It's Your Loss: Classifying Information Loss During Variability Model Roundtrip Transformations

Kevin Feichtinger
LIT CPS Lab
Johannes Kepler University Linz
Austria
kevin.feichtinger@jku.at

Chico Sundermann
University of Ulm
Germany
chico.sundermann@uni-ulm.de

Thomas Thüm
University of Ulm
Germany
thomas.thuem@uni-ulm.de

Rick Rabiser
CDL VaSiCS, LIT CPS Lab
Johannes Kepler University Linz
Austria
rick.rabiser@jku.at

ABSTRACT

In software product lines, variability models are used to explicitly capture commonalities and variability of a set of software systems. Many variability modeling approaches have been developed over a period of more than 30 years. Most of them are only described in academic papers, which makes it difficult to assess their properties and find the right approach for a specific use case. New approaches are developed regularly, adding to the ever-growing plethora of variability modeling approaches. Transforming variability models, i.e., of one type to another, would help to better understand and compare existing approaches and would also enable users to switch between approaches. Since variability modeling approaches differ especially in terms of scope and expressiveness, it is difficult to implement transformations without information loss. Thus, in this paper, we analyze concrete variability modeling approaches, present a mapping of key concepts between them, and identify and classify the information lost in one-way and roundtrip transformations. We evaluate their applicability by transforming different models of varying size and complexity using an existing implementation of transformations. We argue that our classification of information loss contributes to a better understanding of different variability modeling approaches, simplifies the comparability, and allows users to grasp the impact of transformations.

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**.

KEYWORDS

Software product lines, variability modeling, variability model transformations, information loss.

SPLC '22, September 12–16, 2022, Graz, Austria

ACM ISBN 978-1-4503-9443-7/22/09.
https://doi.org/10.1145/3546932.3546990

ACM Reference Format:
Kevin Feichtinger, Chico Sundermann, Thomas Thüm, and Rick Rabiser. 2022. It's Your Loss: Classifying Information Loss During Variability Model Roundtrip Transformations. In *26th ACM International Systems and Software Product Line Conference - Volume A (SPLC '22), September 12–16, 2022, Graz, Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3546932.3546990

1 INTRODUCTION

In Software Product Lines (SPLs), variability models are used to capture common and variable parts of a set of software(-intensive) systems [11, 21]. In the last 30 years, many variability modeling approaches have been developed [7, 11, 41], which, despite differences, can be categorized into different types. Most approaches [16] are of type feature modeling, building on the Feature-Oriented Domain Analysis (FODA) approach [34], and decision modeling, originating from the Synthesis method [15]. Other types include Orthogonal Variability Modeling (OVM) [40], UML-based variability modeling [29], or textual (domain-specific) languages [7]. Further, industry and open-source communities have developed approaches to capture variability [50, 54] for their use cases. Despite past and ongoing efforts to harmonize and standardize variability modeling, e.g., the Common Variability Language (CVL) [30], the Universal Variability Language (UVL) [51] or ISO-26558 [33], diverse approaches with different properties and purposes will remain.

Several researchers have compared existing variability modeling approaches [5, 14, 16, 22, 28, 45, 48] and have investigated their application in industry [12, 36]. Each variability modeling approach has its benefits and drawbacks, but has at least been shown to be useful in a specific domain or use case. Unfortunately, many approaches are only described in academic papers and even fewer come with (still available) tool support [5]. Thus, it remains challenging for researchers and practitioners, especially if they are new to the field, to understand the properties of different approaches. Instead of constantly developing new approaches [25], transforming variability models, i.e., of one type to another, can help to select the right approach for a specific use case, compare between and switch to a different approach without losing (already) invested modeling efforts, or utilize analysis techniques provided by a tool for a different approach [24].

Researchers have investigated variability model transformations to increase the interoperability among modeling tools [23, 42, 43]. We presented our transformation approach TRAVART [25], which enables the transformation between FeatureIDE feature models [37], DOPLER decision models [19] and OVM [40]. In recent work, we integrated UVL [51] as a pivot language in TRAVART. Transformation approaches must provide multiple capabilities [24]. For instance, transformations should *automatically* transform a model of type A into a model of type B (*one-way transformation*), or *automatically* transform the resulting model of type B back to a model of type A (*roundtrip transformation*). Further, transformations should *minimize information loss* to be adopted by researchers and industry [24]. Information easily gets lost during transformation and may the configuration space of a variability model changes, causing potentially invalid/broken products during configuration. Currently, lost information is often not investigated in detail, but rather explained by limited capabilities of the approaches and tools (importers and exporters) and justified and accepted during evaluation.

Investigating the information lost during transformations enables a better understanding of the different variability modeling approaches, improves their comparability, allows users to grasp the transformation impact, and supports the implementation of transformations and tool importers and exporters. For certain use cases, some information loss might be more acceptable than for others. Therefore, in this paper, we analyze the information lost when transforming variability models of different types into each other and define information loss classes to categorize the different types of losses. Specifically, we present a mapping between four concrete variability modeling approaches for one-way and roundtrip transformations and identify concepts, which cannot be transformed into another notation: (a) FeatureIDE feature models [37], (b) UVL feature models [51], (c) DOPLER decision models [19] and (d) OVM models [40]. Additionally, we discuss how certain losses can be addressed or mitigated. We evaluate the applicability of our information loss classes by analyzing their occurrence in transformed variability models of the four types of different size and complexity.

In the remainder of the paper, we first describe our research method in Section 2 and outline core concepts of the four approaches in Section 3. We then present the main contributions, i.e., the information loss classes in Section 4 and their occurrence among approaches in Section 5. We present evaluation results in Section 6 and discuss related work in Section 7. We conclude the paper and outline future work in Section 8.

2 RESEARCH METHOD

This study is is based on our knowledge of the SPL literature, research area and our experiences of using and developing variability modeling approaches and tools for over 15 years. All authors of this paper conducted multiple workshops (overall: 18 digital meetings with an average duration of 1,5 hours) to agree on the analyzed variability modeling approaches, defined mappings of concepts between them and identified information loss classes. We followed a mixed method empirical approach [26] with elements of a field study, using prominent variability modeling approaches as study objects, and a judgement study, with us authors as experts of the field. Specifically, our research method comprises four phases:

(1) Identify variability modeling approaches. Based on our previous experiences of using and implementing different variability modeling approaches and tools [25, 37], we agreed on FeatureIDE feature models [37], UVL models [51], DOPLER decision models [19], and OVM models [40] as the variability modeling approaches for our study. They cover prominent types of variability modeling approaches and have been widely discussed before [16, 41]. We aimed to investigate the lost information when transforming models of these approaches.

(2) Derive mapping tables between approaches. We explored key concepts, properties and data structures of the four variability modeling approaches [19, 37, 40, 51] to derive mapping tables between them. One of the authors created initial mapping tables, which the other authors reviewed and commented followed by open discussions. The same author creating the initial mapping tables gathered the feedback and incorporated it into the tables to refine the mappings. This process was repeated until all authors agreed on the mapping tables. We decided to use UVL [51] as a pivot language for the mapping tables presented in Section 5.

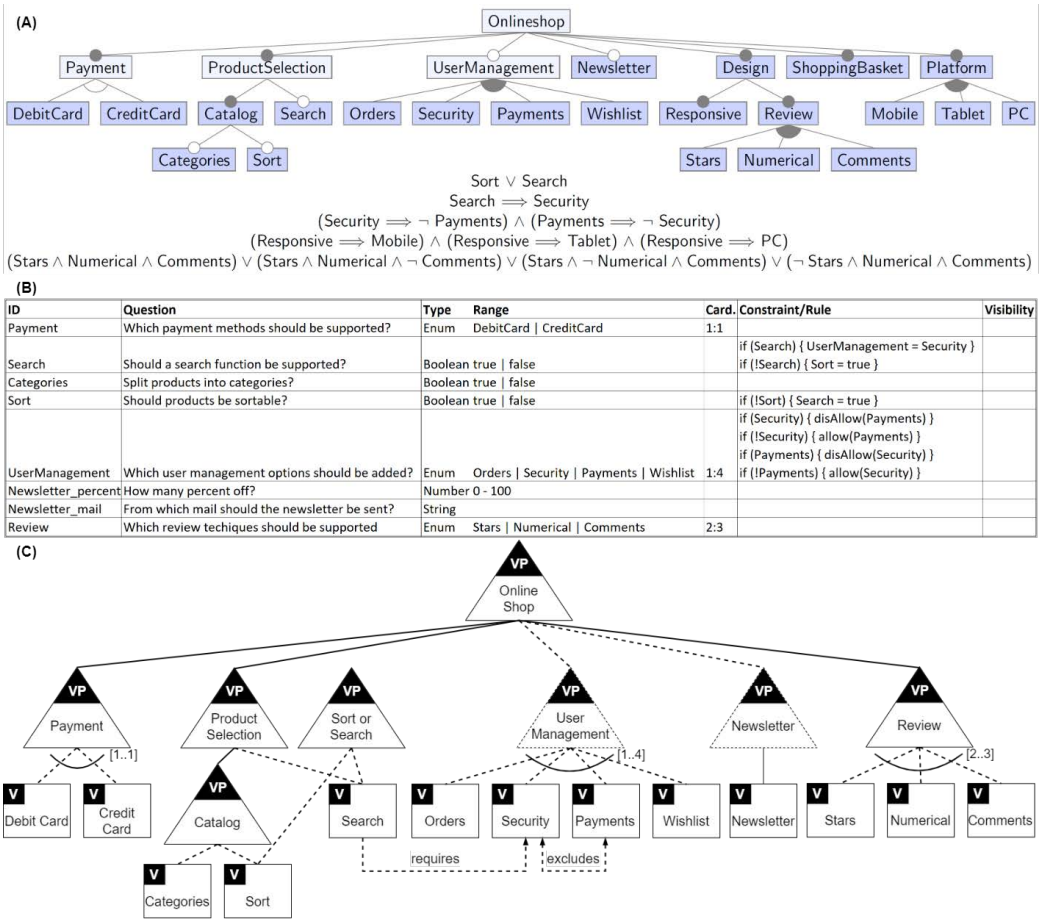
(3) Identify and categorize information loss. We built on published findings comparing the approaches [16, 22, 41, 43] and the mapping tables to identify the information lost when transforming between the four variability modeling approaches. Specifically, we discussed the transformation impact and the lost information in one-way and roundtrip transformed variability models. We defined, reviewed, and agreed on information loss classes based on patterns we observed. We present those information loss classes in Section 4.

(4) Evaluate the applicability of the information loss classes. We transformed multiple models of varying size (in terms of numbers of features, decisions or variation points) and complexity (numbers of constraints and rules) of each variability modeling approach using the transformation approach TRAVART [25]. We evaluated the applicability of the identified information loss classes by investigating whether and how often the information loss classes occur in the transformed models. Additionally, we performed an automated analysis of the resulting variability model using configuration sampling provided by TRAVART [25] to verify models too big to be verified by hand. We present our evaluation results in Section 6.

3 BACKGROUND

We use an extended version of an online shop product line [4] to illustrate the core concepts of feature models, decision models, and OVM models. Figure 1 depicts a feature model created with FeatureIDE [37], which also supports UVL feature models [51], a decision model using a tabular representation, and an OVM model (using the graphical OVM notation). As we can not present all details of the approaches due to space constraints, we refer the reader to existing core publications [19, 37, 40, 51] and published comparative studies and applications [10, 16, 22, 25, 43–45].

Feature models are used to capture the commonalities and variability of a system using features. Features can be optional (e.g., `Search`) or mandatory (e.g., `Catalog`). In FeatureIDE and UVL feature models, a feature can be hidden (e.g., `Platform`), meaning that the feature will not be shown during the configuration process. In most feature modeling approaches, features are hierarchically organized using parent-child relations. When a child is present in



ID	Question	Type	Range	Card.	Constraint/Rule	Visibility
Payment	Which payment methods should be supported?	Enum	DebitCard | CreditCard	1:1		
Search	Should a search function be supported?	Boolean	true | false		if (Search) { UserManagement = Security } if (!Search) { Sort = true }	
Categories	Split products into categories?	Boolean	true | false			
Sort	Should products be sortable?	Boolean	true | false		if (!Sort) { Search = true }	

UserManagement	Which user management options should be added?	Enum	Orders | Security | Payments | Wishlist	1:4	if (Security) { disAllow(Payments) }
					if (!Security) { allow(Payments) }
					if (Payments) { disAllow(Security) }
					if (!Payments) { allow(Security) }	

Newsletter_percent	How many percent off?	Number	0 - 100			
Newsletter_mail	From which mail should the newsletter be sent?	String				
Review	Which review techiques should be supported	Enum	Stars | Numerical | Comments	2:3		

Figure 1: (A) Feature Model [37, 51], (B) Decision Model [19], and (C) Orthogonal Variability Model [40] of an online shop [4].

a configuration, also its parent feature is present, but not necessarily vice versa. Child features of a parent can be organized in groups with a particular cardinality. FeatureIDE [37] allows *OR* (1 to n of the features in the group can be selected; e.g., in the group `UserManagment` the enhancements `Orders`, `Security`, `Payments` or `Wishlist` or any combination of them can be selected) and *alternative* (1 to 1, meaning exactly one of the features in the group can be selected; e.g., exactly one `Payment` technology must be selected) relations. In UVL [51] any cardinality can be used (n to m, meaning between n and m child features of the group can be selected). Additional relationships between features are created with cross-tree constraints. For instance, requires and excludes constraints can be used to model dependencies between features. In FeatureIDE [37] and UVL [51], constraints are defined as propositional formulas.

Decision Models focus on the problem-space view of a product line's variability and the decisions necessary to derive products. The decision model shown in Figure 1 in a tabular representation uses the approach DOPLER [19]. In DOPLER, a decision is specified by a unique id, arbitrary text (e.g., question) and its decision type. A decision is either of type Boolean (e.g., *Search* and *Categories*), String (e.g., *Newsletter_mail*), Number (e.g., *Newsletter_percent*), or Enumeration (e.g., *Payment* and *UserManagement*). For each decision the modeler can specify a value *range*, which specifies an interval or a set of values (e.g., all double values from 1 to 100 for Number decisions or a set of allowed characters for email addresses for String decisions) allowed for the decision. The validity of a provided value is verified when the value is set.

Relationships between decisions can be specified via visibility conditions and rules. A decision's visibility condition defines when the decision can be made in the derivation process. It defines which decisions have to be made before such that the decision becomes answerable, i.e., a value can be selected. Per default a decision can by made at any time, i.e., visibility condition equals true). Thus, visibility conditions define a hierarchy and order, whereas rules define constraints within the decision model. Each decision can specify rules which fire after the decision has been made. A rule can set values to other decisions. For instance, if decision *Search* is set to true, the value of decision *UserManagment* is set to *Security* via a rule. Different functions can be used in visibility conditions and rules. Function IsTaken(D), returns true if decision D has already been made, irrespective of the value, i.e., a Boolean decision with value false is still taken. Function *IsSelected(D)* is more specific and only returns true if the decision has been selected. Functions *allow(V)* and *disallow(V)* enable the modification of allowed values

of enumeration decisions. For instance, in decision *UserManagment* the values *Security* or *Payments* exclude each other, using the allow and disallow functions in their rules. DOPLER heavily relies on decision values, especially with Number and String decisions and provides the *SetValue* and *GetValue* functions to set and get values to/from decisions in rules. DOPLER uses a domain-specific language close to Java to specify the visibility conditions and rules [19].

Orthogonal Variability Models [40] represent a product line's variability using variation points and variants. Each variation point may specify additional, dependent variation points or concrete variants to choose from during product derivation. A variation point is either mandatory (e.g., Payment) or optional (e.g., User Management). Each variation point must comprise at least one variant somewhere among its descendants and each concrete variant is connected to at least one variation point (e.g., variant Search is connected to the variation points Product Selection and Sort or Search). Similar to feature groups in feature models, variation points can group variants together and can define a cardinality (either *OR*, alternative and any n-to-m selection) to select among the group of variants (e.g., an *OR* group for the variants of variation point User management or alternatives for the variants of variation point Payment). Relationships between variation points and variants can be created using requires or excludes constraints (only excludes constraints are bidirectional by default). Both constraint types can be either defined between variants (e.g., variant Search requires variant Security), between a variant and a variation point, or between two variation points. For more complex constraints, OVM uses variation points, e.g., the Sort or Search variation point.

4 INFORMATION LOSS CLASSES

Major concept differences between variability modeling approaches challenge the implementation of transformations [24]. For instance, feature modeling captures *commonalities* of the modeled system, which are out of scope for decision modeling and orthogonal variability modeling. These approaches focus only on the *variability* of the system. Consequently, all mandatory features (cf. `Shopping Basket` or `Design` in Figure 1A) are not part of the decision model. In OVM, the modeler can specify mandatory variation points (cf. variation point Catalog), which means one has to select among specified variants during product derivation. Another example are the supported *data types*. For instance, in DOPLER decision models [19], it is possible to model the configuration of components using Number and String decisions (e.g., decisions *Newsletter_percent* and *Newsletter_mail* to specify the feature `Newsletter` more concretely; cf. Figure 1). Also, DOPLER enables the specification of rules, depending on actual decision values, which is not possible in feature modeling and orthogonal variability modeling. Other examples of such concept differences are *hidden features* in feature modeling, the way *hierarchy* is handled in the approaches, and the expressiveness of the used *constraint languages*. These differences require specific implementations and workarounds to support roundtrip transformations, or lose information on different levels.

To clearly define information loss classes, we define the following key terms: (a) a *unit of variability* is the main modeling concept of a variability modeling approach (e.g., a feature, a decision, a variant) [24], (b) a *property* is a key characteristic of a unit of variability (e.g., abstract or hidden for features or the type of a decision), (c) an *entity* is a unit of variability, including its *properties*, (d) a *relationship* describes dependencies between one or multiple entities (e.g., optional features, constraints or rules between two or multiple entities, or a visibility condition), (e) a *capacity* of an entity describes the set of values that can be assigned to it during product derivation (e.g., selected/deselected for optional features or Boolean decisions, or all possible values of a Number decision).

If an entity or relationship can not be (fully) transformed into the target language, information is lost during transformation. We identified *Conceptual Losses*, i.e., regarding *Configurability* and *Semantics*, as well as *Structural Losses* as distinct information loss classes. We also specify *No Information Loss* to enable the classification of fully supported transformations. We introduce the classes in ascending severity, in terms of their influence on the model semantics and configuration space.

No Information Loss: An entity or a relationship can be transformed into another entity or relationship of the target notation and, hence, can be restored during the roundtrip at its full capacity. For instance, an optional feature of a feature model can be transformed into a Boolean decision of a decision model. Both entities represent an option during the product derivation process.

Structural Loss: An entity or a relationship can sometimes not be transformed back into the identical entity or relationship which was present in the original model. For instance, constraints originally specified using conjunctions and disjunctions changing during a roundtrip transformation to constraints using implies operators. Another example are a set of constraints (with conjunctions and disjunctions), which during transformation can be split up into multiple constraints. During roundtrip transformation these split up constraints will remain split up, as one cannot reconstruct which constraints originally where concatenated. Similar syntactical changes happen during mitigation strategies to avoid information loss during transformation. For instance, when a UVL model imports other UVL models and these models are unfolded during transformation to avoid a *configurability loss*. The structure of the original model then differs from the roundtrip model, as additional entities and relationships are added, but the configuration space of the original and transformed model can still remain identical.

Semantic Loss: Describes that an entity or relationship could not be transformed into the target language because it is not supported in the target approach, but the configuration space of the original and the transformed model still remain identical. For instance, commonalities in feature models are not part of decision models or orthogonal variability models, but this does not change the configuration space of the (target/roundtrip) model. Also, abstract features can be transformed into decisions, but the property abstract is lost. A *semantic loss* is a *conceptual loss*.

Configurability Loss: Describes information losses, which change the configuration space of the resulting (target/roundtrip) model. An entity or relationship cannot be transformed into the (target/roundtrip) model at its full capacity because it is not fully supported. For instance, (most) feature modeling approaches do not support numerical or textual values to be specified during configuration, like Number/String decisions supported in decision modeling, but rather as attributes [48]. As a result, the configuration space of

the transformed (target/roundtrip) model changes compared to the original model. A *configurability loss* is a *conceptual loss*.

5 MAPPING TABLES

We provide an engineering view on the concepts of the four variability modeling approaches as three mapping tables (cf. Tables 1–3), each showing a comparison of two approaches. We use UVL [51] as a pivot language for the mapping tables and thus present which concepts of UVL map to FeatureIDE feature model [37], DOPLER decision model [19], and OVM [40] concepts. We selected UVL [51] as a pivot language as the transformation approach TRAVART [25] uses UVL as pivot language as well. We outline for which concepts *no information* is lost or which type of loss (*structural loss*, or *semantic loss* or *configurability loss*) occurs. In the tables, we highlight both directions between two approaches in the upper part and lower part, respectively. We also use a color coding scheme, i.e., *no information loss* (green), *structural loss* (blue), *semantic loss* (yellow), *configurability loss* (red), to indicate the different cases.

We discuss the transformation of an original model into a target model and point to challenges when transforming the target model back to the original notation, i.e., into a roundtrip model. We explain how different losses can be (partially) addressed through either additional assumptions, normal forms, meta-data or by applying other (mitigation) strategies such as bit blasting [39]. A *normal form* is a standard structure which covers the transformed entities and relationships at their full capacity. During the roundtrip, the standard structure is recognized to restore the original entities and relationships in the roundtrip model. Typically, if an original model is in normal form, then a less severe form of information loss (or possibly no information loss) occurs. For instance, mandatory features in feature modeling can be transformed into a Boolean decision with visibility condition false && parent. The visibility condition ensures that (a) the decision is not visible during product derivation and (b) the feature is restored as a child of the correct parent feature. *Meta-data* is additionally stored structured (String) data about entities and relationships, without having an impact on the model in terms of number of entities, relationships or configuration space. During the roundtrip, the additional data helps to restore the original entities and relationships at their full capacity and structure in the roundtrip model.

5.1 UVL vs. FeatureIDE

UVL [51] and FeatureIDE [37] are both feature modeling approaches and there is a considerable overlap between the two approaches. Nevertheless, there are differences in allowed concepts, which cause certain information to be lost. For both approaches, we focus on the core modeling approach without extensions, such as constraints among feature attributes. Table 1 outlines the mapping between UVL feature models [51] and FeatureIDE feature models [37].

UVL allows the separation of feature models into multiple models (files) and therefore allows to *import* UVL models into each other. Further, UVL allows to specify *namespaces*, to enable referencing features from imported feature models. During transformation to FeatureIDE feature models these concepts are lost as they are not supported. This results in a *configurability loss*. To mitigate this loss, one can unfold the UVL model scattered across multiple files into one single model and transform the resulting model. This model does capture the full configuration space of the original UVL model and hence does lose *no information*. However, during the roundtrip the semantics of *import* and *namespaces* of the original model are lost, resulting in a *semantic loss* in the roundtrip model.

Table 1: Mapping UVL [51] concepts to FeatureIDE [37] (upper part) and vice versa (lower part).

UVL Feature Model		FeatureIDE Feature Model	Roundtrip
Model	imports	not supported	-
	namespaces	not supported	-
Feature Properties	Mandatory	Mandatory Feature	Mandatory Feature
	Optional	Optional Feature	Optional Feature
	Abstract	Abstract	Abstract
	Hidden	Hidden	Hidden
Attributes	String	not supported	-
	Numeric	not supported	-
	Vector	not supported	-
Group	Or	Or	Or
	Alternative	Alternative	Alternative
	Group Cardinality (n:m)	Set of constraints	Set of constraints
Constraints	Not	Not	Not
	And	And	And
	Or	Or	Or
	Requires	Requires	Requires
	Equivalence	Equivalence	Equivalence
FeatureIDE Feature Model		**UVL Feature Model**	**Roundtrip**
Feature	Mandatory	Mandatory Feature	Mandatory Feature
	Optional	Optional Feature	Optional Feature
Group	And	And	And
	Or	Or	Or
	Alternative	Alternative	Alternative
Properties	Abstract	Abstract	Abstract
	Hidden	Hidden	Hidden
Constraint	Not	Not	Not
	And	And	And
	Or	Or	Or
	Implies	Implies	Implies
	Equivalence	Equivalence	Equivalence

no information loss structural loss semantic loss configurability loss

UVL enables features to store additional feature information as attributes. These attributes have one of three types: *String* – any value representable as a String, *Numeric* – any numeric value of type double or *Vector* – a set of String/Numeric values. There is an extension in FeatureIDE available to create an attributed feature model, which would be able to store the attributes 1:1. For our work, we only consider default FeatureIDE feature models, where no attributes are supported and therefore these concepts are not transferable. As a result, attributes specified with a feature in UVL are lost in the transformation to FeatureIDE feature model. However, attributes do not influence the configurability of the feature model and the loss classifies as a *semantic loss*.

UVL allows the specification of feature groups with cardinality *n:m*, whereas FeatureIDE models uses a set of constraints. These constraints capture each possible feature combination to fulfill the mutex cardinality (cf. the last constraint of the FeatureIDE feature model in Figure 1A). Even though replacing the mutex group by a set of constraints does describe the same configuration space and *no information* is lost, it has been shown to be practically infeasible for certain values of n and m [13]. This adds to the mismatch between the provided constraint mechanisms in current variability modeling approaches and the needed capabilities by industry, i.e., mutex groups in this case [35]. During the roundtrip, without additional meta-data, these constraints cannot be restored into a *n:m* cardinality, which creates the same set of constraints in the roundtrip model. Hence, causing a *structural loss*.

Table 2: Mapping UVL [51] concepts to DOPLER [19] (upper part) and vice versa (lower part).

UVL Feature Model		DOPLER Decision Model	Roundtrip
Model	imports	not supported	-
	namespaces	not supported	-
Feature	Mandatory	Boolean decision with visiblity condtion	Mandatory Feature
	Optional	Boolean decision	Optional Feature
Properties	Abstract	not supported	-
	Hidden	not supported	-
Attributes	String	not supported	-
	Numeric	not supported	-
	Vector	not supported	-
Group	Or	Enumeration decision with cardinality 1:n	Or
	Alternative	Enumeration decision with cardinality 1:1	Alternative
	Group Cardinality (n:m)	Enumeration decision with cardinality n:m	Group Cardinality (n:m)
Constraints	Not	Deselect Rule	Not
	And	requires rules	And
	Or	Visibilitiy conditions requires/excludes rules	Or
	Requires	requires rule	Requires
	Equivalence	two requires rules	Equivalence
DOPLER Decision Model		**UVL Feature Model**	**Roundtrip**
Decision	Boolean decision	optional feature	Boolean decision
	Enumeration decision	feature group with cardinality	Enumeration decision
	Number decision	not supported	-
	String decision	not supported	-
Expression	IsSelected	constraint literal	IsSelected
	IsTaken	not supported	-
	Not	not literal	Not
	And	and literal	And
	Or	or literal	Or
	Range	not supported	-
	GreaterThen	not supported	-
	LessThen	not supported	-
	Equals	not supported	-
	GreaterEquals	not supported	-
	LessEquals	not supported	-
Action	Allow	excludes constraint	Allow
	DisAllow		DisAllow
	SetValue	not supported	-
	GetValue	not supported	-
	SetSelected	requires	SetSelected
	DeSelect	excludes	DeSelected

no information loss structural loss semantic loss configurability loss

5.2 UVL vs. DOPLER

El-Sharkawy et al. [22] presented a semantic comparison between feature modeling and decision modeling and found that both approaches are equal under the condition that a powerful constraint language is used. As discussed before, UVL feature models [51] support propositional logic to define constraints between features, whereas DOPLER decision models [19] use a domain-specific language close to Java [19]. Table 2 shows the differences in expressiveness in the mapping table. For our mapping table, we only considered the base concepts of DOPLER decision modeling [19] and excluded the possibility of creating custom meta-models (e.g., to define additional decision properties).

Similar to the UVL to FeatureIDE transformation, *imports* and *namespaces* are lost during the transformation from UVL to DOPLER, because DOPLER does not support *imports* or *namespaces*, resulting in a *configurability loss*. Using the same unfolding and resolving strategy this loss can be mitigated to a *semantic loss*.

DOPLER decision models [19] do not provide a mechanism to store attributes with decisions without defining custom meta-models. Consequently, feature attributes are lost during the transformation, causing a *semantic loss*. One could add additional *String*/*Number*/*Enumeration* decisions to capture *String*/*Numeric*/*Vector* attributes, respectively. This would increase the overall number of features during the roundtrip transformation, causing a *structural loss*. Such a *normal form* would increase the complexity of the resulting decision model considerably, as additional visibility conditions and rules must be introduced to set the resulting values to the respective decisions. Most importantly such a *normal form* would introduce an ambiguity of decisions in a decision model and attribute decisions, which could, despite theoretically reducing the lost information to a *structural loss*, cause problems during the roundtrip transformation and cause a *configurability loss*.

Decision modeling does not provide mechanisms to make decisions *abstract* (cf. feature `Payment` in Figure 1A) or *hidden*. This information is lost during transformation, causing a *semantic loss*. For capturing the *hidden* property of a feature, one could use the *visibility condition* of a decision. By setting the visibility condition to false, it will never be shown to engineers during product derivation and thus remain hidden. However, this mapping could cause ambiguities between visibility conditions because during transformation, visibility conditions might be created to capture the feature tree hierarchy as well as to represent mandatory features using the same visibility condition. Also, utility decisions use visibility conditions to capture relationships between decisions. Additional *meta-data* would be necessary to resolve these ambiguities.

Mandatory features (cf. feature `Design` in Figure 1A) in UVL are out of scope of decision modeling. A rule in the decision representing the parent of the mandatory feature, in the combination with a visibility condition false && parent builds a *normal form* and captures the mandatory feature and its parent feature for the roundtrip.

Number and String decisions in DOPLER [19] are used to request any numerical or String value from a user during the configuration of the system (cf. *Newsletter* decisions in Figure 1B). UVL [51] does not support such numerical or String features. Hence, such decisions are transformed to a single feature and the value *range* of the Number and String decision is not captured in full capacity. This *configurability loss* can be (partially) addressed.

First, one can reuse feature attributes to store the values assigned during configuration. UVL does not support attribute constraints and setting them dynamically via constraints. However, such techniques would be necessary to cover the full capabilities of Number or String decisions. As a result, the *configurability loss* remains.

Second, one can discretize the possible decision values and create an alternative feature group to capture the value range. The parent feature of the feature group represents the Number or String decision and its child features represent the discrete values of the range. This approach allows specifying constraints capturing rules on specific values based on comparison expressions (cf. *GreaterThan*, *LessThan*, *Equals*, *GreaterEquals* or *LessEquals* in Table 2). For instance, if decision *d1* has a value range of 1…5 and specifies the rule if (GetValue(d1) > 3) then select(d2), then decision *d2* is selected if the value of decision *d1* is greater than 3. This rule can then be transformed into a constraint d1:4 or d1:5 implies d2 were features d1:4 and d1:5 represent the values. During roundtrip transformation, more decisions are created by this approach, as for each feature a Boolean decision (as defined by the mapping) is created. Additionally, one has to accept that the Number or String type of the decision is lost and float numbers are still not supported, which causes a *configurability loss*. Adding *meta-data* to the transformations would be necessary to further reduce the information loss.

Third, the previously discussed option to discretize the range values can be extended by bit blasting [39]. However, this approach is only suitable for Number decisions. Bit blasting enables the representation of all numeric values in binary format, which can not be done for all possible String values. For a Number decision an OR group is created, which has a child feature for each bit necessary to cover the value range of the Number decision. For instance, if 64-bit numbers (i.e., the full value range of a Java double [1]) should be supported, the OR group would consist of 64 children. Also comparison expressions can then be fully supported for these 64-bit numbers, as a set of constraints can be used to capture the expressions. Still, even though this mapping results in the biggest coverage of the Number decision values and practically would address the *configurability loss*, it still causes a *structural loss* during the roundtrip. Additionally, as each bit is separately represented by a Boolean decision, the resulting set of rules would grow considerably.

Functions to capture dependencies between decisions (cf. *IsSelected* and *IsTaken* in Section 3) are not fully supported in UVL feature models [51]. Specifically, for the parameter decision of function *IsTaken(D)* the constraint literal D or not D can be derived. If the function is used in a rule selecting a different decision, the rule can be transformed into a constraint. This constraint is then transformed into a rule of different structure in the roundtrip model, causing a *structural loss*. Recognizing the normal form, this loss can be mitigated. When the function is used in a visibility condition, the order in which decisions can be made is lost during the transformation, because the resulting constraint evaluates to true anyway. This does not influence the configurability of the model, but it still is a *semantic loss*. Using *meta-data* this loss can be mitigated. In a rule setting a decision value, i.e., rule if (isTaken(D1)) then SetValue(D2, GetValue(D1)) sets the value of decision *D2* depending of decision *D1*, the rule condition can not be transformed and is lost. Hence, a *configurability loss* occurs. Such rules are often created in the context of Number and String decisions. Again, the previous discussed workaround and *meta-data* can be used to address this loss. For the parameter decision of function *IsSelected*, a constraint literal in a propositional logic constraint can be created, i.e., features A in a A implies B constraint. As a result, it is classified as a *no information loss*. If a visibility condition only consists of one *IsSelected* function, the feature tree can be derived from the function. The parent-child relation is defined by the function parameter decision (parent) and the decision specifying the visibility condition (child).

The functions *SetValue* and *GetValue* are only supported for enumeration decisions. These two functions can be used to create constraints based on the resulting feature group elements. For Number and String decisions these functions are not supported in UVL feature models [51] and therefore lost during transformation. As these functions are an important part of the configurability of decision models, a *configurability loss* occurs. These functions can be transformed in a way that they support the applied workaround to Number and String decisions. As a result, these functions help mitigate the loss, but still do not resolve it and create the same type of losses during the roundtrip. Functions *allow*/*disallow* are transformed to an excludes constraint in feature modeling. The functions are restored during the roundtrip, hence *no information* is lost during the transformation.

Table 3: Mapping UVL [51] concepts to OVM [40] (upper part) and vice versa (lower part).

UVL Feature Model		OVM	Roundtrip
Model	imports	not supported	-
	namespaces	not supported	-
Feature	Mandatory	Mandatory Variation Point/Variant	Mandatory
	Optional	Optional Variation Point/Variant	Optional
Properties	Abstract	not supported	-
	Hidden	not supported	-
Attributes	String	not supported	-
	Numeric	not supported	-
	Vector	not supported	-
Group	Or	Or	Or
	Alternative	Alternative	Alternative
	Group Cardinality (n:m)	Group Cardinality (n:m)	Group Cardinality (n:m)
Constraints	Not	Variation Point with excludes	Not
	And	Variation Point of mandatory children	And
	Or	Variation Point of optional children	Or
	Requires	requires	Requires
	Equivalence	two requires	Equivalence
OVM		**UVL Feature Model**	**Roundtrip**
Unit	Optional Variation Point	Optional Feature	Optional Variation Point
	Mandatory Variation Point	Mandatory Feature	Mandatory Variation Point
	Variant	Feature	Variant
Choices	Or	Or	Or
	Alternative	Alternative	Alternative
	Group Cardinality (n:m)	Group Cardinality (n:m)	Group Cardinality (n:m)
Dependencies	requires	implies constraint	requires
	excludes	excludes constraint	excludes
	optional	optional feature	optional
	mandatory	mandatory feature	mandatory

no information loss | structural loss | semantic loss | configurability loss

5.3 UVL vs. OVM

Table 3 shows the mapping between UVL feature models [51] and OVM [40].

Again, *imports* and *namespaces* are lost during the transformation from UVL to OVM, causing a *configurability loss*. Applying unfolding and resolving references can be applied to prevent a *configurability loss*, but causing a *semantic loss* during transformation. OVM provides the possibility to define multiple entry points in their models, i.e., by specifying multiple root variation points in the model. Consequently, the different referenced UVL models can be integrated within one OVM model, without unfolding. The hierarchy of the UVL model is then enforced using additional requires constraints. During the roundtrip transformation, the *normal form* of separate OVM models is used to split up the models into the separate UVL files, resolving the *semantic loss* during transformations and *no information* is lost.

Similar to decision modeling, OVM [40] focuses on the variability of the modeled system and does not support *abstract* or *hidden* features. Hence, these properties of a feature are not transferable to OVM and therefore lost during transformation. As both properties do not have an influence on the configurability on neither the UVL feature model nor on the OVM, this loss classifies as a *semantic loss*.

Attributes are out of scope of OVM and, similar to FeatureIDE feature models [37] and DOPLER decision models [19], these concepts are lost during the transformation from UVL to OVM. The configurability of neither the UVL model nor the OVM is influenced by this loss, so the loss classifies as a *semantic loss*.

For constraints specified in feature modeling a combination of optional/mandatory variation points and requires/excludes constraints can be used. OVM does not support constraints with a variation point as the source and a variant as the target. As a result,

a constraint defined in UVL may cannot be transformed into OVM, which would cause an *configurability loss*. However, any constraint can be defined between two variation points and a variation point can define mandatory variants. Applying a *normal form* of a variation point for each feature and additionally a mandatory variant if the feature is a leaf feature (cf. feature `Sort` in Figure 1A) resolves the loss. The variation point and the mandatory variant then together represent this feature and ensure the configurability of the resulting UVL model and *no information is lost*. However, during the roundtrip of the model back to an OVM model, this *normal form* adds an additional level to the OVM, i.e., replacing the variant Sort in Figure 1C by a variation point and an mandatory variant of the same name, causing a *structural loss*.

6 EVALUATION

We investigate and evaluate the applicability of our information loss classes. Our aim is to find out whether they occur in existing models and if yes, how frequently. We transformed 31 feature models (25 FeatureIDE feature models [37], 6 UVL feature models [51]), 6 DOPLER decision models [19] and 3 OVM models [40] into each other, using our variability model transformation approach TRAVART [25]. We selected the variability models from online repositories.[1,2]

The FeatureIDE feature models' comprise from 7 to 2,513 features (median 40, rounded average 236) and between 0 and 3,455 constraints (median 6 and average 322). The UVL feature models have between 10 and 18,616 features (median 356.5 and average 3,499), and between 2 and 3,455 constraints (median 126.5 and average 849). The DOPLER decision models and OVM models are smaller. The DOPLER models consist of 4 to 26 decisions (median 6.5 and average 10), and between 0 and 7 rules (median 1.5 and average 3). The OVM models have 9 to 36 variation points/variants (median 17 and average 21), and 1 to 11 constraints (median 3 and average 5). The dataset was used to answer the following research question, to discuss the information lost when transforming the models and whether the information losses conform to our mapping table.

RQ. *How frequently do the information loss classes happen in transformed variability models*? Using the described information loss classes and mapping tables, we manually inspected the source files and visual representation using FeatureIDE [37], of the transformed models to confirm that the defined information loss classes can be found in the transformed models. Therefore, we first visually compared the original model with the resulting *rountrip models* and further analyzed the differences using the respective intermediate model. Additionally, for variability models too large (in number of features/decisions/variation points/variants) or too complex (in number of constraints/rules) to be checked by hand, we rather verified their configuration space. We sampled a set of valid and invalid configurations of the original model using TRAVART [25] (which uses configuration samplers) and verified the samples in the *intermediate* and the roundtrip model. We then classified the observed losses using the defined information loss classes and counted the lost and added entities and relationships in the respective models.

[1] https://github.com/Universal-Variability-Language/uvl-models
[2] https://github.com/FeatureIDE/FeatureIDE/tree/v3.8.1/featuremodels

Specifically, we define the following metrics to assess the applicability of the information loss classes (cf. Table 4):

No. of Configurability Losses (#ConfigLoss). We count the number of features/decisions and constraints/rules, which have a different type as in the original model and changed the configuration space of the rountrip model. Additionally, we consider decisions which have changed, were added or removed because of this change (e.g., one Number decision vs. a set of decisions in the roundtrip model).

No. of Semantic Losses (#SemLoss). We count the number of features/decisions and constraints/rules in the rountrip model, which remained present but changed their properties (e.g., abstract features which are not abstract in the roundtrip model anymore), or mandatory features, which are not present in the decision models.

No. of Structural Losses (#StrucLoss). We count the number of features/decisions and constraints/rules in the roundtrip model, which remained semantically equal but changed their structure (e.g., constraints originally specified using conjunctions and disjunctions changing to constraints using implies operators).

Roundtrip quality (R-Quality). We assess whether a roundtrip model is *identical*, *equal*, or *loss* to the original model. A roundtrip model is *identical* to the original model, if each entity has the same type and no entity or relationship was changed, added, or removed. An *equal* roundtrip model may have additional entities or relationships compared to the original model, but the configuration space remains identical. A *loss* roundtrip model indicates that information was lost and the configuration space changed.

6.1 Applicability (RQ)

Table 4 shows the results of our study. For each model, the consisting entities and relationships of the original model and the intermediate model, as well as the roundtrip model metrics are listed. TRAVART [25] uses UVL as a pivot language (i.e., transform every variability model to a UVL feature model first). Also, its OVM transformations to and from feature models heavily uses *meta-data* of the original models stored in the intermediate models to allow *identical* roundtrip transformations. As a result, the transformations via OVM either did not lose any information (i.e., to/from FeatureIDE and UVL feature models), or lost the same information as the transformation via UVL (i.e., to/from DOPLER decision models). Therefore, we only list the transformations involving feature models either as original model or as intermediate model in Table 4.

During the other transformations, i.e., FeatureIDE feature models and UVL feature models to DOPLER and back, DOPLER to UVL and back, and OVM to UVL and back, one rountrip model was *identical* to the original model. 3 DOPLER models lost information during transformation and have a *R-Quality* of *loss*. The remaining 36 models were transformed *equally* to the original model.

When transforming the 25 FeatureIDE feature models into UVL and back *no information* was lost. We expected this result because of the overlap between FeatureIDE and UVL found in the mapping table (cf. Section 5.1).

Of the 6 DOPLER decision models, 3 roundtrip models are different to the original model and have a *R-Qualtiy* of *loss*. In all 3 cases, the lost information originates from Number decisions, which are not supported in UVL and introduces a *configurability*

Table 4: Information loss classes found in the roundtrip transformed variability models and the roundtrip model quality.

Original Model			Intermediate		Roundtrip Metrics			
FeatureIDE → DOPLER	#Features	#Constraints	#Decisions	#Rules	#ConfigLoss	#SemLoss	#StrucLoss	R-Quality
APL	23	2	24	14	0	9	0	equal
Automotive01	2,513	2,833	2,927	4,456	0	25	203	equal
BankingSoftware	176	4	190	101	0	0	4	equal
BerkeleyDB	76	20	82	71	0	23	26	equal
BigDataSystem	628	0	711	638	0	3	0	equal
BusyBox	631	681	635	556	0	1	36	equal
DellLaptopNotebook	47	105	55	151	0	0	105	equal
ebay	40	6	48	37	0	0	1	equal
eShop_featureide	326	21	365	227	0	134	0	equal
eShop_splot	10	4	12	13	0	0	4	equal
FameDB2	22	0	28	26	0	9	0	equal
FeatureIDE	19	4	20	17	0	0	3	equal
FinancialServices	771	1,080	903	2,926	0	0	3,357	equal
GPL Medium	38	15	44	60	0	11	6	equal
GPL Small	25	7	29	36	0	9	7	equal
GPL Tiny	7	1	9	10	0	0	0	identical
MobilePhone	10	2	12	15	0	0	1	equal
OnlineShop	14	3	16	19	0	4	1	equal
SafeBali	24	0	25	14	0	7	0	equal
TightVNC	28	3	31	30	0	7	0	equal
Ubuntu	263	72	300	382	0	2	72	equal
VMTools	42	3	46	27	0	2	1	equal
WikiMatrix	21	14	25	57	0	0	9	equal
Windows 7	70	0	89	130	0	2	0	equal
Windows 8	78	0	95	104	0	1	0	equal
UVL → DOPLER	#Features	#Constraints	#Decisions	#Rules	#ConfigLoss	#SemLoss	#StrucLoss	R-Quality
automotive02_4	18,616	1,369	19,967	30,591	0	4	451	equal
axTLS	96	14	101	82	0	5	37	equal
Server	10	2	12	12	0	2	0	equal
uClibc	313	56	334	445	0	22	176	equal
uClinux-base	380	3,455	416	1,187	0	0	3,384	equal
uClinux-distribution	1,580	197	1,590	410	0	10	194	equal
DOPLER → UVL	#Decisions	#Rules	#Features	#Constraints	#ConfigLoss	#SemLoss	#StrucLoss	R-Quality
ASEJ1	4	2	109	96	198	0	6	loss
DissModel	11	0	116	0	102	0	10	loss
DOPLERTools	26	7	50	15	16	0	36	loss
eShop	6	7	13	4	0	0	8	equal
HCSSDM	7	0	28	0	0	0	33	equal
VaMoS	5	1	11	1	0	0	8	equal
OVM → UVL	#VP & V	#Constraints	#Features	#Constraints	#ConfigLoss	#SemLoss	#StrucLoss	R-Quality
MPPL	9	1	10	1	0	0	6	equal
OnlineShop	17	3	14	4	0	0	11	equal
RFW	36	11	37	11	0	0	25	equal

loss. We expected this loss when transforming the decision models to UVL. The transformation applies the second workaround (discretize the value range of the Number decision and create an alternative feature group, cf. Section 5.2) to (partially) address the loss in the roundtrip transformation.

Of the 31 feature models, 21 (16 FeatureIDE feature models and 5 UVL feature models) specified abstract features. When transforming the models into a DOPLER decision model and back the abstract property of these features was lost, causing a *semantic loss*. In total 292 abstract features were affected. This loss was expected (cf. mapping Table 2 in Section 5.2).

Only 1 UVL model was available as a multi-file UVL model. The transformation uses an unfolding process to avoid a *configurability loss* in the intermediate model, but this causes a *structural loss* during the roundtrip. In 19 feature models (15 of type FeatureIDE and 4 UVL feature models) constraints changed in the roundtrip model compared to the original model. Most changes originate from requires/excludes constraints modeled with conjunctions and disjunctions. These constraints are each transformed into a requires/excludes constraint using an implies operator, creating a *structural loss*. For the two feature models *BankingSoftware* and *eShop_splot* these were the only cases of lost information. Additionally, in 17 feature models (12 FeatureIDE feature models and 5 UVL feature models) the number of constraints increased, due to splitting up constraints into multiple ones. In total, the 17 feature models contain 8,068 more constraints than the original ones, resulting in a *structural loss*. In 2 feature models the total number of constraints reduced by 1 because the transformation removed a redundant constraint, which classifies as a *structural loss*. All 6 DOPLER decision models contained Enumeration decisions, which during roundtrip transformation with UVL got split up into multiple Boolean decisions, one Enumeration decision and multiple rules. Hence, a *structural loss* was introduced by the transformation. We found *structural losses* in all 3 OVM models, were for each variant in a OVM, an additional variation point was introduced.

Every identified change to one of the variability models matches the elaboration on the mapping tables shown in Section 5. We expected the change in the OVM models because of the limitations of constraints between variation points and variants (cf. Section 5.3). The changes to constraints were expected since the same constraint can be created in different ways. We verified these feature models by manual inspection and configuration sampling. For instance, for the two feature models *ebay* and *VMTools*, which lost a redundant constraint, we verified 11 valid and 72 invalid configurations and 13 valid and 83 invalid configurations to be also valid/invalid in the intermediate DOPLER and OVM models and in the resulting roundtrip models, respectively.

Our results show that the defined information loss classes can be found in transformed variability models and therefore are applicable to existing variability models and indicate how information is lost during a one-way transformation and a roundtrip transformation. Thus, classified lost information can help modelers to investigate transformed and imported/exported variability models, increase their awareness for information loss and guide them when developing new transformations.

6.2 Threats to Validity

We did not perform a systematic approach to identify related work or other variability modeling approaches of interest. We also did not involve other variability modeling experts, which is a threat to validity of our research method. However, we (the authors of this paper) carefully discussed and reviewed each step of the process and reached a consensus on all reported results. We only selected four variability modeling approaches to perform our study and we selected the variability models for the evaluation of the information loss classes ourselves. As a result, we might have introduced a certain bias towards some approaches and models (e.g., feature

models) as we had more models of this type available. However, we selected the models from existing online repositories. Additionally, we only used one existing variability model transformation approach to evaluate our information loss classes and did not assess import/export capabilities of existing variability modeling tools e.g., FeatureIDE [37] or Fama [9]. Thus, we introduced a certain bias towards the implementation of TRAVART [25]. We used the approach because, to the best of our knowledge, it is the only approach supporting all four variability modeling approaches. We argue that our study still shows the applicability of the defined information loss classes and can easily be extended towards other approaches and tools. We consider this as future work.

We did not perform a usefulness study of the defined information loss classes with external modelers, but rather demonstrated their applicability to existing variability models. We consider a usefulness study part of our future work.

7 RELATED WORK

The software product line community has surveyed and compared *variability modeling approaches* frequently [5, 7, 14, 17, 28, 41, 45]. For instance, Seidl et al. [48] compared multiple different feature modeling approaches as the basis for their approach. Acher et al. [2] presented an approach to formally reason about the differences of feature modeling approaches on the semantic level, i.e., on the sets of configurations. Thüm et al. [52] presented an algorithm to identify the relationship between two feature models and to check whether one of them is a specialization, a refactoring, or a generalization of the other feature model, or neither of these types. Czarnecki et al. [16] compared feature modeling and decision modeling in detail, discovering more similarities than differences. El-Sharkawy et al. [22] compared the capabilities and expressiveness of basic feature modeling with basic decision modeling. Eichelberger et al. [20] analyzed the expressiveness and analyzability of variability modeling concepts of multiple variability modeling approaches. These works have been a key basis for our mapping tables and definitions of information loss classes.

Researchers have worked towards *unifying variability modeling* from multiple directions. For instance, Schulze and Hellebrand [47] presented the Variability Exchange Language (VEL) to enhance the information exchange among variant management tools. Sepúlveda et al. [49] proposed a unified feature language meta-model building on existing feature model meta-models. Schobbens et al. [46] presented a generic formal semantics for feature diagrams based on an analysis of existing feature diagram variants. The Common Variability Language (CVL) by Haugen et al. [30] aimed to create a common meta-model and standard. Sundermann et al. [51] presented the Universal Variability Language (UVL) as a community effort of the MODEVAR initiative [8]. A language specification at different levels was proposed by Thüm et al. [53] and adopted by Horcas et al. [32] for their extensible and modular abstract syntax for feature models. Hinterreiter et al. [31] presented a common programming interface for feature models to support evolution. All these works show that, despite different approaches, researchers aim to deal with the plethora of existing variability modeling approaches. Our work can be an important input for this goal.

Several researchers have been working on *variability model transformations*. Batory [6] and Czarnecki and Wąsowski [18] have translated feature models into propositional formulas or binary decision diagrams for automated analysis. Andersen et al. [3] transformed product comparison matrices into feature models. Roos Frantz et al. [43] transformed feature models to OVMs to enhance the interoperability among tools. Several variability modeling tools provide *import/export capabilities* [5]. FeatureIDE [37] supports import and export to diverse other feature modeling tools such as S.P.L.O.T [38]. Galindo et al. [27] proposed a service-based approach to provide one common configuration tool interface for multiple (types of) variability models. Knüppel et al. [35] developed an algorithm that eliminates complex cross-tree constraints in a feature model, e.g., to support export and import into feature modeling tools without capabilities to deal with such complex constraints. All these works highlight the need for transforming variability models and their import/export, but often neglect the information lost during the transformation/import/export. We address this gap by defining information loss classes applicable to multiple types of variability modeling approaches.

8 CONCLUSION

In this paper, we analyzed four variability modeling approaches, presented mapping tables between approach concepts and classified the lost information during one-way and roundtrip transformations using four information loss classes. We inspected each variability modeling approach concept and described how certain losses can be mitigated during transformations. We studied the applicability of our classes by transforming 40 variability models of different types into each other. We manually inspected each transformed variability model and counted identified information loss. We found that our information loss classes classify the lost information of the transformed models and indicate how information is lost during transformation. The classification can help modelers to identify improvements for existing transformation implementations. Additionally, we have shown that two languages (FeatureIDE and OVM) can be transformed to UVL and back without information loss. For DOPLER decision models, this is also possible if only a subset of the language, without specific concepts such as Number decisions, is used, or if *normal forms* and/or *meta-data* are used.

For future work, we will extend our evaluation to additional variability modeling approaches and models. We plan to investigate the usefulness of our information loss classes and the presented mapping tables by performing a study with modelers working on real-world variability models. Further, we aim to investigate and evaluate the proposed but yet not implemented encodings (e.g., for cardinalities) between feature models and decision models.

ACKNOWLEDGMENTS

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. This work has been partially supported by the German Research Foundation within the project *VariantSync* (TH 2387/1-1).

REFERENCES

[1] 1985. IEEE Standard for Binary Floating-Point Arithmetic. *ANSI/IEEE Std 754-1985* (1985), 1–20. https://doi.org/10.1109/IEEESTD.1985.82928

[2] Mathieu Acher, Patrick Heymans, Philippe Collet, Clément Quinton, Philippe Lahire, and Philippe Merle. 2012. Feature Model Differences. In *CAiSE*. Springer, 629–645.

[3] Nele Andersen, Krzysztof Czarnecki, Steven She, and Andrzej Wąsowski. 2012. Efficient Synthesis of Feature Models. In *Proc. of the 16th International Software Product Line Conference - Volume 1*. ACM, 106–115.

[4] Taslim Arif, Frank de Boer, Michiel Helvensteijn, Karina Villela, and Peter Wong. 2012. Evaluation of Modeling. Deliverable 5.3 of project FP7-231620 (HATS). Website. http://www.hats-project.eu/sites/default/files/Deliverable5.3.pdf#page=55.

[5] Rabih Bashroush, Muhammad Garba, Rick Rabiser, Iris Groher, and Goetz Botterweck. 2017. Case tool support for variability management in software product lines. *ACM Computing Surveys (CSUR)* 50, 1 (2017), 14:1–14:45.

[6] Don Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Software Product Lines*, Henk Obbink and Klaus Pohl (Eds.). Springer, 7–20.

[7] Maurice H. ter Beek, Klaus Schmid, and Holger Eichelberger. 2019. Textual Variability Modeling Languages: An Overview and Considerations. In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B (SPLC '19)*. ACM, New York, NY, USA, 151–157.

[8] David Benavides, Rick Rabiser, Don Batory, and Mathieu Acher. 2019. First International Workshop on Languages for Modelling Variability (MODEVAR 2019). In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A (SPLC '19)*. ACM, New York, NY, USA, 323.

[9] David Benavides, Pablo Trinidad, Antonio Ruiz-Cortés, and Sergio Segura. 2013. *FaMa*. Springer Berlin Heidelberg, Berlin, Heidelberg, 163–171. https://doi.org/10.1007/978-3-642-36583-6_11

[10] Thorsten Berger, Daniela Lettner, Julia Rubin, Paul Grünbacher, Adeline Silva, Martin Becker, Marsha Chechik, and Krzysztof Czarnecki. 2015. What is a feature?: a qualitative study of features in industrial software product lines. In *Proc. of the 19th International Software Product Line Conference*. ACM, 16–25.

[11] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A survey of variability modeling in industrial practice. In *Proc. of the 7th International Workshop on Variability Modelling of Software-intensive Systems*. ACM, 7–14.

[12] Thorsten Berger, Jan-Philipp Steghöfer, Tewfik Ziadi, Jacques Robin, and Jabier Martinez. 2020. The state of adoption and the challenges of systematic variability management in industry. *Empirical Software Engineering* 25, 3 (2020).

[13] Paul Maximilian Bittner, Thomas Thüm, and Ina Schaefer. 2019. SAT Encodings of the At-Most-k Constraint. In *Software Engineering and Formal Methods*, Peter Csaba Ölveczky and Gwen Salaün (Eds.). Springer International Publishing, Cham, 127–144.

[14] Lianping Chen and Muhammad Ali Babar. 2011. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology* 53, 4 (2011), 344–362.

[15] Software Productivity Consortium. 1991. *Synthesis Guidebook*. Technical Report. SPC-91122-MC. Herndon, Virginia: Software Productivity Consortium.

[16] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *Proc. of the 6th International Workshop on Variability Modeling of Software-Intensive Systems*. ACM, 173–182.

[17] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. 2005. Formalizing cardinality-based feature models and their specialization. *Software process: Improvement and practice* 10, 1 (2005), 7–29.

[18] Krzysztof Czarnecki and Andrzej Wąsowski. 2007. Feature Diagrams and Logics: There and Back Again. In *Proc. of the 11th International Software Product Line Conference*. IEEE, 23–34.

[19] Deepak Dhungana, Paul Grünbacher, and Rick Rabiser. 2011. The DOPLER Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study. *Automated Software Engineering* 18, 1 (2011), 77–114.

[20] Holger Eichelberger, Christian Kröher, and Klaus Schmid. 2013. An Analysis of Variability Modeling Concepts: Expressiveness vs. Analyzability. In *Safe and Secure Software Reuse*, John Favaro and Maurizio Morisio (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 32–48.

[21] Holger Eichelberger and Klaus Schmid. 2013. A systematic analysis of textual variability modeling languages. In *Proc. of the 17th International Software Product Line Conference*. ACM, 12–21.

[22] Sascha El-Sharkawy, Stephan Dederichs, and Klaus Schmid. 2012. From Feature Models to Decision Models and Back Again: An Analysis Based on Formal Transformations. In *Proc. of the 16th International Software Product Line Conference*. ACM, 126–135.

[23] Kevin Feichtinger, Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2020. Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain. In *Proc. of the 3rd International Workshop on Variability and Evolution of Software-Intensive Systems (VariVolution), co-located with SPLC 2020*. ACM.

[24] Kevin Feichtinger and Rick Rabiser. 2020. Towards Transforming Variability Models: Usage Scenarios, Required Capabilities and Challenges. In *3rd International Workshop on Languages for Modelling Variability (MODEVAR), co-located with SPLC 2020*. ACM, Montréal, Canada.

[25] Kevin Feichtinger, Johann Stöbich, Dario Romano, and Rick Rabiser. 2021. TRAVART: An Approach for Transforming Variability Models. In *15th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS'21)*. ACM, New York, NY, USA, Article 8, 10 pages.

[26] Michael Felderer and Guilherme Horta Travassos (Eds.). 2020. *Contemporary Empirical Methods in Software Engineering*. Springer. https://doi.org/10.1007/978-3-030-32489-6

[27] José A Galindo, Deepak Dhungana, Rick Rabiser, David Benavides, Goetz Botterweck, and Paul Grünbacher. 2015. Supporting distributed product configuration by integrating heterogeneous variability modeling approaches. *Information and Software Technology* 62 (2015), 78–100.

[28] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou. 2013. Variability in software systems-a systematic literature review. *IEEE Transactions on Software Engineering* 40, 3 (2013), 282–306.

[29] Hassan Gomaa. 2005. *Designing software product lines with UML*. IEEE.

[30] Øystein Haugen, Andrzej Wąsowski, and Krzysztof Czarnecki. 2013. CVL: common variability language. In *Proc. of the 17th International Software Product Line Conference*. ACM, 277–277.

[31] Daniel Hinterreiter, Michael Nieke, Lukas Linsbauer, Christoph Seidl, Herbert Prähofer, and Paul Grünbacher. 2019. Harmonized Temporal Feature Modeling to Uniformly Perform, Track, Analyze, and Replay Software Product Line Evolution. In *Proc. of the 18th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*. ACM, 115–128.

[32] Jose-Miguel Horcas, Mónica Pinto, and Lidia Fuentes. 2020. Extensible and Modular Abstract Syntax for Feature Modeling Based on Language Constructs. In *Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A - Volume A (SPLC '20)*. ACM, New York, NY, USA, Article 10, 7 pages.

[33] ISO/IEC 26558:2017(en) 2017. *Software and systems engineering — Methods and tools for variability modelling in software and systems product line*. Standard. International Organization for Standardization/International Electrotechnical Commission, Geneva, CH. https://www.iso.org/obp/ui/#iso:std:iso-iec:26558:ed-1:v1:en

[34] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. 1990. *Feature-oriented domain analysis (FODA) feasibility study*. Technical Report. Carnegie-Mellon Univ., Pittsburgh, Pa, Software Engineering Inst.

[35] Alexander Knüppel, Thomas Thüm, Stephan Mennicke, Jens Meinicke, and Ina Schaefer. 2017. Is There a Mismatch between Real-World Feature Models and Product-Line Research?. In *Proc. of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 291–302.

[36] Jabier Martinez, Wesley KG Assunção, and Tewfik Ziadi. 2017. ESPLA: A catalog of Extractive SPL Adoption case studies. In *Proc. of the 21st International Systems and Software Product Line Conference*. ACM, 38–41.

[37] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability with FeatureIDE*. Springer.

[38] Marcilio Mendonca, Moises Branco, and Donald Cowan. 2009. S.P.L.O.T.: Software Product Lines Online Tools. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*. ACM, New York, NY, USA.

[39] Daniel-Jesus Munoz, Jeho Oh, Mónica Pinto, Lidia Fuentes, and Don Batory. 2019. Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features. In *SPLC*. ACM, 289–301.

[40] Klaus Pohl, Günter Böckle, and Frank J van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer Science & Business Media.

[41] Mikko Raatikainen, Juha Tiihonen, and Tomi Männistö. 2019. Software product lines and variability modeling: A tertiary study. *Journal of Systems and Software* 149 (2019), 485–510.

[42] Dario Romano, Kevin Feichtinger, Danilo Beuche, Uwe Ryssel, and Rick Rabiser. 2022. Bridging the Gap between Academia and Industry: Transforming the Universal Variability Language to pure::variants and Back. In *Proc. of the 5th International Workshop on Languages for Modelling Variability (MODEVAR), co-located with SPLC 2022*. ACM.

[43] Fabricia Roos Frantz, David Felipe Benavides Cuevas, and Antonio Ruiz Cortés. 2009. Feature model to orthogonal variability model transformation towards interoperability between tools. In *Knowledge Industry Survival Strategy Initiative, Kiss Workshop (ASE 2009), Auckland, New Zealand*.

[44] Klaus Schmid, Rick Rabiser, and Paul Grünbacher. 2011. A comparison of decision modeling approaches in product lines. In *Proc. of the 5th International Workshop on Variability Modelling of Software-Intensive Systems*. ACM, 119–126.

[45] Pierre-Yves Schobbens, Patrick Heymans, and Jean-Christophe Trigaux. 2006. Feature diagrams: A survey and a formal semantics. In *Proc. of the 14th IEEE International Requirements Engineering Conference*. IEEE, 139–148.

[46] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. 2007. Generic semantics of feature diagrams. *Computer Networks* 51,

2 (2007), 456–479.

[47] Michael Schulze and Robert Hellebrand. 2015. Variability Exchange Language-A Generic Exchange Format for Variability Data.. In *Software Engineering (Workshops)*. 71–80.

[48] Christoph Seidl, Tim Winkelmann, and Ina Schaefer. 2016. A software product line of feature modeling notations and cross-tree constraint languages. In *Modellierung 2016*, Andreas Oberweis and Ralf Reussner (Eds.). Gesellschaft für Informatik e.V., Bonn, 157–172.

[49] Samuel Sepúlveda, Carlos Cares, and Cristina Cachero. 2012. Towards a unified feature metamodel: A systematic comparison of feature languages. In *Proc. of the 7th Iberian Conference on Information Systems and Technologies*. 1–7.

[50] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wąsowski, and Krzysztof Czarnecki. 2010. The Variability Model of The Linux Kernel. In *Proc. of the 5th International Workshop on Variability Modelling of Software-intensive Systems*. ACM, 45–51.

[51] Chico Sundermann, Kevin Feichtinger, Dominik Engelhardt, Rick Rabiser, and Thomas Thüm. 2021. Yet Another Textual Variability Language? A Community Effort Towards a Unified Language. In *Proc. of the 25th International Systems and Software Product Line Conference*. ACM, Leicester, United Kingdom.

[52] Thomas Thüm, Don Batory, and Christian Kästner. 2009. Reasoning About Edits to Feature Models. In *ICSE*. IEEE, 254–264.

[53] Thomas Thüm, Christoph Seidl, and Ina Schaefer. 2019. On Language Levels for Feature Modeling Notations. In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B (SPLC '19)*. ACM, New York, NY, USA, 158–161.

[54] Bart Veer and John Dallaway. 2011. The eCos Component Writer's Guide. Manual, available online at http://www.gaisler.com/doc/ecos-2.0-cdl-guide-a4.pdf.