Changing Products, Changing Processes:
Dealing With Small Updates in Product-Based Design

Hajo A. Reijers1, Jan Vogelaar1,2, Irene Vanderfeesten1

1School of Industrial Engineering
Eindhoven University of Technology,
The Netherlands
2Process Management Group
Provincie Noord-Brabant,
The Netherlands

Abstract – In the method of “product-based design” (PBD), a direct relation exists between the design of a business process and the characteristics of the end product that such a process should deliver. This offers many benefits over more traditional business process redesign approaches. This paper takes into consideration that any end product is subject to (frequent) change. While it is technically feasible with PBD to generate a new design from an updated product specification, it may be more efficient to update the existing process design directly when a change is small. In this paper, such small product changes are investigated and it is described how a process design should be changed in accordance with them. The presented results contribute to an improved application of PBD in practice in the field of process management.

Keywords: process redesign; product-based design; product data model; modification.

I. Introduction

Product-Based Design (PBD) [6] is a relatively young method for the redesign and improvement of administrative business processes (or workflows). Its characteristic focus is on the end product of a process. Where mainstream business process redesign approaches often assume an evolutionary style, where an existing process is improved in an incremental fashion, PBD first investigates the product that is to be delivered by the process, and then delivers a complete design for a process that produces this product in a way that is optimal with respect to criteria like operational cost, cycle time, etc. As such, PBD can be characterized as a revolutionary approach: a new process design is generated directly and in full.

In comparison with other approaches, PBD is time-consuming in terms of analyzing and determining the product specification, which is also known as the product data model. However, the generation of a design in the form of a process model is then straightforward, automatic, and rational.

In this paper, the link between changes in a product data model and its effects on the corresponding process model are discussed. It is argued that it is sensible to directly adjust a process model that is generated with PBD if updates to the according product data model are small. (The alternative is to again generate a complete new design.) The direct relation between such small changes in the product data model on the one hand and the process adjustments needed on the other hand is discussed in this paper. These insights are considered to lead to efficiency gains in the domains where PBD is actively applied, like banks, insurance companies and the government.

Section II describes in short the origin and applications of PBD in practice. In Section III a running example is introduced and explained, together with the corresponding process model. Section IV describes several changes to the product data model and how these affect the corresponding process models, while Section V concludes the paper.

II. Background

PBD builds on ideas that were first published in [1], where the concept of a Bill of Material was used as starting point for business process redesign. This idea was elaborated in [6] into a complete, formally grounded design method. Recently, it has been extended with various algorithms to derive process designs from product data models, as well as automated tools to do so [7]. PBD is in active use by various organizations, including the governmental organization that one of the authors is affiliated with. Previous applications include the redesign of a process for awarding unemployment benefits in the Netherlands, where, as a result, the
average throughput time of the process was reduced by 73% [4]. Another successful application of PBD aimed at redesigning the process of handling credit applications for commercial parties at a large bank in the Netherlands. The evaluation of the project showed an efficiency increase of 40% [5]. Finally, the redesign of an annual reporting process at the same bank led to a reduction in throughput time of 50% [7].

Because of the active, industrial usage of PBD new issues emerge, in particular when dealing with the maintenance of product data models and previously derived process designs. This is the subject of this paper.

III. Running Example

In this section, a running example will be introduced. A product data model always represents some form of end product that is to be produced, and the specification of the data elements that are needed to produce the end product. Figure 1 depicts a simplified product data model for an imaginary grant.

The data element at the top, the root element labelled $A$, is the final decision to grant a particular subsidy (or not). This is the end product of the “production” process that the workflow entails. To be able to make this decision, in this example, two other pieces of information are assumed to be needed, being the financial situation of the company requesting the grant (data element $B$) and information about potential other grants already received by the company (data element $C$). Data element $B$ in fact also covers information about the request itself.

These two pieces of information need to be determined, to be able to make the final decision, therefore a conjunctive connection exists between them (indicated by the knotted lines between $A$, $B$, and $C$).

Data elements $D$, $E$ and $F$ respectively represent the amount of money the company requesting the grant has available for the project itself, the amount of money requested as grant from the authorities, and the subject of the project. This last data element also holds the motivation and goals that are to be reached with the grant. The information in this data element is supposed to convince the authorities the company should get the money for the project it is willing to start. Data elements $D$, $E$ and $F$ are all needed to determine whether the grant request is acceptable. Therefore, all three data elements are connected to data element $B$ through one operation. Data element $C$ represents a decision on the acceptability of the grant request, based on other grant requests made by the same company. For some grants, for example, it is not allowed to get more than a certain amount of money in grants in total or not to get more than a certain amount of money in grants for a specific subject within some period. To be able to determine whether the grant request is acceptable from this view, either data element $G$, or data elements $H$ and $I$ are needed.

Data element $G$ represents the information that the company is unknown to the authorities to have ever requested any grants before. This particular piece of information is supposedly easy to uncover, and entails that there are no further considerations to be made concerning certain rules about total amounts of grants.

Data element $H$ holds information about the total amount of money in other grants that has been received within the last year. Data element $I$ represents the rules to which the specific grant request is to be held.

Using the information in data elements $H$ and $I$, the consideration can be made whether the grant request can be allowed within the rules.

Adopting this product data model, to reach the end decision on a grant request, several information needs need to be satisfied. On the one hand, the assessment of the request itself needs to be done; on the other hand, the grant request needs to be
considered with respect to other requests by the same company. Within either of these two decision branches, a negative result can occur, which would mean a negative decision about the entire grant. Therefore, it is attractive to handle these branches in series when the objective is to reduce costs or effort, or to process them in parallel when the aim is to reduce cycle time.

There are different ways of generating a process model from the product data model. A selection can be made on basis of the desirable properties of the resulting process (model), but also on the properties of the generating algorithm, which influences the resulting process model as well. A number of algorithms is available at this time, to be able to generate process models automatically from a product data model. These algorithms are named after the international radiotelephony spelling alphabet and are illustrated in [7] and [3]. For this running example, an algorithm that allows for parallel actions is chosen that puts the focus on data elements, i.e., algorithm Alpha. The resulting process model is depicted in Figure 2.

Algorithm Alpha represents each data element and each operation with a corresponding transition in the process model. The algorithm starts with an input- and output place and one transition, and then progressively replaces the transitions with more detailed subnets. The transitions named after letters, represent the data elements carrying the same name.

The first transition in the process model is needed to make the model a correct workflow net cf. [2], i.e., with one start place. From there, through the initializing transitions each transition producing a leaf element is enabled, i.e., the data elements at the bottom of the product data model. Next, the values of the other data elements can be determined, e.g., leaf elements D, E, F are needed to produce B.

IV. Updates in product data model
When a product data model has been constructed and its corresponding process model has been implemented, it is very common that rules and legislations change over time. As a consequence, changes in the product data model occur. Following the logic of PBD, this would mean that the process model should be regenerated from the new product data model with every change. This can lead to a big burden for the organization applying PBD. After all, it should be considered that such a process model – even though it is automatically generated – must be manually extended with all kinds of information to embed it within an organization. For example, in such a model links to legacy applications should be incorporated at the proper places so that they are correctly invoked when the process model is enacted. Other information that must be manually added, corresponds to work instructions, visual interface issues, checks, etc. A better understanding of the relationship between process model and product data model can therefore be helpful to anticipate changes to the process model, based on changes to the product data model. In other words, this means that process models should be able to be tweaked such that they still correspond to a changed product data model, saving enormous amounts of time. Another merit is the increase of insight in the effect of changes to the product data model, which can be used in the construction or alteration of the product data model.

As discussed in Section III there are currently several algorithms implemented to automatically generate process models from product data models.
Those algorithms, seven in total, all result in a different process model when applied on the same product data model, supporting different preferences for what an 'optimal' process looks like. Some of the differences between the algorithms relate to the (im)possibility to have actions performed in parallel, the moment of choice in the process model – which can be late or early – and the possibility to continue actions when the end product has already been determined, which is desirable in some situations. All properties and algorithms are described in detail in [7] and [3]. Considering these differences, it is sensible to consider the relation between an update in the product data model and an update in the process model for each of these algorithms separately.

In this paper four change primitives are studied, and they are considered to be the only elemental changes needed to accomplish any other change to a product data model. These change primitives are (1) the addition and (2) deletion of data elements and (3) the addition and (4) deletion of operations. The move of an operation from one place in the product data model to another can, for example, be accomplished by first deleting it, and later adding it in the right place.

All four change primitives are investigated for all seven algorithms to determine the effects of changes to the product data model, on the corresponding process model. All possible combinations of change primitive and algorithm are discussed in the technical report [8]. In the remainder of this section a small, representative selection is discussed.

In Figure 1 the basic product data model is depicted, with its corresponding process model generated through algorithm Alpha, in Figure 2.

The first change considered in this paper is the deletion of a data element. Data element $C$ is deleted from the product data model and, subsequently, the sub tree holding several data elements and operations. It is possible that the legislation changes in the described case, and other grants are no longer taken into account for new grants. This would mean that data element $C$ and, consequently, the branch starting from there is no longer relevant and can be deleted from the product data model.

The resulting product data model is depicted in Figure 3, with the removed parts indicated in lighter grey. In comparison with the product data model in Figure 1, the difference is that data elements $C$, $G$, $H$ and $I$ are removed. Aside from this, the operation (1) producing data element $A$ is altered in that it now only uses data element $B$.

As algorithm Alpha represents each data element from the product data model with a transition in the process model, the removal of the data elements results in the removal of the corresponding transitions from the process model. Similarly, the transitions in the process model corresponding to the operations producing the data elements are removed as well.

As indicated before, the operation producing data element $A$ is now altered: it only uses data element $B$ and no longer data element $C$. In the corresponding process model, this is visible through one less place needed for transition $A$ to be able to fire.

The process model corresponding to the product
The process model shows the same transitions as the model in Figure 2, with the exception of the transitions labelled C, G, H and I (because the corresponding data elements have been removed from the product data model) and the consequently “dangling sections” in the process model. Removing transitions from the process model is risky in terms of quality assurance, but in this paper we assume the right parts are deleted.

The described example presumes that algorithm Alpha is used to generate the process model. In the next case the remaining two change primitives that have not been discussed, are applied to the product data model. The product data model from Figure 3 is used as the starting model and now algorithm Golf is used to generate the process model.

Algorithm Golf creates places in the process model as representations of the collections of all data elements that have been determined at those places. Starting from the start-place transitions are added, leading to new places representing the available data elements. Before adding a new place, it is checked whether an existing place already represents the available data elements and if so, this is used as output place of that transition.

This algorithm has a big advantage in the fact that there is no need to explicitly choose any path that must be followed at any time during execution, offering much flexibility to process execution. At all times, all operations that should be available to be executed considering the collection of determined data elements, are executable. And the ones that are not enabled based on the available data elements are not executable.

As algorithm Golf represents all operations in the product data model with transitions in the corresponding process model, the addition of operations 3 and 4 results in the addition of transitions representing both operations. The transitions are added to the process model at all places where they should be available. This way, operation 4, should for example be available at the very start, but also after operations 5, 6, 7, 2 and 3. This way the process model can change drastically due to a small change in the product data model.
The resulting process model is depicted in Figure 7, with changes due to the additions coloured.

V. Conclusion
Making changes in a product data model has its effects on the corresponding process model (and rightly so). These effects, when sufficiently small, can be completely determined so that the process model does not have to be regenerated from scratch again. This also brings insight in the construction of product data models and what the corresponding process model would look like.

These findings are most useful for big product data models, or models that change periodically, so the time needed for regenerating the process model can be saved. As it is no longer needed to completely regenerate process models after small updates to the product data model, the method of PBD becomes more efficient to be applied by companies. Using this knowledge, it is possible to develop systems in which product data model and process model(s) coexist and can be developed dynamically.

Future work includes researching the cases where PBD is applied to find out which additional information in the product data model could be useful to generate improved designs.

VI. References