Decision Model and Notation for Describing Variability in Business Process Product Lines

Andreas Speck, Aljoscha Jagenow and Melanie Windrich
Institute of Computer Science, Christian-Albrechts-University, Kiel, Germany

Keywords: Decision Model and Notation, Modeling Variability, Software Product Lines for Business Process Models.

Abstract: Product line techniques have proven being a successful concept of software reuse. Hence, product line techniques have been applied for business process models. Although there are well established models for describing business processes such as BPMN or ARIS there is a lack of describing the variability in business processes.
In the paper, we propose to apply *Decision Model and Notation* (DMN) for modeling process variability. DMN has the advantage that it is a known concept and well understood. DMN supports the decision techniques by defining the variability in business processes and with the rules defining the variability.

1 INTRODUCTION

Software product lines support the reuse of software components on a large scale. Software systems are considered as set of reusable assets with specific relations among them. The configuration of the assets is variable. The variability of assets in software product lines is an important issue in modeling. This requires a comprehensive notation. (Svahnberg et al., 2001) presents aspects of features and their modeling.

Based on the successful early approach, the combination of software product lines and business processes was discussed. Business processes are key elements in defining and describing the sequences of activities / workflows mainly in a commercial or administration context. Such software systems like enterprise resource planning (ERP), e-commerce or administrative systems are the organizational backbone of most businesses or administrative units. Business process models are used to define the behavior of such systems (dos Santos Rocha et al., 2015). Due to the high importance of business process model systems it is very desirable to find techniques making such systems more reusable like the product line concepts.

A summary and survey about approaches combining business process models and product lines may be found in (dos Santos Rocha and Fantinato, 2013). A conclusion of this review is that the approaches of applying software product line techniques to business process models is very promising. However, there is further need for improvement. A specific weak point is the modeling of the variability in business process models. This need of improved modeling of variability is pointed out in more recent surveys like (La Rosa et al., 2017) and (Pol'la et al., 2020). This demands a modeling concept for variability in business processes.

In the paper, we focus on modeling variability in business process product lines. We propose to use already existing and established model types for variability modeling. An analogous approach proven already as successful in product line modeling in which UML models are using UML stereotypes (La Rosa et al., 2017). First approaches introducing stereotypes are for instance (Clauß, 2001) or (Speck et al., 2002). In this paper, we propose to use BPMN (Business Process Model and Notation) as a basic notation for describing the processes which are the invariant parts of the product lines. This is an attempt taken by a number of approaches applying product line techniques to business processes (La Rosa et al., 2017; Pol'la et al., 2020).

For describing variability in business process models we propose to look on *Decision Model and Notation* (DMN). As a tool, we use the CAMUNDA modeler. However, our approach is independent from a specific tool.

In this paper, we first introduce the background of business process models, product line modeling and DMN. Then we show how business process variability may be modeled by DMN which is followed by an example in the domain of e-commerce systems which

Speck, A., Jagenow, A. and Windrich, M.
Decision Model and Notation for Describing Variability in Business Process Product Lines.
DOI: 10.5220/0010497204450452
In *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021)*, pages 445-452
ISBN: 978-989-758-508-1

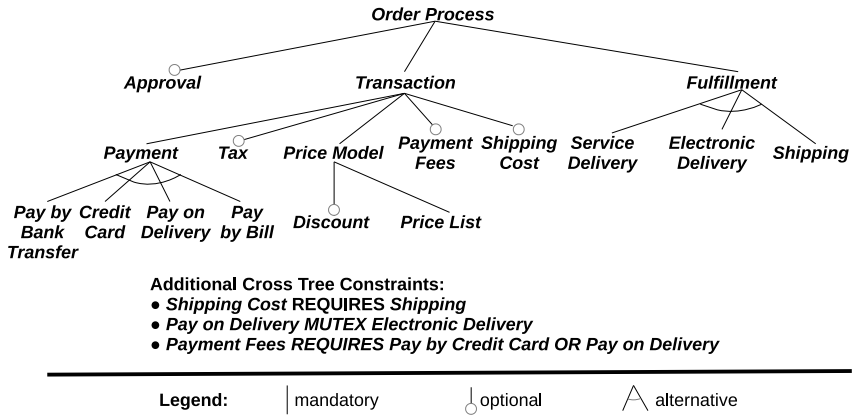

Figure 1: FODA Product Line Model for an *Order Process.*

demonstrates how to apply this approach.

2 BASIC CONCEPTS OF PRODUCT LINES AND BUSINESS PROCESS MODELS

At first, we have a look how variability is modeled in product line systems. The we present the elements of the Business Process and Model Notation (BPMN) used in the paper and introduce then the Decision Model and Notation (DMN).

2.1 Variability in Product Lines

The Feature-Oriented Domain Analysis (FODA) (Kang et al., 1990) is a valid model for describing dependencies between features and feature variability. An example of a FODA model representing an order process in an e-commerce system is depicted in Figure 1. FODA models arrange the features in tree structure (in our example *Approval*, *Transaction* and *Fulfillment* are child features of the *Order Process*). There are three different types of dependencies between the parent feature and its child features (or sub-features):

- The *mandatory* dependency requires a child feature.
- A child feature may be *optional.*
- *Alternative* dependencies requires the selection of one of the child features.

However, systems may rarely be modeled in a pure tree structure. In most cases the systems are organized in a net structure. The *cross tree constraints* (*REQUIRES* and *MUTEX*) of FODA express relations between the features in different branches. Two of these cross tree constrains in the example are:

- *shipping cost* requires *shipping* and
- *pay on delivery* and *electronic delivery* are mutually excluded.

2.2 Business Process and Model Notation

The Business Process Model and Notation (BPMN) supports the graphical modeling of business processes. The main elements comprise tasks (representing the basic functionality of the process), workflow objects (flow objects such as different gateway types or events or connection objects representing the sequence flow). Pools and swim lanes organize and identify major participants. Further artifacts are data objects or annotations. A pure example of a BPMN diagram with some important BPMN elements is shown in Figure 2 (left side). The tool used to model this diagram is Camunda. (BPMN, 2021a) is a Camunda BPMN documentation.

The BPMN diagram (on the left side of Figure 2) shows a business process model which may serve as a base for product lines. The basic mechanisms to realize variability in process models are the attachment or detachment of sequences to the process models (Sinnhofer et al., 2015; La Rosa et al., 2017). This means that the paths in the process model may be extended or shortened. BPMN supports the modeling of variability. Branching operators of the gateways such as *parallel* and *exclusive* may be sufficient for describing options and alternatives in a process model which are important for product lines concepts. However, purely using these operators will hamper the ease of understanding business process product line model and will lead to confusion. The question is now how to express the variability beyond the pure usage of gateways as branching operators. The description of variability need to be separated from the process

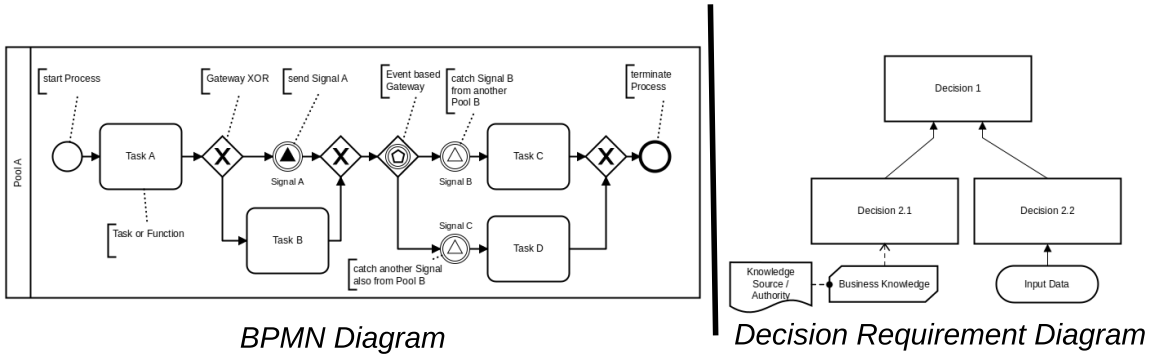

Figure 2: Exemplary BPMN process model (left) and DMN decision requirement diagram on the right.

models.

2.3 Decision Model and Notation

Decision Model and Notation (DMN) (OMG, 2021; BPMN, 2021b) supports modeling variabilities and decisions to be taken. There are diagrams, tables and and the expression language (Friendly Enough Expression Language, FEEL). The decision diagram or to be more precise decision requirements diagrams is at the right side of Figure 2:

- *Decisions*: several inputs (like input data or the knowledge) determine an output of the decision. Tables may be used to describe the decision logic.
- *Input data*: Information necessary determining a decision.
- *Knowledge Model*: decision logic which may be reused. The decisions based on the same knowledge model may lead to different results depending on inputs or sub-decisions.
- *Knowledge Source*: Authorities, committees, regulations or policies which influence the knowledge model.

The *decisions* may be hierarchical. The dependencies are represented by the directed edges.

The **decision tables** describe the logic of the decision. The rules are expressed by the rows. The elements of the rules (input and output) are kept in the columns. The input and output within a rule are connected. The *decision tables* are attached to the *decisions* in the DRDs (decision requirements diagram). Figure 5 in the application example (Section 4) shows a decision table. The decisions are attached to the business process models and the decisions represent the behavior at the gateways.

In modeling the knowledge of the decisions is separated in the DMN and not kept directly in the business process models.

3 VARIABILITY DEFINITION BY DECISION MANAGEMENT NOTATION

There are three elements for describing the variability in business process models by DMN: decision requirement diagrams (DRD), decision tables and the variability in business process models itself.

The decisions in the DRD (introduced in previous Section 2.3) may describe the variability in the process. Like the FODA models it is suitable for arranging the decision requirement diagrams in a tree structure. This supports a hierarchical structure of variants (or decisions) in which decisions may be composed to form underlaying sub-decisions. In contrast to FODA the DRD do not support the differentiation of types of dependencies between parent and child features (or decision and sub-decision). This may be solved by the decision table. The DRD gives hints about the motivation for the decisions:

- *Knowledge models* represent the decision logic.
- *Knowledge sources* are origins of *knowledge models*. These may be authorities or regulations.
- *Input data* may directly drive the decisions.

The cross-tree dependencies are *mutex* and *requires*. *Requires* requests a specific output. *Mutex* may be expressed by a combination of input and falsification of the output value. Similarly *alternative* and *optional* rules may be expressed by the combination of rows in the tables.

The activation of the rules is determined by the **Hit Policy:** The policy *Unique* (used in our example in Figure 5) means that only one rule matches. The policy *first* means that the first rule is applied. Examples for further rules are *any*, *rule* orders, *output* order or different types of rule collection. Examples of the usage of rules with decision tables may be found in documentations like (BPMN, 2021b; OMG, 2021).

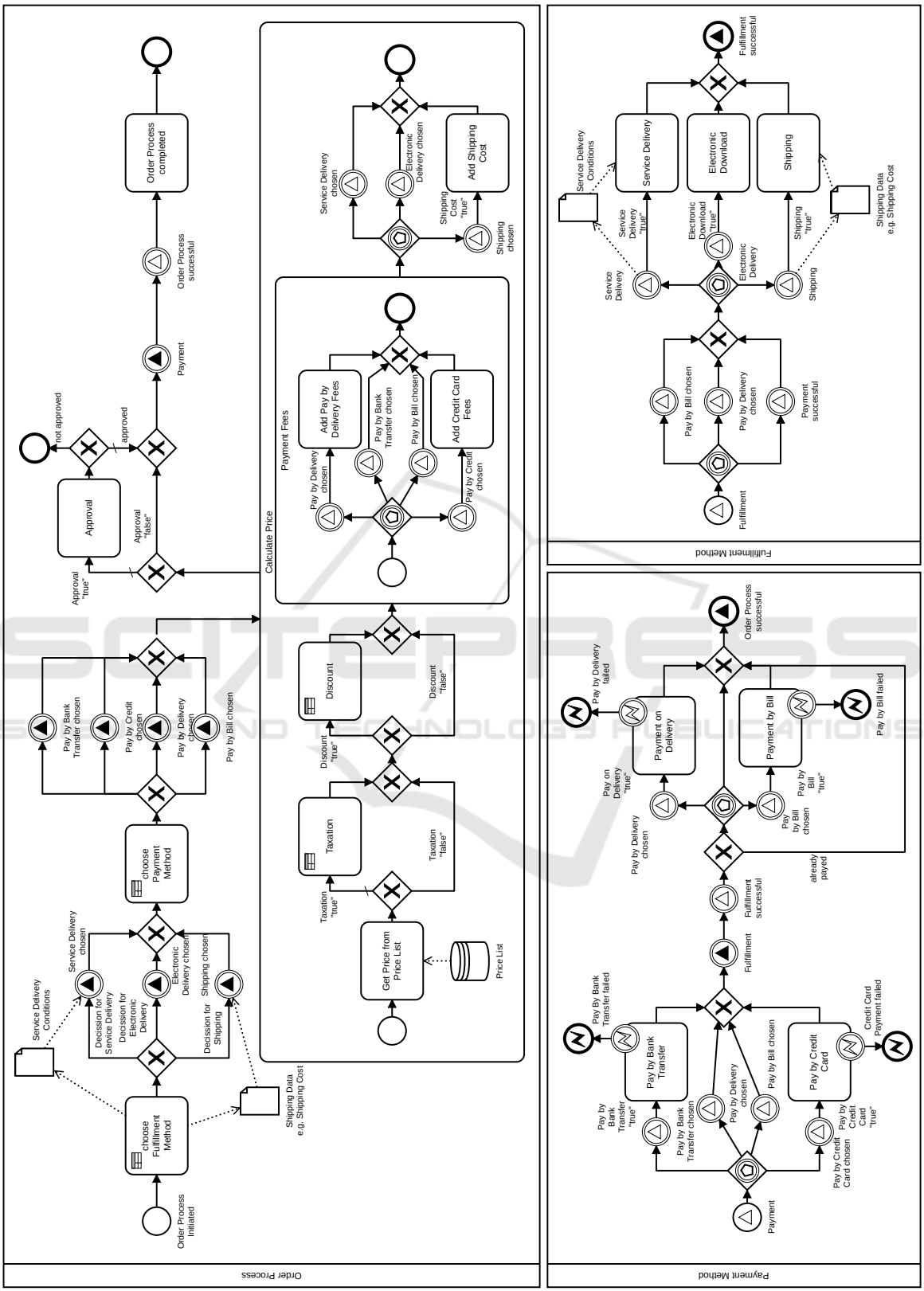

Figure 3: Example: BPMN Order Process Model.

4 EXAMPLE FOR APPLYING DMN

The order process in e-commerce systems serves as example in the paper. We already introduced this process in Section 2) as example for the FODA variability model (c.f. Figure 1). We now focus on the dynamic behavior modeled as a process. For comparison we use the similar functionality as in the FODA example.

The example consists of a decision requirements diagram (DRD), an exemplary decision table that are separated from the BPMN process model in which the decisions define the variability.

4.1 BPMN of the Order Process

The decisions presented in the subsequent subsections affect the BPMN process. In the process these decisions are taken and evaluated according to the rules in the decision tables.

The *Order Process* BPMN is shown in Figure 3. All branches with positive decisions are marked by ``true''.

The decision about the fulfillment method and the payment method are made at the beginning of the process. Each of these decisions result in events thrown representing the respective decisions.

The *Calculate Price* sub-process first looks up the concrete price in the *Price List*. This task is the only mandatory task in the entire sub-process. The tasks *Taxation* and *Discount* are based on the sub-decisions respectively and use the decision tables receiving input form their knowledge models. The *Payment Fees* sub-process and the potential adding of *Shipping Cost* are based on the events thrown by the tasks in which these methods are chosen.

All the previously mentioned tasks are part of the *Order Process* pool. The participants of this pool take care of the order process. However, there are other participants caring about the payment and the fulfillment. In e-commerce systems payment and fulfillment services are performed by specific systems called when required. Hence, these functionalities are modeled in separate pools. The communication between the processes is realized by *signal events* instead of *message events* for a better overview since the explicit connecting edges are not required (BPMN, 2021a).

The *Payment Method* process is divided in two parts in between which the *Fulfillment Method* is processed, since there is the possibility to pay before or after the fulfillment. All the tasks in the *Payment Method* and *Fulfillment Method* pools are depending on the events thrown at the beginning of the *Order Process*. The payment attempts may fail which is considered. However, the concrete effects of a failed payment are not modeled in this example. The decision table for the *Payment Method* is described in Section 4.3

4.2 DRD Example

The *Order Process Product Line* DRD in Figure 4[1] is considered as a decision which consists of the sub-decisions *Aproval*, *Calculate Price*, *Payment Method* and *Fulfillment Method*.

The (sub-)decision *Approval* is driven by the *Compliance Rules for Approval* as business knowledge which is imposed by the knowledge source of the *Compliance Rules Team* which may be the legal advisors of the company.

The (sub-)decision *Calculate Price* depends on the rules of the *Price List* knowledge model. The sub-decisions are *Taxation*, *Discount* , *Payment Fees* and *Shipping Cost*. *Taxation* is driven by the knowledge model *Taxation Rules* which are based on the source of *Legal Taxation Regulations*. An example of such a rule may be that for purchases from certain other countries no value-added tax has to be withheld. If a *Discount* is realized depends on the rule in the *Discount Decision Table*. Two different types of fees – *Credit Card Fees* and *Pay on Delivery Fees* – determine the *Payment Fees* decision. The *Credit Card Fees* and *Pay on Delivery Fees* also have an impact on the sub-decisions *Pay by Credit Card* or *Pay on Delivery* as part of the *Payment Method* (sub-)decision. The sub-division of *Shipping Cost* requires *Shipping*.

The (sub-)decision *Payment Method* uses one concrete kind of payment. The alternatives are the sub-decisions *Pay by Bank Transfer*, *Pay by Bank Credit Card*, *Pay on Delivery* and *Pay by Bill*. The concrete realization of one of these sub-decisions is driven by the input data *Choose Payment Method*. *Pay on Delivery* interacts with *Electronic Delivery*. This mutual exclusion is expressed explicitly in the decision table.

The sub-decisions *Service Delivery*, *Electronic Delivery* and *Shipping* form the *Fulfillment Method* (sub-)decision.

4.3 Exemplary Decision Table

We take the *Payment Method* decision table as depicted in Figure 5. This decision table is directly affil-

[1]The figure is not a screenshot of CAMUNDA but redrawn for better visibility. The DRD model of Camunda is rather voluminous.

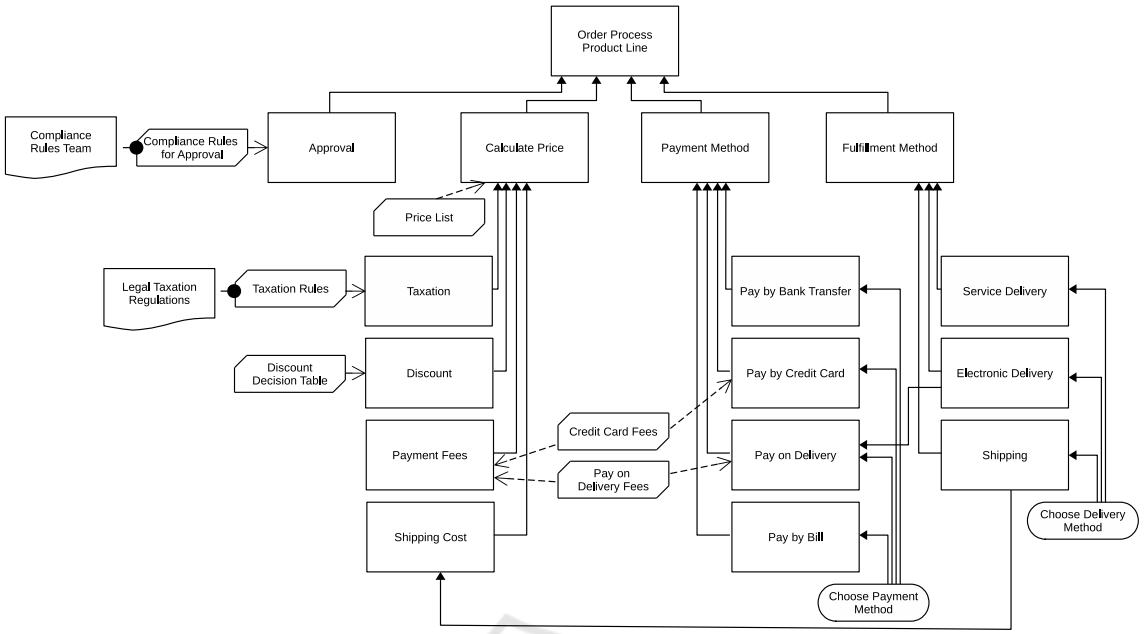

Figure 4: Decision Requirements Diagram (DRD) for the *Order Process Model.*

iated to the *Payment Method* decision and defines the decisions.

The table consists of four rules, each in a row. All rules depend on the input of the *Payment Method chosen*. These are the already presented *Pay by Bank Transfer*, *Pay by Credit Card*, *Pay on Delivery* and *Pay by Bill*. These will result in the corresponding events with similar names, *Payment Fees* when there are such fees and a determination of the *Fulfillment Method*. In case of *Pay on Delivery* there is only *Shipping* permitted.

Such decision tables may exist for the other (sub-) decisions like *Calculate Price* and *Fulfillment Method*. Since the rules in these tables will be in general similar, we do not present them.

5 RELATED WORK

In general the concept of reuse is to some extend supported by business process models, however, reuse is not the main focus. Reuse has been considered in reusing certain functions (Becker et al., 2000). Complex systems like product lines have not been considered.

With the orchestration of web services the reuse on a large scale emerged. Not only single services are to be reused, but also entire process sequences. An example for typical first approaches to apply product line concepts for business processes for web services is (Fantinato et al., 2010). The Business Process Execution Language (BPEL) is the modeling language for orchestration web service systems. As micro services may be considered as successors based on web services micro services are also subject of reuse concepts (Sinnhofer et al., 2020).

A concept of adapting business processes in variant-rich workflow-based systems is using the context in a certain state for determining which variant of the process is to be realized. The variants need to have a context-aware configuration. (Saidani et al., 2015) presents a concept of context awareness for adapting business process models. This context awareness is achieved by a context meta-model supported by an ontology. This adaptation of business process models is some kind of variability. Business objects are considered as long-living reusable assets while the processes are refined (Rulle and Siegeris, 2014).

A recent overview of the modifications in business process models in order to achieve variability is presented in (La Rosa et al., 2017). All approaches share the same basic concept of extending or shortening process paths in the same way as presented in this paper. Feature trees are also used by many approaches. However, these feature trees are used in the traditional way like FODA (c.f. Figure 1). Although BPMN as process model notation is used by many approaches, none of these previous approaches proposes to use decision model and notation (DMN).

A remarkable idea is supporting the distributed product line configuration by web-based systems. An example is ProductlinRE, an online management tool for requirements engineering of software product lines (Ghofrani and Fehlhaber, 2018). The on-

Payment Method | Hit Policy: Unique

	When	Then	And	And	
	Pament Method chosen (string)	thrown Event (string)	Payment Fees (double)	Fulfillment Method (string)	Annotations
1	"Pay by Bank Transfer"	"Pay by Bank Transfer"	0	"Service Delivery","Electronic Delivery", "Shipping"	Payment after Fulfillment
2	"Pay by Credit Card"	"Pay by Credit Card"	4	"Service Delivery","Electronic Delivery", "Shipping"	Payment before Fulfillment
3	"Payment by Delivery"	"Payment by Delivery"	12	"Shipping"	Payment before Fulfillment
4	"Payment by Bill"	"Payment by Bill"	0	"Service Delivery","Electronic Delivery", "Shipping"	Payment after Fulfillment
+	-				

Figure 5: Decision Table for *Payment Method*.

line system uses feature trees describing the variability. From different locations these feature tree models may be accessed and the variants may be selected in teamwork. Such distributed working would also be supported by using decision requirements diagrams (DRD) which also bear the option to be used in a distributed environment.

Some of the features in a decision requirements diagram (DRD) may be cross-cutting in the meaning of aspect-oriented programming (AOP) or separation of concerns (SoC) concepts. Applying aspect-oriented concepts allows to identify the cross-cutting features with dependencies across the entire feature model (Varela et al., 2011). These cross-cutting features may be considered as cross tree constraints. The advantage of clustering features as aspects and separating them in concerns is that these are more easy to handle. And entire groups of features may be activated or deactivated in one step. Such an approach may be supported decision requirements diagrams (DRD) and decision tables. However, a specific notation for expressing the aspects does not exist yet.

The basic understanding of the capabilities of Decision Management and Notation (DMN) may be found in the standard documentation (OMG, 2021) or basic tutorials like (BPMN, 2021b). (Taylor et al., 2013) explains the background for applying DMN. (Figl et al., 2018) presents further examples for applying DMN, e.g. hierarchies of decisions, the separation of decisions and process models or conventions. These are base of our work.

6 CONCLUSION AND FUTURE WORK

Software product lines are well established concepts supporting software reuse on a large scale. Business process models are a typical method for describing the behavior of commercial software systems like ERP (enterprise resource planing) systems, e-commerce or administrative systems. Due to the importance of both product lines and business process models it is an obvious choice to seek for a combination. As an extension of the well established business process model and notation (BPMN) the concept of decision model and notation (DMN) has emerged. DMN may help to solve the lack of appropriate modeling languages for variabilities (as desired by the product line concept) in business process models and to reduce the complexity of the model as proposed in (Taylor et al., 2013).

We propose applying DMN models and decision tables as means to define the variabilities in business process product lines. The variability in product lines are typically modeled in a tree structure. Decision requirements diagram (DRD) provide a similar tree structure in which the variability of the features which in DMN are called decisions (resulting in functions which realize the features) may be defined. In contrast to traditional feature tree structures DRD also provide the model elements representing the origin of the decision rules. The rules themselfes are defined in the decision tables. In rule tables both types of dependencies (mutual exclusion and require) between decisions may be expressed. The concrete location of the variants in the business process model are defined in the process model (in BPMN notation).

In our current approach, we do not consider yet another approach to express variants besides DMN: Case Management Model and Notation (CMMN). CMMN provides modeling elements which reduce the effort for describing the variants in a process model. The models provide a more condensed view of the variants. The open issue is if such simplified modeling will be accepted be the systems developers and if CMMN provides all the required techniques for modeling the variants.

Further future challenges are techniques for auto-

mated checking in the phase of the development of a business process product line. This comprises the development of the process model itself as well as the design of the variants. The consistency of the variants and correctness of order in the process models need to be assured. An automated support of this quality control would be desirable.

REFERENCES

Becker, J., Rosemann, M., and von Uthmann, C. (2000). Guidelines of business process modeling. In *Business Process Management, Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 30–49. Springer.

BPMN (2021a). BPMN Camunda, Documentation, https://camunda.com/bpmn/.

BPMN (2021b). Camunda, DMN Tutorial, https://camunda.com/dmn/.

Clauß, M. (2001). A proposal for uniform abstract modeling of feature interactions in UML. In *Proceedings of the ECOOP 2001 Workshop on Feature Interaction in Composed Systems (FICS 2001), Budapest, Hungary, June 18-22, 2001*, volume 2001-14 of *Technical Report*, pages 21–25. University of Karlsruhe, IPD.

dos Santos Rocha, R. and Fantinato, M. (2013). The use of software product lines for business process management: A systematic literature review. *Information and Software Technology*, 55(8):1355–1373.

dos Santos Rocha, R., Fantinato, M., Thom, L. H., and Eler, M. M. (2015). Dynamic product line for Business Process Management. *Business Process Management Journal*, 21(6):1224–1256.

Fantinato, M., de Souza Gimenes, I. M., and de Toledo, M. B. F. (2010). *Product Line in the Business Process Management Domain*, chapter 20, pages 497–530. Taylor and Francis Group, LLC.

Figl, K., Mendling, J., Tokdemir, G., and Vanthienen, J. (2018). What we know and what we do not know about DMN. *Enterprise Modeling and Information Systems Architectures EMISA Forum*, 13(2):24–26.

Ghofrani, J. and Fehlhaber, A. L. (2018). Productlinre: online management tool for requirements engineering of software product lines. In *Proceeedings of the 22nd International Systems and Software Product Line Conference - Volume 2, SPLC 2018, Gothenburg, Sweden, September 10-14, 2018*, pages 17–22. ACM.

Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-2, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA.

La Rosa, M., van der Aalst, W. M., Dumas, M., and Milani, F. P. (2017). Business Process Variability Modeling: A Survey Article. *ACM Computing Surveys*, 50(2):1–45.

OMG (2021). OMG, Decision Model and Notation (DMN), https://www.omg.org/dmn/.

Pol'la, M., Buccella, A., and Cechich, A. (2020). A. Analysis of variability models: a systematic literature review. *Software and Systems Modeling*, pages 1619–1374.

Rulle, A. and Siegeris, J. (2014). From a family of state-centric PAIS to a configurable and parameterized business process architecture. In *Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, volume 8659 of *Lecture Notes in Computer Science*, pages 333–348. Springer.

Saidani, O., Rolland, C., and Nurcan, S. (2015). Towards a generic context model for BPM. In *48th Hawaii International Conference on System Sciences, HICSS 2015, Kauai, Hawaii, USA, January 5-8, 2015*, pages 4120–4129. IEEE Computer Society.

Sinnhofer, A. D., Oberhauser, R., and Steger, C. (2020). From adaptive business processes to orchestrated microflows. In *Proceedings of Business Modeling and Software Design. BMSD 2020*, pages 152–168. Springer, LNCS.

Sinnhofer, A. D., Pühringer, P., and Kreiner, C. (2015). varbpm - a product line for creating business process model variants. In *Proceedings of the Fifth International Symposium on Business Modeling and Software Design - Volume 1: BMSD*, pages 184–191. INSTICC, SciTePress.

Speck, A., Clauß, M., and Franczyk, B. (2002). Concerns of variability in "bottom-up" product-lines. In *In: Proceedings of Second Workshop on Aspect-Oriented Software Development*, pages 19–24. GI, Gesellschaft für Informatik.

Svahnberg, M., van Gurp, J., and Jan, B. (2001). On the notion of variability in software product lines. Technical report, University of Karlskrona/Ronneby.

Taylor, J., Fish, A., Vanthienen, J., and Vincent, P. (2013). *Emerging Standards in Decision Modeling—an Introduction to Decision Model & Notation*, pages 133–146. WfMC, Workflow Management Coalition.

Varela, P., Araújo, J., Brito, I. S., and Moreira, A. (2011). Aspect-oriented analysis for software product lines requirements engineering. In *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC), March 21 - 24, 2011*, pages 667–674. ACM.