. This may eventually lead to several executions of transition 'A'. Also, in this process model $Op02$ may not be executable anymore if $Op03$ has fired. These situations are not possible in the process model of Figure 4 because of the different structure of the model. However, the process model of algorithm Echo is still not sound.

Table 1. This table contains a classification of the algorithms based on the criteria: (i) parallel execution, (ii) late choices, (iii) soundness

	Alpha	Bravo	Charlie	Delta	Echo	Foxtrot
Parallel execution	+	+	+	+	+	-
Late choices	-	-	+	+	+	+
Soundness	+	+	-	+	-	+

The above presented similarities and differences between the process models generated by the various algorithms are summarized in Table 1. Each of the process models has its own particularities and properties. Selecting the best algorithm to derive a process model from a PDM is therefore not trivial, but may be guided by the desired properties of the process model, e.g. if a sound process model is desired algorithms Alpha, Beta or Delta should be used, or the process model generated by one of the other algorithms should be adapted.

3.3 Tool Support

The algorithms presented in this paper are implemented as *conversion plugins* in ProM. ProM [5] 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 [5]. ProM has a plug-able architecture. ProM 5.2 has 286 plug-ins, each realizing a particular type of functionality. Each PDM conversion plugin that we have developed takes a PDM as input and produces the process model (represented by a Petri net or by a YAWL model) as output. Figure 6 shows a screenshot of the ProM environment with an example PDM, the list of conversion plugins that can be applied to this PDM, and a number of process models generated by the different algorithms. Using the basic functionality which is already present in ProM, these process models can be converted to other languages, e.g. a Petri net can be converted to an EPC, or exported to a file that can be loaded into another system. In addition, process model analysis and verification can be done using the functionality of ProM.

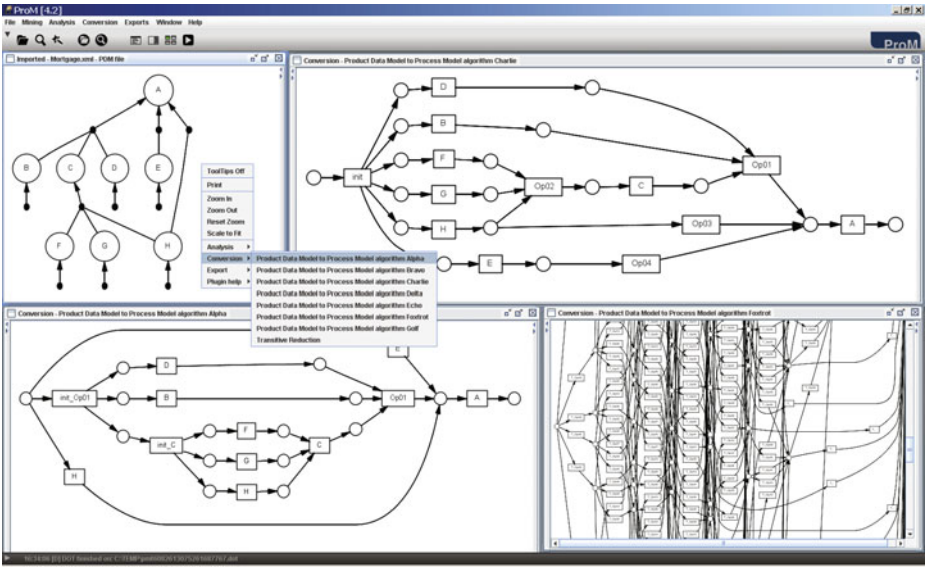


Fig. 6. A screenshot of ProM showing the conversion plugins available for a PDM

4 Related Work

The idea of using product structures to design a process model for a workflow process was introduced in [2] and further detailed in [14,15]. The former article also presents a (theoretical) algorithm for the derivation of a process model from the product structure, but it was not implemented. This first algorithm has been the foundation of the research presented in this paper and has served as a direct basis for algorithm Alpha.

Related to this work is the research on object life cycles and business processes. Müller et al. [11] derive large process models based on the life cycle information of the elements in the (product) data structure. Each element in the (product) data structure has a subprocess that describes the life cycle of this element. Also, Küster et al. [9] present a technique to automatically derive compliant process models from given life cycles of the objects in the process.

Browne et al. [3] propose to use goal graphs to structure workflow processes in the healthcare domain. A goal graph describes the goals to be achieved to cure a patient with a certain disease. They define a mapping from the goal graph to a workflow process model in which every goal is translated to a sub process that is designed to achieve the specific goal.

Finally, a number of other approaches for modelling business processes based on products have been proposed, e.g. artifact centric design [10], and document centric modelling [16]. However, these approaches only focus on the manual task of designing a process model and do not provide (automatic) support nor any guidance for this task yet.

5 Conclusion

Automatic support is a necessary step towards the larger applicability of the PBWD method. This paper presents several algorithms for the automatic derivation of process models from a PDM and shows that a single PDM may lead to very different process models. The similarities and differences between the properties of the process models generated by these algorithms are discussed as a basis for selecting the most suitable algorithm in a certain situation.

The algorithms presented in this paper are only based on the structure of the PDM and do not take into account other information that can be added to the PDM, such as resource information, knock-out probabilities, costs, and duration of an operation. Considering these kinds of data when generating a process model would lead to an even larger variety of process models to support the execution of the process which would be worthwhile to explore in further investigations.

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.

References

1. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
2. van der Aalst, W.M.P.: On the Automatic Generation of Workflow Processes based on Product Structures. Computers in Industry 39, 97–111 (1999)

3. Browne, E.D., Schrefl, M., Warren, J.R.: Goal-Focused Self-Modifying Workflow in the Healthcare Domain. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS 2004), pp. 1–10 (2004)
4. Davenport, T.H.: Process Innovation: Reengineering Work through Information Technology. Harvard Business School Press, Boston (1993)
5. van Dongen, B.F., Alves de Medeiros, A.K., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A New Era in Process Mining Tool Support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
6. Hammer, M., Champy, J.: Reengineering the Corporation. Nicolas Brealey Publishing, London (1993)
7. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N.: Modern Business Process Automation: YAWL and its Support Environment. Springer, Heidelberg (2010)
8. Kamphuis, J., Vanderfeesten, I., Reijers, H.A., van Hattem, M.: From Product Data Model to Process Model. BETA Working Paper Series, WP 238, Eindhoven University of Technology, Eindhoven (2008)
9. Küster, J.M., Ryndina, K., Gall, H.: Generation of Business Process Models for Object Life Cycle Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 165–181. Springer, Heidelberg (2007)
10. Liu, R., Bhattacharya, K., Wu, F.Y.: Modeling Business Contexture and Behavior Using Business Artifacts. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 LNCS, vol. 4495, pp. 324–339. Springer, Heidelberg (2007)
11. Müller, D., Reichert, M., Herbst, J.: Data-Driven Modeling and Coordination of Large Process Structures. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 131–149. Springer, Heidelberg (2007)
12. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE 77(4), 541–580 (1989)
13. Orlicky, J.A.: Structuring the Bill of Materials for MRP. Production and Inventory Management, pp. 19–42 (December 1972)
14. Reijers, H.A.: Design and Control of Workflow Processes. LNCS, vol. 2617. Springer, Berlin (2003)
15. Reijers, H.A., Limam Mansar, S., van der Aalst, W.M.P.: Product-Based Workflow Design. Journal of Management Information systems 20(1), 229–262 (2003)
16. Wang, J., Kumar, A.: A Framework for Document-Driven Workflow Systems. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 285–301. Springer, Heidelberg (2005)