A Comparison of Decision Modeling Approaches in Product Lines

Klaus Schmid\textsuperscript{1} and Rick Rabiser\textsuperscript{2} and Paul Grünbacher\textsuperscript{3}

\textsuperscript{1}University of Hildesheim, Germany  
\texttt{schmid@sse.uni-hildesheim.de}

\textsuperscript{2}Christian Doppler Laboratory for Automated Software Engineering, JKU Linz, Austria  
\texttt{rabiser@ase.jku.at}

\textsuperscript{3}Johannes Kepler University Linz, Austria  
\texttt{paul.gruenbacher@jku.at}

Please cite this publication as follows:

The corresponding BibTeX-entry is:
\begin{verbatim}
@INPROCEEDINGS{SchmidRabiserGruenbacher11,
    author = {Klaus Schmid and Rick Rabiser and Paul Grünbacher},
    title = {A Comparison of Decision Modeling Approaches in Product Lines},
    booktitle = {Proceedings of the 5th International Workshop on Variability Modeling of Software-intensive Systems (VaMoS’11)},
    year = {2011},
    editor = {Patrick Heymans and Krzysztof Czarnecki and Ulrich W. Eisenecker},
    pages = {119--126},
    publisher = {ACM},
    doi = {10.1145/1944892.1944907}
}
\end{verbatim}

ACM, 2011. This is the authors version of the work. It is posted here by permission of the ACM for your personal use. Not for redistribution. The definitive version was published in Proceedings of the 5th International Workshop on Variability Modeling of Software-intensive Systems (VaMoS’11), DOI: 10.1145/1944892.1944907.
A Comparison of Decision Modeling Approaches in Product Lines

Klaus Schmid
Software Systems Engineering
University of Hildesheim
Marienburger Platz 22
31141 Hildesheim, Germany
schmid@sse.uni-hildesheim.de

Rick Rabiser
Christian Doppler Laboratory for
Automated Software Engineering
Johannes Kepler University
Altenberger Str. 69, 4040 Linz, Austria
rabiser@ase.jku.at

Paul Grünbacher
Systems Engineering and
Automation
Johannes Kepler University
Altenberger Str. 69, 4040 Linz, Austria
paul.gruenbacher@jku.at

ABSTRACT
It has been shown that product line engineering can significantly improve the productivity, quality and time-to-market of software development by leveraging extensive reuse. Variability models are currently the most advanced approach to define, document and manage the commonalities and variabilities of reusable artifacts such as software components, requirements, test cases, etc. These models provide the basis for automating the derivation of new products and are thus the key artifact to leverage the flexibility and adaptability of systems in a product line. Among the existing approaches to variability modeling feature modeling and decision modeling have gained most importance. A significant amount of research exists on comparing and analyzing different feature modeling approaches. However, despite their significant role in product line research and practical applications, only little effort has been devoted to compare and analyze decision modeling approaches. In order to address this shortcoming and to provide a basis for more structured research on decision modeling in the future, we present a comparative analysis of representative approaches. We identify their major modeling concepts and present an analysis of their commonalities and variabilities.

Categories and Subject Descriptors
D.2.1 [Requirements/Specifications]: Methodologies.
D.2.13 [Reusable Software]: Reuse Models.

General Terms
Management, Standardization, Languages.

Keywords
Product lines, variability modeling, decision models, comparison, survey.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1. INTRODUCTION AND MOTIVATION
Software product line engineering (PLE) has become an increasingly prominent approach to software development [8][30]. Results from industrial experience show that PLE can significantly improve software development productivity, quality and time-to-market by leveraging extensive reuse. Most companies not only develop individual products, but whole sets of products. One can thus expect that product line engineering will become a predominant development approach in the software industry in the long run. A core activity in product line engineering is variability management [8][19]. This process is important to understand the complexity and diversity of products in the domain of interest and it is central to achieve an efficient product derivation process.

This paper aims at providing a comparative analysis of decision modeling approaches, which represent an important subclass of variability modeling approaches. We define what we regard as a decision modeling approach and provide an overview of existing approaches. We hope that our analysis will contribute to improving the overall understanding of product line variability management. We expect that our analysis also provides a first step for comparing decision modeling in general with feature modeling in the future.

Most existing approaches in variability management can be classified (and classify themselves) as feature modeling and have been more or less directly derived from the work on feature-oriented domain analysis (FODA) by Kang et al. [16]. At this point different surveys of feature-oriented variability modeling exist, e.g., [4][5][7][28]. However, feature-oriented approaches are not the only way to deal with variability management. For instance, decision modeling approaches are a rather large family of approaches that exists nearly as long as feature-oriented modeling. Besides these two families of approaches other approaches exist belonging to neither of them. Examples are orthogonal variability management [19], the approach by Haugen et al. [14], or UML-based techniques [13].

The existing general surveys of variability modeling techniques [6][22] do not discuss the specific characteristics of decision modeling in detail. In this paper, we thus aim at filling this gap and provide a focused analysis of decision modeling approaches. Similar to the role of the FODA report [16] in the context of feature-based variability management most (if not all) existing decision modeling approaches have been influenced by the Synthesis method [29] which introduced the basic idea of...
decision models in 1993. In this paper we provide an overview of the diversity of approaches that have been developed in this category in the last 17 years.

The remainder of this paper is structured as follows: we will provide an overview of key decision modeling approaches in Section 2. We will then analyze their commonalities and variabilities in Section 3 by discussing their differences with respect to semantics. We conclude the paper with a discussion of our most important findings and provide an outlook on future work.

2. AN OVERVIEW OF DECISION MODELING APPROACHES

We restrict our overview to approaches that make an explicit reference to decision modeling and variability management in PLE. Thus, we take a loose definition of decision modeling by accepting any approach that self-qualifies as decision modeling, relates to software variability management and captures the kind of decisions that need to be made in order to arrive at a specific product in the product line. Besides this no further restriction was made.

We further decided to perform an in-depth analysis of some key approaches, instead of a very broad survey. We selected five approaches that to the best of our knowledge provide a broad overview of the characteristics of decision modeling. In our analysis we ignored the existence of variations over time in the different approaches. There are further approaches (e.g., [18][32][33]), which build on or are strongly related to the five selected approaches and which we did exclude, as we had the impression that including them would not contribute significantly to our comparison. Thus, we assume that with the named exceptions our comparison is ‘complete’. The five approaches we analyze are: Synthesis [29], Schmid and John [25], KobrA [1], DOPLER [10] and VManage [12].

For each of the approaches we give a brief overview of its characteristics and underlying concepts. Of course, a detailed description is beyond the scope of this paper. Thus, we also provide the relevant citations for the interested reader.

2.1 Synthesis

The earliest documented approach to decision modeling was defined as part of the Synthesis method [29]. Synthesis is an early comprehensive reuse process model, which even supports different modes of reuse like an opportunistic and a leveraged variant. The decision modeling approach was thus only one of the innovations of the approach. Most – if not all – approaches to decision modeling can ultimately trace their heritage back to Synthesis. The approach was developed by the Software Productivity Consortium for use in industry. However, it cannot be clearly determined from literature in which case studies this approach was used.

In Synthesis a decision model is defined [29] as “… a set of decisions that are adequate to distinguish among the members of an application engineering product family and to guide adaptation of application engineering work products”. Already in this early definition we see the strong focus on supporting product derivation as opposed to describing the domain as is typical for many feature modeling approaches. In Synthesis a decision model consists of three components: decision specifications, decision groups, and decision constraints. A decision specification contains a mnemonic for further reference, the possible resolutions, and a question to explain the decision to the application engineer. Decision groups cluster semantically related decisions. Decision constraints can be of two types: they can describe under which circumstances a decision group may occur; and they may constrain possible selections. Unfortunately, no precise description of the modeling approach is provided and the information can only be deduced from the examples (e.g., that constraints are described as free text). It is only defined that the decision model can be represented either as a list of questions or in a tabular format.

2.2 Schmid and John

The approach by Schmid and John [25] builds on Synthesis [29], but extends it in several ways and provides some formalization. Some of these extensions where inspired from industrial experience, e.g., from [23]. Based on this work some other applications were performed, although sometimes with further adaptations like in [26].

The approach consists of five components [25]: a decision model, a mechanism to describe interactions among different decisions, a mechanism to describe the relations among the artifacts and the decisions, a common (maximal) set of selector types (like optional or alternative), and a mapping of the selector types to a specific notation to express the variation points in the artifacts.

The decision model is defined based on Synthesis and therefore aims at supporting application engineering. There are several extensions relative to Synthesis. For example, binding times are an explicit part of the model (they are mentioned in Synthesis, but not a part of the decision model). Furthermore, multiplicity is added which allows selecting subsets of possible resolutions. The approach also clearly separates constraints on the presence of decisions and on the values of decisions. The work does not define a specific notation but rather emphasizes the separation between modeling concepts and representation. Relations are explicitly defined, both among decisions as well as between decisions and artifacts. These include set-based relations. In the approach artifact-decision relations are embedded in the artifacts and directed from the artifact to the decision model. This requires an extension of the target notation of the artifacts. This is explicitly supported by abstract selector types that are mapped to a target notation.

2.3 KobrA

The KobrA method supports the component-based development of information systems. The approach is relevant to our survey as it also addresses product line engineering and uses a decision modeling approach. The main reference is [1], however, a strongly related decision modeling approach is described in [18]. Here we will focus on the description given in [1]. The product line aspects of this work were also detailed in [2]. The decision model in the KobrA method is described using a tabular notation. The approach differentiates between decision models that describe simple decisions and decision models that describe high-level decisions. In addition there exist decision resolution models which gather the specific selections for a product instance.

The decision models for high-level and for simple decisions vary slightly. Both have an identifier and are accompanied by “a
textual, domain-related question that represents the decision to be made" together with the “set of possible answers to that question” [1]. For simple decisions references are given to the affected assets and variation points. Also, for each possible answer (decision resolution) a description of the effects on assets is provided. On the other hand, for high-level decisions references are given to the other decisions affected by resolving a decision in the model and for each possible answer a description of the effect is described.

The KobrA method relies on the assumptions that models are configured and that they are described using the UML. Product line assets thus are described in UML diagrams which contain the combination of the elements for the various variants. The elements relevant to only some variants are annotated by a variant stereotype. The effect of resolving a decision is given by describing the removal of the corresponding elements for simple decisions. The resolution of higher-level decisions is given by setting values for lower-level (or simple) decisions. The decision models thus form a hierarchy. However, it seems that the impact among decisions can only be described as setting a value along the hierarchy. For instance, the exclusion of a specific answer from a set of answers, e.g., using constraints, does not seem to be possible. Also, a formal model of the approach is lacking. In [18] we find a more generalized approach that also explicitly mentions exclusion constraints. A characteristic of the KobrA approach is that the decision model explicitly captures all impacts of decision resolutions on artifact models. This is exactly opposite to the two approaches discussed before [25][29].

2.4 DOPLER

DOPLER [10] has been developed since 2006 at the Christian Doppler Laboratory for Automated Software Engineering (JKU Linz) in collaboration with Siemens VAI and Siemens CT to support modeling the variability of industrial product lines with an emphasis on automating the derivation of customer-specific products. DOPLER uses decision models to define problem space variability, asset models to define reusable elements of different types and their dependencies, as well as mappings among assets and decisions (from solution space to problem space). Besides the component-based automation software product line of Siemens VAI [9], the approach has also been used in the domains of enterprise resource planning, service-oriented systems, and Eclipse-based software engineering tools [10][21].

In contrast to KobrA [1] but similar to Synthesis [29] and Schmid and John [25], assets in DOPLER define dependencies to decision while decisions are “unaware” of the assets in the solution space. Decisions have a unique name and a question that is asked to a user during product derivation. Answering a question sets the value of a decision. Possible answers depend on the type of the decision (Boolean, enumeration, string, or number). The range of allowed values can be further restricted by validity conditions. Decisions can depend on each other hierarchically (if a decision needs to be taken before another decision becomes "visible") or logically (if taking a decision changes the value of another decision). Assets can depend on each other functionally (e.g., if one component requires or excludes another component) or structurally (e.g., if a component is part of a sub-system). DOPLER allows defining asset types at arbitrary granularity and with user-defined attributes and dependencies based on a given set of basic types. Users can create domain-specific meta-models to define the types of assets, their attributes, and dependencies for a particular domain. Dependencies between assets and decisions are explicitly modeled via inclusion conditions that define for an asset when it will be part of a derived product. DOPLER is supported by an Eclipse-based tool suite [10][21] and comprises approaches and tools for variability management [9], model consistency and evolution [31][9], and product derivation [20].

2.5 VManage

VManage has been developed at the European Software Engineering Institute Spain and IKV++ Technologies AG Germany [12] and is based on a decision modeling process described by Mansell and Sellier in [17]. Similar to Weiss and Lai [32], the authors define a decision model as “...a document defining the decisions that must be made to specify a member of a domain” [17]. A decision model is described in XML and represents all possible user requirements defined during the domain analysis and the set of rules and constraints associated with them. Each decision is defined with a set containing the following information [17]: a unique identifier, a description, a type expressing the possible values supported by the decision, a default value, validity criteria to make or not to make the decision, as well as two kinds of decision dependencies: (i) The value of a decision can impact on the range or value of another decision and (ii) the value of a decision can impact whether another decision should be made.

VManage explicitly differentiates between restricted and unrestricted decisions. Unrestricted decisions do not support constraints other than their data type restrictions. Restricted decisions have other restriction specifications making their specification and their implementation within XML more complex. Furthermore, VManage supports collections of decisions, which are instances of a decision or a set of decisions. For example, when two instances of a particular component are required, the process of taking decisions to configure these components has to be repeated for each required component, i.e., decisions need to be duplicated. The number of instances can be specified as a collection restriction. VManage is supported by three tools: V-Define supports creating decision models; V-Implement allows defining variation points in component descriptions (files); and V-Resolve supports making decisions in product derivation by setting values based on decision type and validity criteria.

3. AN ANALYSIS OF THE COMMONALITIES AND VARIABILITIES OF DECISION MODELING APPROACHES

We have analyzed the five aforementioned approaches by carefully reviewing representative publications and information about the supporting tools and discussing commonalities and differences in several meetings. In our analysis of decision modeling in product lines we present a discussion of the commonalities of the five approaches described in Section 2 and their most essential variabilities. We will structure the discussion along the following categories: model structure, decision data types, decision dependency management, handling of artifacts, and product derivation. Finally we discuss various other aspects that cannot be assigned to the earlier categories, but are of interest nevertheless.
A key question for any systematic comparison is how to derive the criteria. In this study we combined a bottom-up with a top-down approach. Bottom-up we used other surveys on variability modeling and also identified several aspects we agreed upon as characterizing decision modeling. Bottom-up we identified various aspects in the method descriptions and then decided to compare these across the approaches. A filter across all criteria was whether the method descriptions contained enough information to decide if an aspect is really covered. The remaining criteria were then clustered into the categories used below (and shown in Table 1).

We explicitly do not aim to provide a ranking of approaches. We only focus on characterizing, not on evaluating. This decision was made, as we strongly expect that what is better or worse depends on the usage context.

### 3.1 Model Structure

We discovered that all approaches share some commonalities with regard to the basic model structure (see Figure 1). All approaches define an element decision representing a set of choices available at a certain point in time and a decision model containing a set of decisions. Further, some attributes are defined for a decision. The specific set of attributes varies among the modeling approaches, however, all have a unique decision name or id and all define some attribute describing the decision in a manner understandable by an end-user, e.g., a question (cf. Figure 1; called description in some approaches). The data type of a decision defines the set of choices possible for it. All approaches support at least the data types Boolean and Enumeration.

![Figure 1. UML class diagram of common decision model elements.](image)

All approaches allow creating dependencies among decisions. Value constraints for defining interdependencies among values are supported by most approaches. However, no two approaches use exactly the same concept for value constraints and more generally for dependency management. Some approaches use constraints (similar to feature modeling approaches) and allow defining Boolean expressions to be valid for the whole decision model that can be validated by constraint solving engines like satisfiability solvers. Other approaches model dependencies with conditions or rules that are executed or checked when a decision is taken.

Figure 1 shows that different approaches support different decision data types (Section 3.2) and manage constraints and dependencies differently (Section 3.3). The relation of decisions with the artifacts of a product line varies among the approaches (Section 3.4). Furthermore, there are differences regarding product derivation support (Section 3.5) and additional modeling concepts (Section 3.6). We provide an overview of the main characteristics of the different approaches in Table 1.

### 3.2 Decision Data Types

The data type defines the set of choices possible for a decision. An assessment of the various supported types of decisions is rather difficult, as several approaches do not provide a clear or even formal definition of the possible types.

All approaches seem to support at least enumerations and in particular Boolean decisions. This is also the case in feature-oriented modeling approaches. However, most decision modeling approaches support a significantly broader range of data types (cf. Table 1). The only approach which seems rather restricted appears to be KobrA [1]. We say appears to, as KobrA does not provide an explicit definition of the decision data types, but from the structure of its models, it seems that only enumerations (including Boolean) are possible.

Also for Synthesis we had to derive the form of possible decision data types indirectly from the examples and descriptions given. The approach takes a rather liberal approach and examples are given for many data types such as Strings, Integers, Enumerations, or Date and Time [29]. Moreover, the approach provides capabilities for lists and even structured data types (e.g., comparable to structs in programming languages like C). It is unclear whether floating point numbers are considered, but certainly lists of other (composed) types are possible.

The approach by Schmid and John provides a very simple answer to the data type definition problem: it says "This can be basically any of the typical data types used in programming languages" [25]. However, they then take an somewhat unusual approach by including set-valued data types and treating classical data types as a special case of set-types.

Only KobrA has a rather restricted range of decisions (enumerations). It seems all other approaches at least support the typical discrete data types, while only Schmid and John, DOPLER and VManage support floating point numbers. Synthesis and VManage additionally support the types date and time. Sub-ranges are additionally supported by Synthesis, DOPLER and VManage. DOPLER supports defining Validity Conditions which restrict the value range determined by the basic decision data type (e.g., to only allow values between 1 and 4 for a number decision). Validity conditions can be arbitrarily complex Boolean expressions. This can also be seen as a special form of constraint where no other variable is involved.

### 3.3 Decision Dependency Management

The decision modeling approaches differ with respect to the form and extent of support for dependency management (cf. Table 1). The first major difference is how decision dependencies are modeled by the various approaches. Some use only text, while others use rules, logic, etc.
For Synthesis it is not clear whether it was expected to be automated. The main focus was to provide a process model and method guide. As a consequence the constraints in [29] are described as free text. Thus, arbitrary constraints can be formulated. While sharing a lot of aspects with Synthesis, this aspect is fundamentally different in the approach by Schmid and John [25]. Here, the focus was explicitly on a formalization of the constraints. As a consequence, all dependencies are described as formulas. However, the paper does not go into details regarding any implementation, as this was considered outside the scope. On the other hand, there exist implementations like the one described in [27], which in turn do not support the full range of capabilities of the underlying approach.

Although KobrA focuses more on the process than on a tool environment, it provides a clear definition of its dependency management. It is, however, very simple: selecting values for a decision can lead other decision values to be set. Both VManage and DOPLER provide a formalized approach to dependency management, as they also provide tool support (i.e., the described dependency management corresponds to the tool implementation). While DOPLER relies on a rule-based description and implementation, VManage relies on a logic-based formulation with an event-based implementation mechanism.

We can further categorize the dependencies into three types: Value constraints are used for defining interdependencies among decision values leading to restrictions on the permissible values of a decision. Relevancy conditions formalize under which conditions decisions are included in the model. This is necessary as some decisions may be included only under certain conditions in the model (variants of the decision model). Relevancy conditions thus lead to conditional extensions of the variability model. Visibility conditions determine when values for certain decisions may be set by the user. Thus, opposed to the previous two, visibility conditions do not modify the model, but structure the model in a hierarchy and provide a means of user guidance.

From these types value constraints are supported by all approaches. This seems quite natural as interdependencies among values are quite common in variability management. Value constraints are directly related to feature dependencies in feature-oriented approaches. While all approaches support relations among individual decisions, Synthesis and VManage also support logical relations among members of decision groups. A peculiarity in the VManage approach is that it explicitly differentiates between restricted and unrestricted conditions. Unrestricted decisions do not support constraints other than their data type restrictions. Restricted decisions support additional concepts like logical relations. The concept of logical relations is similar to the validity conditions that DOPLER supports to define permissible values for individual decisions. However, in Synthesis and VManage these constraints are defined at a group level. This can imply in particular quantification. Note, however, that the approaches need to provide grouping capabilities in the first place for this to work.

The language (and thus the expressiveness) provided for supporting value constraints is also very different among the approaches. The simplest approach is used in KobrA as decisions can only lead to setting other decisions while more complex constraints seem not to be possible. The other approaches allow rather complex formulas over decisions as a basis for determining the restrictions. The approaches by Schmid and John and DOPLER are peculiar insofar as they explicitly support set-based operators as well as relations. Synthesis in turn allows full natural language. Only some approaches support relevancy conditions. These are Synthesis, Schmid and John, and VManage. They allow the same kind of expressiveness as they allow in value constraints. Visibility conditions specify when a particular decision becomes "relevant" to the user (before the condition becomes true, the user cannot make the decision). They are supported only by DOPLER. Of course relevancy conditions are similar to some extent. A decision can only become visible to the user once it is clarified that it is relevant. The main difference is that visibility conditions do not modify the underlying decision model, while relevancy conditions do.

With the exception of Synthesis all investigated approaches assume tool support for the automated evaluation of constraints. For Synthesis this is not exactly clear, but the reliance on pure text makes tool support basically infeasible. The approaches used, respectively foreseen, by the different decision modeling approaches for constraint evaluation differ significantly.

DOPLER supports defining decision effects, i.e., conditions of arbitrary complexity that are checked after a decision has been taken together with one or more actions to be executed depending on the conditions. Actions can for example be used to set values of decisions. Actions can also reset a decision to “not yet taken” or change the range of possible values of a decision. VManage is similar, as it relies on an event-based approach. The modification of a decision value raises an event that leads to a re-evaluation of the various constraints.

The KobrA approach in its original form is rather simplistic as a decision definition directly describes for all impacted decisions the modification of their values. Experimental tools for implementing this approach existed.

Finally, the approach by Schmid and John built on a pure constraint-logic description. The expectation was that this would be implemented by using a sophisticated reasoning framework like FaMa [5], but existing implementations of parts of this approach used much simpler technology [24][27].

### 3.4 Artifacts

This aspect is different from feature modeling, where approaches mostly focus on modeling variability, without considering the relationship to artifacts. Researchers have, however, been working on such mappings of feature models to artifacts (e.g., [15]). The compared decision modeling approaches address the relationship of decisions to the reusable artifacts in a product line in different ways (cf. Table 1).

In Synthesis, the approach by Schmid and John and in VManage, decisions are referenced from the artifacts. In Synthesis and the approach by Schmid and John arbitrary artifacts can refer to decisions. Synthesis provides different kinds of realizations (like text and diagram notations). The approach by Schmid and John provides a set of variability selectors and requires a mapping of these selectors to individual (artifact) notations. This approach allows the use of the method with arbitrary notations. VManage allows defining component descriptions (i.e., in a domain-specific notation) that directly refer to decision values in the decision model.
KobrA has been specifically defined for and restricted to UML. The impact of decisions on artifacts is specified as part of the decision model (which might lead to issues in terms of model evolution). The impact of a decision in KobrA is typically the removal of a model element from a UML model (i.e., negative variability; the domain model is designed as a superset).

DOPLER defines solution space artifacts explicitly in dedicated Asset Models that define the assets available in a product line. Examples of assets are software components, test cases, or documentation fragments. An asset model describes the solution space at the level of abstraction necessary for subsequent derivation of products. The flexibility of asset models is achieved by defining a domain-specific asset meta-model that addresses domain-specific concepts based on the generic core DOPLER meta-model. Artifact-Decision dependencies in DOPLER are specified using Inclusion Conditions that describe the context and situation when a particular asset is required in the desired product. Inclusion conditions can refer to several decision values.

Table 1. Support for modeling concepts in five decision modeling approaches (+ means the approach supports the concept).

<table>
<thead>
<tr>
<th>Modeling Concepts</th>
<th>Synthesis</th>
<th>Schmid &amp; John</th>
<th>DOPLER</th>
<th>VMage</th>
<th>KobrA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model Structure</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td><strong>Decision Data Types</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision Type</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Boolean</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>String</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Enumeration</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Integer</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Real</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Date/Time</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-ranges</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Set-based data types</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Constraints and Dependencies</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value Constraints</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Relevancy Conditions</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visibility Conditions</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Only Text-based (no formalization)</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expression-based (formalization)</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Automated evaluation of constraints</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td><strong>Artifacts</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arbitrary Artifacts</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Only Artifacts of a specific type (e.g., UML)</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relation from Artifact (model) to Decision model</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Relation from Decision model to Artifact</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Product Derivation Information</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision Rationale and/or Description</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Tasks, Roles, Users</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meta-Variability (e.g., binding times)</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision Group</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision Collections</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Fragments (distributed work)</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tool Support</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.5 Product Derivation Information
We observed that product derivation support is addressed in all decision modeling approaches. For instance, they allow defining additional information for subsequent product derivation in decision models. In particular, on the modeling level, they support humans in configuring specific product line members by a number of means.

For instance, Synthesis allows specifying a rationale for a decision (why it should be taken). Product derivation in Synthesis means going through a list of questions. Schmid and John as well as VManage allow defining arbitrary descriptions of decisions to be used in product derivation to communicate decisions to people responsible for configuration. KobrA also uses questions as a means to clarify decisions to the user. DOPLER allows defining questions and arbitrary descriptions of decisions. DOPLER also provides further means to support, control and manage derivation like guidance (to explain decisions), tasks (to group and prioritize decisions), roles (to be assigned to tasks), and users (to be assigned to one or more roles). These are, however, not part of the variability model but part of a separate model called the derivation model [20].

A specific form of support for product derivation in the approach by Schmid and John is the capability to define binding times on a per-product basis. In particular, the approach allows defining several binding times per decision. The concept behind multiple binding times in this approach is that different forms of derivation for the same underlying functionality may lead to different variants where the decision resolution happens at different points in time. This was later refined by introducing the concept of meta-variability [24] to address the variation of derivation processes.

3.6 Other
The different approaches provide additional concepts to support variability management with decision models (cf. Table 1). We discuss some we found most interesting. In particular, we focus on concepts for structuring large decision models and tool support.

One concept supported by several approaches is the grouping of decisions with the goal to structure large models. Synthesis, DOPLER and VManage support decision groups. A group has a name and defines its members, which are several individual decisions. Introducing a decision group only provides a logical clustering of the decisions. It does not change the underlying semantics of a decision model. The VManage approach supports a special variant of groups called collections, which allow instantiating decisions multiple times. DOPLER supports the decomposition and later integration of large decision models into multiple model fragments. A model fragment defines public and private decisions (and assets). For instance, a fragment can declare placeholder decisions that are replaced with public decisions from other fragments in a subsequent semi-automatic merge process [9]. Model fragments are merged into a single integrated model for product derivation.

Tool support exists for several of these approaches, but we must clearly differentiate between the methods as described in this paper vs. the corresponding tool support. For VManage and DOPLER this is actually very easy as their mapping is basically one to one. So actually, the status of the method described here also corresponds to the latest tool implementation. It is more difficult for KobrA and the approach by Schmid and John. In both cases detailed descriptions [1][25], are available which provide the basis for this discussion. There also exist tools, but the tools do not support for the full range of the methodology or extend it in certain ways. For example the approach by Schmid and John is partially implemented in [27] and [24], while aspects of a tool implementation for KobrA can be found in [11].

4. CONCLUSIONS AND FUTURE WORK
Our analysis of key decision modeling approaches reveals significant differences in terms of modeling concepts, degree of formalization, and tool support. There is a small core of common concepts but we found variability even in the core concepts. For instance, while all approaches explicitly support relating decisions to artifacts, they all do it in different ways.

The approaches also differ regarding their degree of formalization. Synthesis and KobrA are defined rather informally and provide only little guidance. Other approaches such as Schmid and John, VManage, or DOPLER provide a significantly stronger formalization. We identified certain areas of decision modeling concepts like constraints and dependencies, artifact relationships, and so forth. In all these areas we identified (sometimes very significant) variations.

We believe that our work can lay some ground for further research activities. For instance, as pointed out in [3] decision modeling languages share many characteristics with feature modeling. We hope that this paper can thus provide an initial step towards a better understanding of the similarities and differences of these two families of approaches.

5. ACKNOWLEDGMENTS
This work has been partially supported by the Christian Doppler Forschungsgesellschaft, Austria and Siemens VAI Metals Technologies.

6. REFERENCES