Comprehensibility of Orthogonal Variability Modeling Languages: The Cases of CVL and OVM

Iris Reinhartz-Berger
Department of Information Systems
University of Haifa, Israel
iris@is.haifa.ac.il

Kathrin Figl
Institute for Information Systems &
New Media, Vienna, Austria
kathrin.figl@wu.ac.at

ABSTRACT

As the complexity and variety of systems and software products have increased, the ability to manage their variability effectively and efficiently became crucial. To this end, variability can be specified either as an integral part of the development artifacts or in a separate orthogonal variability model. Lately, orthogonal variability models attract a lot of attention due to the fact that they do not require changing the complexity of the development artifacts and can be used in conjunction with different development artifacts. Despite this attention and to the best of our knowledge, no empirical study examined the comprehensibility of orthogonal variability models.

In this work, we conducted an exploratory experiment to examine potential comprehension problems in two common orthogonal variability modeling languages, namely, Common Variability Language (CVL) and Orthogonal Variability Model (OVM). We examined the comprehensibility of the variability models and their relations to the development artifacts for novice users. To measure comprehensibility we used comprehension score (i.e., percentage of correct solution), time spent to complete tasks, and participants' perception of difficulty of different model constructs. The results showed high comprehensibility of the variability models, but low comprehensibility of the relations between the variability models and the development artifacts. Although the comprehensibility of CVL and OVM was similar in terms of comprehension score and time spent to complete tasks, novice users perceived OVM as more difficult to comprehend.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications – *languages*; D.2.13 [**Software Engineering**]: Reusable Software – *domain engineering*

General Terms

Experimentation, Languages, Human Factors

Keywords

Variability analysis, Model Comprehension, Empirical Study, CVL, OVM

SPLC '14, September 15 - 19 2014, Florence, Italy

http://dx.doi.org/10.1145/2648511.2648516

1. INTRODUCTION

Software systems are an essential part of almost any business. Independently, their requirements increased and became more complex, raising variability management challenges. Variability can be specified either as an integral part of the development artifacts or in a separate orthogonal variability model [24]. The former way commonly yield annotation-based approaches, in which the development artifacts are marked (annotated) introducing variability-related aspects. Examples of such methods are presented in [8; 26; 37]. Among the shortcomings of this kind of modeling approaches, Pohl et al. [24] mention: (1) consistency problems arising from the fact that variability may be spread across different models; (2) difficulties to trace variability across different development stages; (3) increasing complexity of the development artifacts, which are commonly complex without introducing variability; (4) differences in the concepts used to define variability between different development artifacts; and (5) ambiguity in variability information.

To overcome the aforementioned shortcomings, orthogonal variability modeling promotes specifying variability in separate models which are linked to the development artifacts, termed *base models*. Two such languages are Orthogonal Variability Model (OVM) and Common Variability Language (CVL). OVM [24] aims at representing variability as first class models, through the concepts of variation point and variant. A *variation point* represents a variable item or a property of an item, while a *variant* defines different instances of the variable item or property. *Trace links* relate variability information to elements in the base models that are affected by the variability. OVM supports specifying the base models in a variety of languages, including natural languages and UML. CVL [10], a proposal for a standard submitted to Object Management Group (OMG), is a domain-independent language for specifying and resolving variability. It facilitates the specification and resolution of variability over base models specified in any Meta-Object Facility (MOF)-based language (such as UML and SysML). One of the main concepts in CVL is VSpec, which stands for variability specification. *VSpecs* are specifications of abstract variability and are similar to features in feature modeling. They are organized in trees representing logical constraints on their resolutions. The relationships between elements of the variability model and elements of the base model are specified via different types of variation points, e.g., *object existence*, which indicates that the existence of a particular object, link, or value in the base model is in question.

In both OVM and CVL, one can specify in variability models mandatory and optional elements, OR and XOR relations between elements, and constraints (e.g., "requires"/"implies" and "excludes" dependencies). In both languages, the variability

models are linked to base models. However, these languages differ in several aspects: (1) variability models in CVL are structured as trees, while variability models in OVM have no hierarchical tree structure; (2) CVL enables specifying common and variable aspects of software products, while OVM concentrates on variability modeling; (3) OVM diffrentiates between variation points and variants in the specification level, while CVL does it only when resolving variability; (4) the relationships between variability models and base models are specified as links in OVM and as objects in CVL; and (5) small differences in the concrete syntaxes of the languages exist.

Despite the attention that orthogonal variability modeling receives, there may be difficulties in understanding the different involved models, namely, variability models and base models, as well as the relations between the two types. To the best of our knowledge, no empirical studies analyze such difficulties, raising points for improving those languages. To fill this gap, the main aim of this study was to examine the cognitive difficulty of understanding orthogonal variability models. In particular, we conducted an exploratory experiment using OVM and CVL as examples of orthogonal variability modeling languages and examined the comprehensibility of the variability models and their relations to the base models in both languages. In both cases the base models were specified in standard UML class diagrams and, hence, the comprehensibility of the base models was left out of the experiment scope.

The paper proceeds as follows. Section 2 reviews related work. Section 3 elaborates on the experiment design and procedure, while Section 4 presents the analysis procedure and the results. Section 5 discusses the results and the threats to validity. Finally, Section 6 summarizes and points on future research directions.

2. RELATED WORK

Since orthogonal variability modeling is quite new, the literature about evaluating orthogonal variability models is reduced. Thus, we review in this section studies that compare to some extent variability modeling languages in general, including feature diagrams.

Several frameworks for evaluating, comparing, or classifying feature or variability modeling methods have been suggested. Istoan et al. [14], for example, primarily distinguish between methods that use a *single (unique) model* to represent both commonality and variability and methods that distinguish and keep the *variability model separate* from the base model. Methods in the first category may annotate the development artifacts by means of extension or combine a general, reusable variability meta-model with different domain metamodels. Methods in the second category specify the variability models using notations such as feature diagrams, decision models, CVL, and OVM.

Haugen et al. [11] propose a reference model for comparing feature modeling approaches. This model makes distinction between the *generic sphere*, which includes feature models and product line models, and the *specific sphere*, which includes feature selection and product models. Three approaches to system families modeling are compared based on this reference model: standard languages, annotations, and domain-specific languages.

Matinlassi [19] suggests an evaluation framework that is based on Normative Information Model-based Systems Analysis and Design (NIMSAD) [15]. According to this framework, there are four essential categories of elements for method evaluation: (1) *context*, including specific goals, product line aspects, application domains, and method inputs/outputs; (2) *user*, including target groups, motivation, needed skills, and guidance; (3) *contents*, including method structure, artifacts, architectural viewpoints, language, variability, and tool support; and (4) *validation*, including method maturity and architecture quality.

Heidenreich et al. [12] classify variability mapping methods, namely, methods that explicitly specify the relations between feature models and the models used to describe the details of the product line (base models). The primary classification is to declarative and operational methods. *Declarative methods* focus on the needed changes and not on how to perform them, while *operational methods* concentrate on how target models must be modified when specific features are selected or deselected. Using a case study, the paper further explores two languages: FeatureMapper, a representative of the declarative approach, and VML*, a representative of the operational approach.

Sinnema and Deelstra [32] claim that three aspects are important to engineers when applying variability modeling techniques: modeling (expressiveness), tool support, and (supporting) processes. Since only a few modeling approaches refer to recommended processes, the focus is on the first two aspects: (1) *modeling* – How are choices modeled? How are products modeled? Which abstractions are used to manage complexity? How are the constraints and quality attributes modeled? How are incompleteness and imprecision addressed?; and (2) *tools* – What are the supported views, their focuses and purposes? How is inconsistency prevented? How is configuration guided? Does the tool include an inference engine? How is the mapping of the decisions to actual product family artifacts done? Based on these questions, Sinnema and Deelstra compared six variability modeling techniques: CBFM, COVAMOF, VSL, ConIPF, Pure::Variants, and Koalish.

Several attempts have been made to compare feature modeling languages [2; 4; 13; 29]. These studies focus on the expressiveness of the compared languages or methods and their representation and support characteristics. Czarnecki et al. [2], for example, compared feature modeling and decision modeling along ten dimensions: applications, unit of variability (features vs. decisions), orthogonality, data types, hierarchy, dependencies and constraints, mapping to artifacts, binding time and mode, modularity, and tool aspects. They further showed how the main properties of feature modeling and decision modeling are reflected in three specific methods including an initial version of CVL.

Schobbens et al. [29] surveyed and compared seven feature diagram notations. These notations differ in their graph types (trees vs. directed acyclic graphs – DAG), the supported node types (e.g., cardinality support), the supported graphical constraint types (namely, “requires”, “excludes”, none, or both), and the supported textual constraint types (i.e., textual composition rules support). In a later work, Heymans et al. [13] evaluated the formal properties of feature diagram languages using Krogstie et al.’s semiotic quality framework [17] and Harel and Rumpe’s guidelines for defining formal visual languages [9]. The list of evaluation criteria included: (1) expressiveness: what can the language express? (2) embeddability: can the structure of a diagram be kept when translated to another language? and (3) succinctness: how big are the expressions of one and the same semantic object?

Djebbi and Salinesi [4] provided a comparative survey on four feature diagram languages for requirements variability modeling. The languages are compared according to a list of criteria that includes readability, simplicity and expressiveness, type distinction, documentation, dependencies, evolution, adaptability, scalability, support, unification, and standardizeability.

The above studies neglect usage aspects, such as comprehensibility and ease of learning. Comprehensibility is of special importance in modeling, as the abstract goal of modeling is to formally describe some aspects of the physical and social world around us for the purpose of *understanding* and *communication* [22]. Indeed, recent research has started to examine comprehensibility aspects of variability modeling languages. The work in [13], for example, looks into comprehensibility appropriateness, namely whether or not language users understand all possible statements of the language. Comprehensibility appropriateness is, however, handled subjectively through embeddability and succinctness. The work in [27; 28] compared the comprehensibility of CBFM [3], which is a feature-oriented language, and ADOM [26], which is a UML-based approach, according to commonality- and variability-related concepts, including mandatory vs. optional elements, constraints (dependencies), and variation points and variants.

Despite those initiatives, no studies have addressed so far comprehensibility of orthogonal variability modeling in general and CVL and OVM in particular. In addition, the research to date has focused on variability models alone, while no studies investigating the relations to development artifacts have been undertaken so far. Clearly, a deeper understanding of comprehensibility of orthogonal variability modeling including relations to base models is needed as a basis for designing and shaping modeling languages in this domain. In this paper, we therefore describe our research to identify difficulties in understanding orthogonal variability models and their relations to base models. Our motivation is to complement the previous research and examine specifically two common orthogonal variability modeling languages, namely CVL and OVM.

3. EXPERIMENT DESIGN AND PROCEDURE

3.1 Research Goal and Questions

The main goal of this paper is to develop an improved understanding of potential comprehensibility problems in orthogonal variability modeling. Specifically, we focus on comprehensibility of the main semantic constructs of variability modeling languages, namely, mandatory/optional elements, OR/XOR relations, and constraints ("requires"/"implies" and "excludes' dependencies), as well as the relations to base models. With 'semantic construct' we refer to the underlying meaning of modeling symbols – their content, as defined by the metamodel [21]. We are interested in identifying the semantic constructs that are difficult to understand and may lead to comprehension problems. Due to the lack of existing cognitive theories on comprehending such software variability aspects, we refrain from developing exact hypotheses, as it would not be helpful in such an exploratory setting [18]. Instead, we seek to answer the following research question:

RQ1: Are there differences in comprehension of basic semantic constructs of orthogonal variability modeling (mandatory/optional elements, OR/XOR relations, constraints, and relations to base models)?

The same semantic constructs are represented differently in various languages. The languages' visual notation defines different graphical symbols and composition rules to visualize the same underlying concepts. As visual notation is a relevant influence factor for comprehensibility of visual models [7; 21], it is not possible to gain reliable insights on the comprehension of represented semantic constructs, when considering only one visual notation. To be able to make general inferences on comprehensibility of the relevant semantic constructs in orthogonal variability modeling, we, therefore, use two different modeling languages in our experimental design: OVM and CVL. This allows us to additionally examine the following research question:

RQ2: Are there differences in comprehension of CVL and OVM models?

Comprehension difficulties that are common in both CVL and OVM may be attributed to orthogonal variability modeling in general, while difficulties that arise only in one language hint to potential comprehensibility problems of the notational design of the respective language. Thus, we are further interested in the interaction effects between the semantic construct type and the language.

To complement and extend our goal, we are additionally interested in the users' views and their evaluation. Specifically, we aim to assess how users subjectively rate the difficulty of the different types of models involved in orthogonal variability modeling and how they rate their preferences concerning the modeling language (CVL and OVM). Accordingly, we phrased the following research questions:

RQ3a: Are there differences in users' perception of the difficulty of the three types of models involved in orthogonal variability modeling (variability models, base models, and their relations)?

RQ3b: Are there differences in users' perception of the difficulty to use, comprehend, and learn CVL and OVM?

To answer our research questions we used a *randomized experimental* design. We used two different experimental groups, in which participants got two models of different application domains in the two modeling languages. Our design ensured that each participant answered comprehension questions related to the main semantic constructs and targeting model elements on a CVL model as well as on an OVM model. The exact procedure is explained in Section 3.4. The main *independent variables* in our research design were the *modeling language* and the *type of semantic construct*. The *dependent variables* were *comprehension score* (measured using the percentage of correct solution), *time spent to complete tasks*, and *user's perception of difficulty*.

3.2 Experimental Material

To enable each participant to experience both modeling languages, we constructed two models in different application domains. The models were similar in complexity (in terms of the number of elements) and in the examined model elements and semantic constructs. The first application domain was of *mobile phones*, including features referring to media, display, connectivity, and sensors. The second application domain was of

smart homes, including features referring to security settings, alarm, light management, and air-conditioners. In each application domain, an OVM model and a CVL model were built, preserving their informational equivalence. Thus, the *objects* of the experiment were four models in two application domains. Each model included two parts: a variability model and a model depicting the relations between part of the variability model and a base model specified in a UML class diagram. Due to space limitations, only the models depicting the relations between the variability and base models in the mobile phones application domain are presented in the appendix.

3.3 Measurement of Comprehension

The comprehension tasks were embedded within an online questionnaire. On each model 19 questions were asked. We constructed the questions so that it was necessary to understand a specific semantic construct for answering each question. 14 questions examined whether specific configurations are allowed in the application domain according to the variability model. Of these questions, 6 questions were related to optional and mandatory elements, 6 to OR and XOR relations and 2 to constraints. Further 5 questions examined valid configuration designs based on the relations between the variability model and the base model.

All the questions in the questionnaire can be described as surface-level tasks which measure comprehension of models more directly than deep-level tasks that require participants to work with the models in a usage context [23].

The participants were presented with a model and one question at a time (the questions were presented in the same order for each model). The participants had to choose for each question between the following answers: Correct, Wrong, Cannot be answered from model, I don't know. After answering a question, the participant proceeded to the next question, but could not return to previous questions. This way we could accurately measure the time needed to answer an individual question.

We ensured that the wording of questions was comparable and therefore each question started with "can". Examples of questions used in the experiment for the mobile phone domain and their categorization are:

1. Can a mobile phone have no sensors? (optional element)
2. Can a mobile phone with sensors have no accelerometer? (mandatory element)
3. Can a mobile phone with mp4 have both download and stream capabilities? (OR relation)
4. Can a mobile phone with non-touchscreen have neither front keyboard nor hidden keyboard? (XOR relation)
5. Can a mobile phone have USB, but no camera and download (of mp4)? (constraint)
6. Can a mobile phone design include the classes USB Info and MP4 Info with download method, but without Camera Info class? (relations to base model)

3.4 Procedures

The participants were randomly divided into two main experimental groups, as described in Table 1. Each participant got the models of the two application domains, but in different modeling languages. In addition, we counterbalanced the orders of the models to control for possible learning and fatigue effects. For instance, about half of the participants in the first experimental group got the models in the following order: a CVL model of mobile phones followed by an OVM model of smart homes, while the other half got the same models in the opposite order.

Table 1. The experimental groups

Group	Mobile Phones	Smart Homes	Order	No of participants
1	CVL	OVM	mobile-CVL smart-OVM	12
			smart-OVM mobile-CVL	10
2	OVM	CVL	smart-CVL mobile-OVM	12
			mobile-OVM smart-CVL	11

The participants were requested to open the online questionnaire, which was divided into four parts: a pre-questionnaire, Part A (questions on the first model and post-evaluation), Part B (questions on the second model and post-evaluation), and a post-experiment questionnaire.

The pre-questionnaire obtained general information about the participants and their background, including age, gender, degree and subject of studies, and familiarity with the application domains (mobile phones and smart homes). As the base models were specified in class diagrams, we asked in the pre-questionnaire about familiarity with class diagrams and knowledge of class diagrams. To measure (self-rated) familiarity with class diagrams, we adopted the three-item modeling grammar familiarity scale of Recker [25]: (1) Overall, I am very familiar with class diagrams; (2) I feel very confident in understanding class diagrams; (3) I feel very competent in modeling class diagrams. We further objectively examined the prior knowledge of the participants in modeling class diagrams through three comprehension questions on a simple class diagram. Each question presented a statement and four possible answers: Correct, Wrong, Cannot be answered from model, I don't know.

After filling the pre-questionnaire, the participants were sequentially presented with two parts. In each part slides explaining and exemplifying the modeling language concepts were presented. The number of slides, their subjects, and the used examples were similar for the two modeling languages. The participants also got hard-copies of these slides which they could consult while answering the questions. The participants had to study the modeling language on their own from the slides and proceed to the model and its questions. The time spent on each question was recorded by the online questionnaire. No rigid time constraints were imposed on the participants.

After completing each part, the participants had to fill a post-part questionnaire that collected feedback on the difficulty to understand the variability model (with 4 items asking about mandatory and optional elements, OR and XOR relations), the base model (with 3 items asking about the base model in general, classes and packages, associations), and the relations between these two models. The answering options ranged from 1=very easy to 7=very difficult.

Finally, after completing the two main parts of the questionnaire and experimenting with both modeling languages, the participants had to fill a post-experiment questionnaire with three single-choice items which required choosing the preferred modeling language (or selecting a neutral response option) in terms of usage, comprehension, and learning difficulties.

3.5 Participants

The participants were information systems students in their second year of studies. It has already been shown in [33] that students have a good understanding of the way industry behaves, and may work well as subjects in empirical studies in areas such as requirements engineering. Additionally, students are a relatively homogenous group concerning knowledge about and experience with conceptual modeling [31].

The experiment took place in the last week of the winter semester of the academic year 2013-14 in a course entitled "design and development of information systems", whose main focus was modeling. The students studied in that course modeling in ER, DFD, and UML, but were not exposed to software product line engineering or variability modeling. They had homework to practice their capabilities in the different modeling languages. To assure sufficient motivation, the participants received up to 5 points bonus to their course grades depending on their achievements in the experiment. A total of 45 students participated in the study (22 and 23 per experimental group): 26 males (58%) and 19 females (42%) with a mean age of 24 years.

4. RESULTS

4.1 Comprehension Tasks

To answer the two first research questions, we performed for each dependent variable two mixed-design analyses of covariance (ANCOVA) with four factors: (1) semantic construct type (optional and mandatory elements, OR and XOR relations, constraints, relations to the base model) as a within-subjects factor; (2) modeling language (OVM, CVL) as a between-subjects factor; (3) application domain (mobile phone, smart home) as a between-subjects factor; and (4) model order (first model, second model) as a between-subjects factor. As dependent variables, we used the mean percentage of correct answers (for each model) in the first analysis and the mean time to complete the tasks in the second analysis. The 'semantic construct type' factor relates to RQ1, while the 'modeling language' factor relates to RQ2. We included the factors 'application domain' and 'model order' in the analyses to control for possible domain knowledge, learning, and fatigue effects. Prior to our analyses, we also checked whether the control variables 'familiarity with class diagrams' and 'knowledge of class diagrams', which were collected in the pre-questionnaire as described in Section 3.4, had an influence on comprehension and on time to perform the tasks. Since 'familiarity with class diagrams' did not have an influence, we decided to drop it from the final statistical tests we report. 'Knowledge of class diagrams' had a significant effect on time to perform the tasks, but not on the comprehension score; therefore we included this variable as covariate in the ANCOVA for time.

Where the assumption of sphericity [64] was violated for within-subject effects, we report the Greenhouse-Geisser [65] corrected test value. In general, we report all significant effects below $p \leq .05$.

4.1.1 Comprehension Score

We first turn to the results concerning semantic construct type (RQ1). There was a significant main effect of semantic construct type on comprehension ($F_{df=2.57,249}=70.65$, p=.000; see Figure 1-Figure 3). Questions related to optional and mandatory elements were the easiest to comprehend and correctly answer (93% mean solution rate), followed by questions related to OR/XOR relations (85%) and constraints (82%). Questions referring to relations to the base model were the most difficult ones (54% correctness). To find out the comprehensibility of which semantic construct types significantly differed from each other, we calculated pairwise comparisons using Fisher's Least Significant Difference (LSD) test. All differences between semantic construct types were significant ($p \leq .01$), with the exception of the difference between OR/XOR relations and constraints (these constructs did not significantly differ).

Regarding the effect of the application domain, there was no significant effect on comprehension score. However, there was a significant interaction effect of application domain and semantic construct type ($F_{2.57,249}=6.46$, p=.001). From the data in Figure 1, we observe that comprehension difficulties concerning the relations to base models were pronounced more strongly in the smart home application domain than in the mobile phone application domain. This could possibly be explained by the fact that participants were more familiar with mobile phones (Mean=3.87 on a 5-point scale from 1=very low to 5=very high, SD=.89) than with smart homes (Mean=1.98, SD=.97; $t_{44}=-11.87$, p=.000). In addition, the three-way interaction effect between semantic construct type * language * application domain was significant ($F_{2.57,249}=3.85$, p=.01). We suppose that this is due to the fact that the relations to the base model were most difficult to interpret in the CVL model of smart homes (34%), while solution rates varied between 57% and 65% in the other combinations.

The model order had a significant influence on the comprehension score ($F_{df=1,83}=4.73$, p=.03, see Figure 2). As would be expected, participants performed better on the second model (82%) than on the first model (76%).

Turning now to the experimental evidence on the modeling language (RQ2), results revealed that there were no significant differences between OVM (80% solution rate) and CVL (77% solution rate) in comprehension. Figure 3 demonstrates no significant differences in the semantic construct level either.

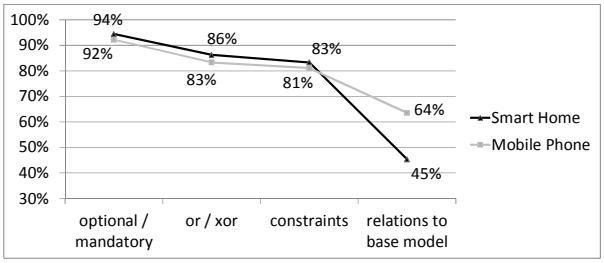

Figure 1. Comprehension of semantic construct types with respect to application domain

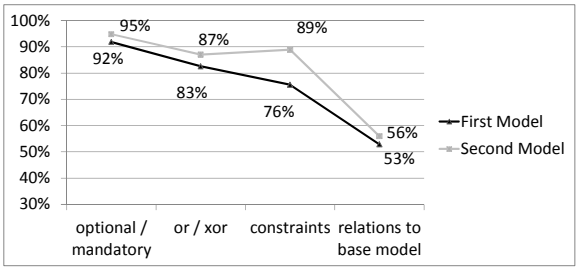

Figure 2. Comprehension of semantic construct types with respect to model order

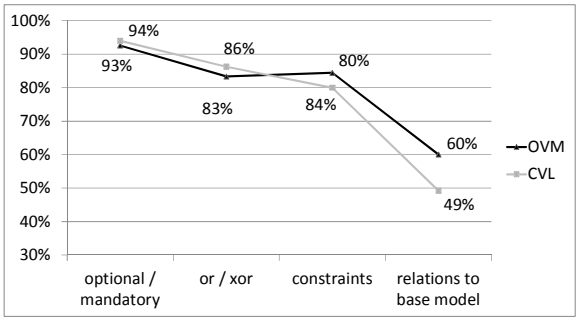

Figure 3. Comprehension of semantic construct types with respect to modeling language

4.1.2 Comprehension Time

Results showed that there were three significant influence factors on time to complete the comprehension tasks: semantic construct type ($F_{df=1.49,246}$=6.26, p=.000), model order ($F_{df=1,82}$=19.23, p=.000) and familiarity with class diagrams ($F_{df=1,82}$=7.49, p=.008). As we can observe from Figure 4, semantic construct type did affect comprehension time. Participants spent most time to solve questions on relations to base models (109 seconds per question on average), followed by OR/XOR relations (51 sec.), constraints (44 sec.) and optional/mandatory elements (39 sec.). All pairwise comparisons using Fisher's LSD test for semantic construct types were significant (p≤.03). In addition, participants spent significantly more time to solve tasks in the first model (69 sec.) than in the second (53 sec.). Participants with higher knowledge on class diagrams spent more time to solve tasks.

There was no significant effect of language on comprehension time. On average, participants spent 62 seconds in OVM and 59 seconds in CVL to solve a comprehension task.

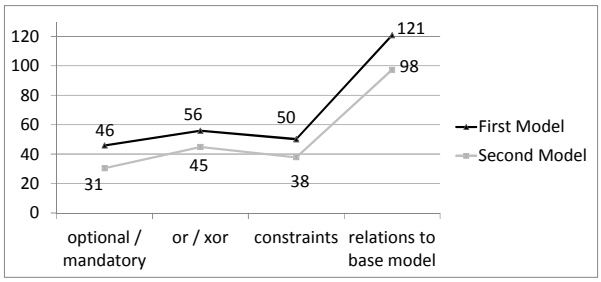

Figure 4. Comprehension time of semantic construct types with respect to model order (y-axis: seconds)

4.2 Users' Perception and Preference

As noted, we complement our objective tests on comprehension with subjective tests of user perception on difficulty of the different model types and preference of the modeling languages. This section reports our results regarding RQ3.

4.2.1 Perceived Comprehension Difficulties

The questions in the post-part evaluation referred to the perceived difficulty of elements of the variability model, the base model, and the relations between the two models. Figure 5 presents the results regarding users' perception of difficulty (with 1 – very easy and 7 – very difficult) as response to RQ3a.

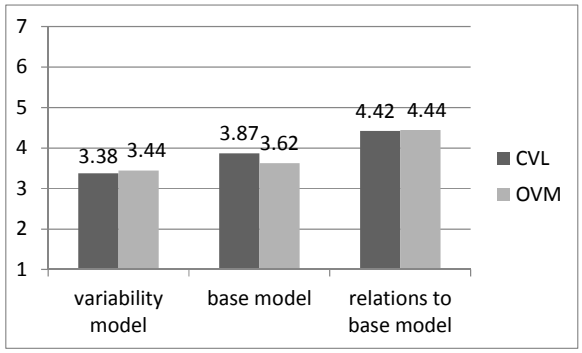

Figure 5. Users' perception of difficulty according to model type

A repeated measures analysis of variance with the mean difficulties ratings of both languages as dependent variables showed that the type of model (variability model, base model, relations to base model) was significant *($F_{1.8,135}$=23.63, p=.000)*. Pairwise comparisons using LSD test showed that all differences were significant. As can be seen from Figure 5, variability models were perceived to be of medium difficulty, easier than or almost in the same level of difficulty as base models, with which the participants had previous familiarity and knowledge. The relations between the two types of models were more difficult to understand. According to a t-test for paired samples with the perceived comprehension difficulty of the variability model there was no difference in the mean rating for OVM and CVL.

4.2.2 Users' Preferences of the Modeling Language

Finally, we report on preferential choice of the modeling languages in response to RQ3b (see Figure 6). We performed one-sample t-test between proportions to determine whether there was a significant difference between the percent choosing CVL and OVM. Results revealed non-significant trends that more users prefer CVL over OVM (t_{44}=1.78; p=.08), and rate the learning difficulty of OVM higher (t_{44}=1.77; p=.08). Significantly more users rated the model comprehensibility better for CVL than for OVM (t_{44}=2.78; p=.007).

5. DISCUSSION AND THREATS TO VALIDITY

The objective of this study was to examine comprehensibility of orthogonal variability modeling. We identify a number of interesting results. First, we found that the relations to elements of the base model were the most difficult semantic construct. The most likely explanation for the high difficulty of this construct is that users have to assimilate different pieces of information from two models (the variability model and the base model) simultaneously and cognitively integrate them. This may lead to high cognitive load for users because of a split-attention effect [38]. Our result is in line with prior research which has demonstrated users' difficulties in navigating and relating information items from multiple diagrams [16; 20]. We therefore encourage the use of appropriate visual cues to show which model elements belong to each other to support users' cognitive integration processes of the information from two different visual models [16].

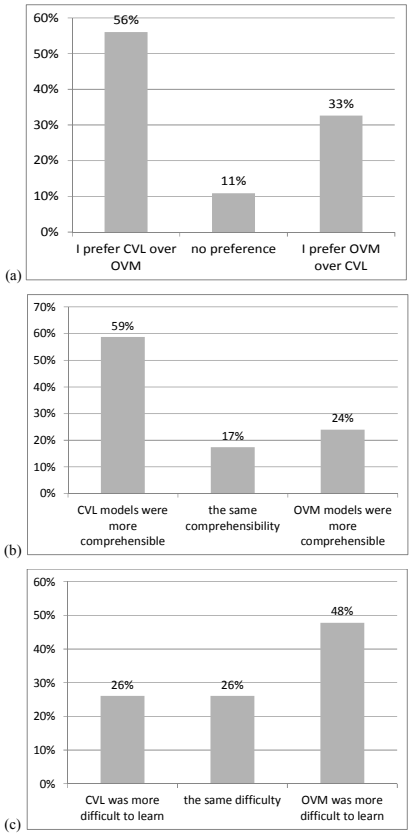

Figure 6. Users' preference of the modeling languages in terms of (a) usage, (b) comprehension, (c) difficulty to learn

Next, we like to discuss why questions related to optional/mandatory elements were the easiest to answer. This result could have been caused by the fact that these resemble binary relations, while OR/XOR relations involve at least three elements. Based on the cognitive load theory [34], a higher amount of elements that need to be paid attention leads to higher cognitive load resulting in higher comprehension difficulty. This result is also in agreement with findings in other modeling areas which showed that comprehensibility is lower if attention has to be divided between a higher number of modeling elements for solving a specific task [6].

It is somewhat surprising that we did not find significant differences in comprehension between the two modeling languages investigated CVL and OVM. The most likely explanation for this finding is that the two languages use similar symbols (rectangles and triangles) and syntactic rules. However, user ratings showed a preference for CVL over OVM. Users also subjectively rated OVM harder to learn and CVL more comprehensible. This result may be explained by the differences in the visual model structure. CVL, which has a hierarchical, tree structure, was perceived as easier to use, comprehend, and learn. The common root node for all model elements might help the user to search the required information and to better distinguish the variability model from the base model. There are, however, other possible explanations. For instance, we notice that in OVM there are two different concepts, equipped with two different symbols (triangle and rectangle), for representing variability in the specification level: variation points and variants. In CVL, a single construct is used – VSpec – for representing abstract variability. Variation points and variants are used in CVL only when resolving variability. In addition, the names of variation points in the OVM models we used (those that are created with REMiDEMMI[1] – the OVM supporting tool) are placed next to the corresponding symbol, but not inside of it as in CVL. According to the Gestalt law of proximity [35] this makes it more difficult to recognize the elements as belonging together. While these notational differences are minor and did not result in measurable comprehension differences, they were large enough to influence the users' impression and intentions to use the modeling language. These findings are consistent with those of other studies which found that users' perception of criteria, such as graphic economy or combination of text and symbols, influence perceived usefulness of visual conceptual modeling languages [5].

Following the well-known classification of threats to validity [30; 36], the limitations of our experiment are acknowledged below.

External validity. One source of weakness is the use of student subjects. As already mentioned, using students as participants is acceptable in different software engineering areas. Moreover, the participants in our experiment had received training in modeling and, therefore, we do believe that they serve as an adequate proxy for future users of orthogonal variability modeling in general and OVM and CVL in particular.

Despite the clear support for research questions, the generalizability of findings reported here should be undertaken with caution, because we could only include two different application domains in the study and we selected two specific variability modeling languages – OVM and CVL. As the two application domains and their models included in the questionnaire were typical representatives we argue that they provided a reasonable test of comprehensibility despite their simplicity. The selection of the languages was done perceiving OVM and CVL as common variability modeling languages that are well known in the literature of software product line engineering and variability modeling.

Internal validity. This type of threats reflects whether observed differences can be attributed to the independent variables and aims to rule out potential alternative explanations. We chose to use an experiment as it affords higher internal validity than other methods [1]. In our repeated measures design we counterbalanced the order of the conditions and randomly assigned participants to the different treatments. Each of the four different models was presented equally often as first and as second model to the participants. As results in fact showed a significant learning effect from the first to the second model, we included the variable model order in the analysis to control for practice effects. We identified and measured potentially confounding factors on the individual level, such as familiarity with class diagrams and knowledge of class diagrams, and included them as control variables in the analyses. We created standardized slides for students to self-study the languages, so no influence of the lecturers' capabilities, knowledge, and opinions were introduced to the training. In addition, we tried to use the same structure and wordings in the slides for CVL and OVM to avoid a bias for one language.

Construct validity. This type of threats refers to the extent of which the operationalizations of the constructs actually measure

[1] Available at http://remidemmi.cdhq.de/

the constructs. We used a questionnaire to assess the comprehension level. In order to lower guessing probability in the true/false questions we used two additional answer options: Cannot be answered from model, I don't know.

In addition, the models used as experimental objects were checked by OVM and CVL experts, who were not involved in the research (as authors or researchers). Time was measured by the online questionnaire and required no human interventions.

Conclusion validity. We assured that random influences to the experimental setting were low. We used a homogenous group of participants who were committed to the experiment by giving course credit of up to 5 points (bonus) according to performance. This way we reduced variance in motivation and competence to answer the tasks correctly.

6. CONCLUSIONS AND FUTURE WORK

This study set out to identify comprehension difficulties in orthogonal variability modeling in general and to determine specific difficulties in CVL and OVM in particular. One of the more important findings to emerge from this study is that relations to base models are difficult to understand for users, while optional and mandatory elements are easy to understand. The difficulty to comprehend OR/XOR relations and constraints lay in the middle. This paper therefore encourages the exploration of alternative modeling strategies for visualizing relations to base models in orthogonal variability models.

The second major finding was that CVL and OVM did not differ concerning their comprehensibility. Both languages can be recommended to a similar extend. However, users subjectively rated CVL as more comprehensible than OVM, which might be due to some minor shortcomings of the visual notation of OVM. The findings from our study might inform ongoing revisions of CVL and OVM.

Several possible directions for future research emerge from our study. This experiment needs to be replicated in various forms with a larger variety of models and application domains to understand difficulties in comprehending semantic variability constructs in more detail. Opportunities exist for fellow scholars to examine comprehension difficulties in additional orthogonal variability modeling languages and compare them with difficulties in annotation-based approaches that support single models to capture both commonality and variability. Future studies could also extend this work and examine difficulties in modeling (and not just understanding) orthogonal variability models. Finally, we will strive for experiments in industrial settings. Looking ahead, further research in this field has the potential to guide developers in their ongoing design efforts and to significantly improve variability modeling in practice.

7. ACKNOWLEDGMENTS

The authors would like to thank Prof. Øystein Haugen, Vanessa Stricker, and Prof. Klaus Pohl for reviewing and commenting on the experiment's models and materials.

8. REFERENCES

[1] Cook, T.D. and Campbell, D.T. 1979. *Quasi-Experimentation: Design and Analysis Issues*. Houghton Mifflin, Boston, Massachusetts.

[2] Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., and Wąsowski, A. 2012. Cool features and tough decisions: a comparison of variability modeling approaches. In *Proceedings of the Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems* (Leipzig, Germany2012), ACM, 2110167, 173-182. DOI= http://dx.doi.org/10.1145/2110147.2110167.

[3] Czarnecki, K. and Kim, C.H.P. 2005. Cardinality-based feature modeling and constraints: a progress report. In *International Workshop on Software Factories at OOPSLA* ACM, San Diego, California, USA.

[4] Djebbi, O. and Salinesi, C. 2006. Criteria for Comparing Requirements Variability Modeling Notations for Product Lines. In *Workshops on Comparative Evaluation in Requirements Engineering*, S. Camille Ed., 20-35.

[5] Figl, K. and Derntl, M. 2011. The impact of perceived cognitive effectiveness on perceived usefulness of visual conceptual modeling languages. In *Proceedings of the Proceedings of the 30th international conference on Conceptual modeling* (Brussels, Belgium2011), Springer-Verlag, 2075154, 78-91.

[6] Figl, K. and Laue, R. 2011. Cognitive Complexity in Business Process Modeling. In *Advanced Information Systems Engineering*, H. Mouratidis and C. Rolland Eds. Springer Berlin / Heidelberg, 452-466. DOI= http://dx.doi.org/10.1007/978-3-642-21640-4_34.

[7] Figl, K., Mendling, J., and Strembeck, M. 2013. The Influence of Notational Deficiencies on Process Model Comprehension. *Journal of the Association for Information Systems 14*, 6.

[8] Gomaa, H. 2004. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison Wesley Longman Publishing Co.

[9] Harel, D. and Rumpe, B. 2004. Meaningful Modeling: What's the Semantics of "Semantics"? *Computer 37*, 10, 64-72. DOI= http://dx.doi.org/10.1109/mc.2004.172.

[10] Haugen, Ø. 2012. *Common Variability Language (CVL) – OMG Revised Submission. OMG document ad/2012-08-05.*

[11] Haugen, Ø., Møller-Pedersen, B., and Oldevik, J. 2005. Comparison of System Family Modeling Approaches. In *Software Product Lines*, H. Obbink and K. Pohl Eds. Springer Berlin Heidelberg, 102-112. DOI= http://dx.doi.org/10.1007/11554844_12.

[12] Heidenreich, F., Sánchez, P., Santos, J., Zschaler, S., Alférez, M., Araújo, J., Fuentes, L., Kulesza, U., Moreira, A., and Rashid, A. 2010. Relating Feature Models to Other Models of a Software Product Line. In *Transactions on Aspect-Oriented Software Development VII*, S. Katz, M. Mezini and J. Kienzle Eds. Springer Berlin Heidelberg, 69-114. DOI= http://dx.doi.org/10.1007/978-3-642-16086-8_3.

[13] Heymans, P., Schobbens, P.Y., Trigaux, J.C., Bontemps, Y., Matulevicius, R., and Classen, A. 2008. Evaluating formal properties of feature diagram languages. *Software, IET 2*, 3, 281-302. DOI= http://dx.doi.org/citeulike-article-id:3020746.

[14] Istoan, P., Klein, J., Perouin, G., and Jezequel, J.-M. 2011. A Metamodel-based Classification of Variability Modeling Approaches. In *VARiability for You Workshop*, 23–32.

[15] Jayaratna, N. 1994. *Understanding and Evaluating Methodologies: NIMSAD, a Systematic Framework*. McGraw-Hill, Inc.

[16] Kim, J., Hahn, J., and Hahn, H. 2000. How Do We Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning. *INFORMATION SYSTEMS RESEARCH 11*, 3, 284-303. DOI= http://dx.doi.org/http://dx.doi.org/10.1287/isre.11.3.284.12206.

[17] Krogstie, J., Sindre, G., and Jørgensen, H.D. 2006. Process Models Representing Knowledge for Action: a Revised Quality Framework. *European Journal of Information Systems 15*, 1, 91-102.

[18] Kumar, S. and Karoli, V. 2011. *Handbook Of Business Research Methods*. Thakur Publishers.

[19] Matinlassi, M. 2004. Comparison of software product line architecture design methods: COPA, FAST, FORM, KobrA and QADA. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, 127-136. DOI= http://dx.doi.org/10.1109/icse.2004.1317435.

[20] Moody, D.L. 2006. What Makes a Good Diagram? Improving the Cognitive Effectiveness of Diagrams in IS Development. In *15th International Conference on Information Systems Development (ISD 2006)*, Budapest, Hungary.

[21] Moody, D.L. 2009. The "Physics" of Notations: Towards a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 35*, 5, 756-779.

[22] Mylopoulos, J. 1992. Conceptual Modeling and Telos. In *Conceptual Modeling*, P. Loucopoulos and R. Zicari Eds. John Wiley and Sons, New York, 49-68.

[23] Parsons, J. and Cole, L. 2005. What do the Pictures mean? Guidelines for Experimental Evaluation of Representation Fidelity in Diagrammatical Conceptual Modeling Techniques. *Data and Knowledge Engineering 55*, 3.

[24] Pohl, K., Böckle, G., and Van Der Linden, F. 2005. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer.

[25] Recker, J. 2010. Continued Use of Process Modeling Grammars: The Impact of Individual Difference Factors. *European Journal of Information Systems 19*, 1, 76-92.

[26] Reinhartz-Berger, I. and Sturm, A. 2009. Utilizing domain models for application design and validation. *Inf. Softw. Technol. 51*, 8, 1275-1289. DOI= http://dx.doi.org/10.1016/j.infsof.2009.03.005.

[27] Reinhartz-Berger, I. and Tsoury, A. 2011. Experimenting with the Comprehension of Feature-Oriented and UML-Based Core Assets. In *Enterprise, Business-Process and Information Systems Modeling*, T. Halpin, S. Nurcan, J. Krogstie, P. Soffer, E. Proper, R. Schmidt and I. Bider Eds. Springer Berlin Heidelberg, 468-482. DOI= http://dx.doi.org/10.1007/978-3-642-21759-3_34.

[28] Reinhartz-Berger, I. and Tsoury, A. 2011. Specification and Utilization of Core Assets: Feature-Oriented vs. UML-Based Methods. In *Advances in Conceptual Modeling. Recent Developments and New Directions*, O. Troyer, C. Bauzer Medeiros, R. Billen, P. Hallot, A. Simitsis and H. Mingroot Eds. Springer Berlin Heidelberg, 302-311. DOI= http://dx.doi.org/10.1007/978-3-642-24574-9_38.

[29] Schobbens, P.-Y., Heymans, P., and Trigaux, J.-C. 2006. Feature Diagrams: A Survey and a Formal Semantics. In *Proceedings of the Proceedings of the 14th IEEE International Requirements Engineering Conference* (2006), IEEE Computer Society, 1174002, 136-145. DOI= http://dx.doi.org/10.1109/re.2006.23.

[30] Shadish, W., Cook, T., and Campbell, D. 2001. *Experimental and quasi-experimental designs for generalized causal inference*. Houghton Mifflin.

[31] Siau, K. and Loo, P.-P. 2006. Identifying Difficulties in Learning UML. *Information Systems Management 23*, 3, 43-51.

[32] Sinnema, M. and Deelstra, S. 2007. Classifying variability modeling techniques. *Information and Software Technology 49*, 7, 717-739. DOI= http://dx.doi.org/http://dx.doi.org/10.1016/j.infsof.2006.08.001.

[33] Svahnberg, M., Aurum, A., and Wohlin, C. 2008. Using students as subjects - an empirical evaluation. In *Proceedings of the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement* (Kaiserslautern, Germany2008), ACM, 1414055, 288-290. DOI= http://dx.doi.org/10.1145/1414004.1414055.

[34] Sweller, J. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive Science: A Multidisciplinary Journal 12*, 2, 257-285.

[35] Wertheimer, M. 1938. Laws of organization in perceptual forms. In *A sourcebook of Gestalt psychology*, W.D. Ellis Ed. Routledge and Kegan Paul, London, UK.

[36] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., and Wesslén, A. 2000. *Experimentation in Software Engineering – An Introduction*. Kluwer Academic Publishers.

[37] Ziadi, T. and Jezequel, J.-M. 2006. Software Product Line Engineering with the UML: Deriving Products. In *Software Product Lines*, T. Käköla and J. Duenas Eds. Springer Berlin Heidelberg, 557-588. DOI= http://dx.doi.org/10.1007/978-3-540-33253-4_15.

[38] Zugal, S., Pinggera, J., Weber, B., Mendling, J., and Reijers, H.A. 2012. Assessing the Impact of Hierarchy on Model - A Cognitive Perspective. In *EESSMod*.

Appendix: Mobile Phones Models

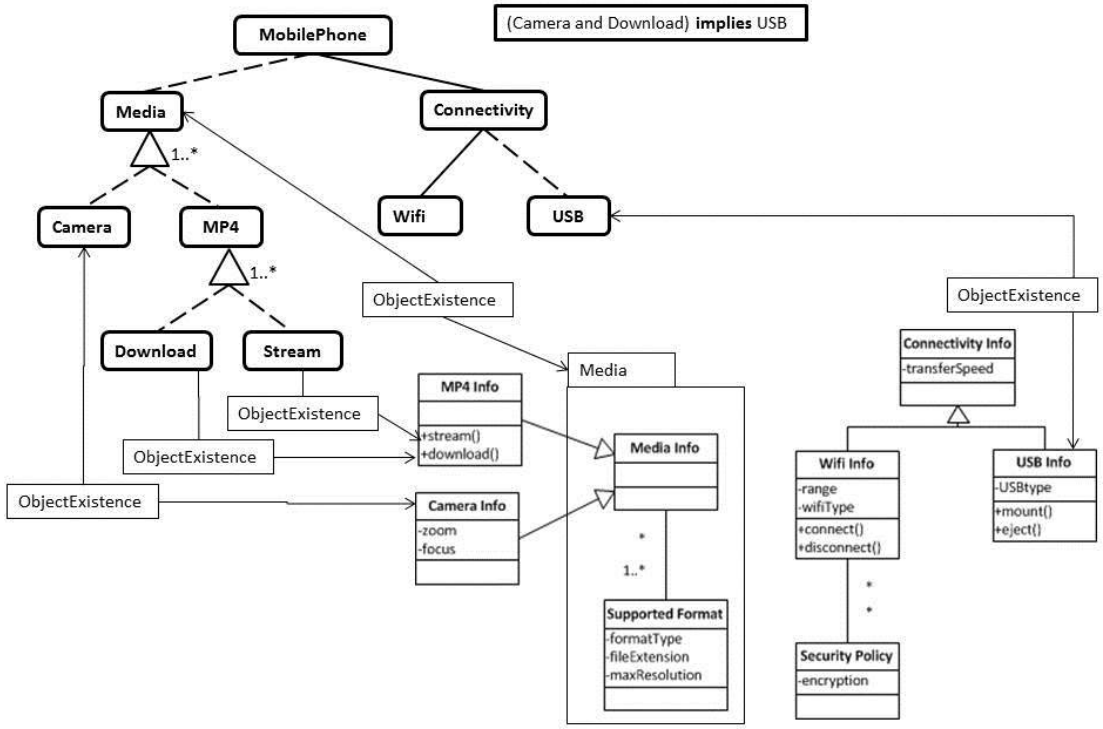

Figure 7. Part of the CVL model of mobile phones: relations to the base model

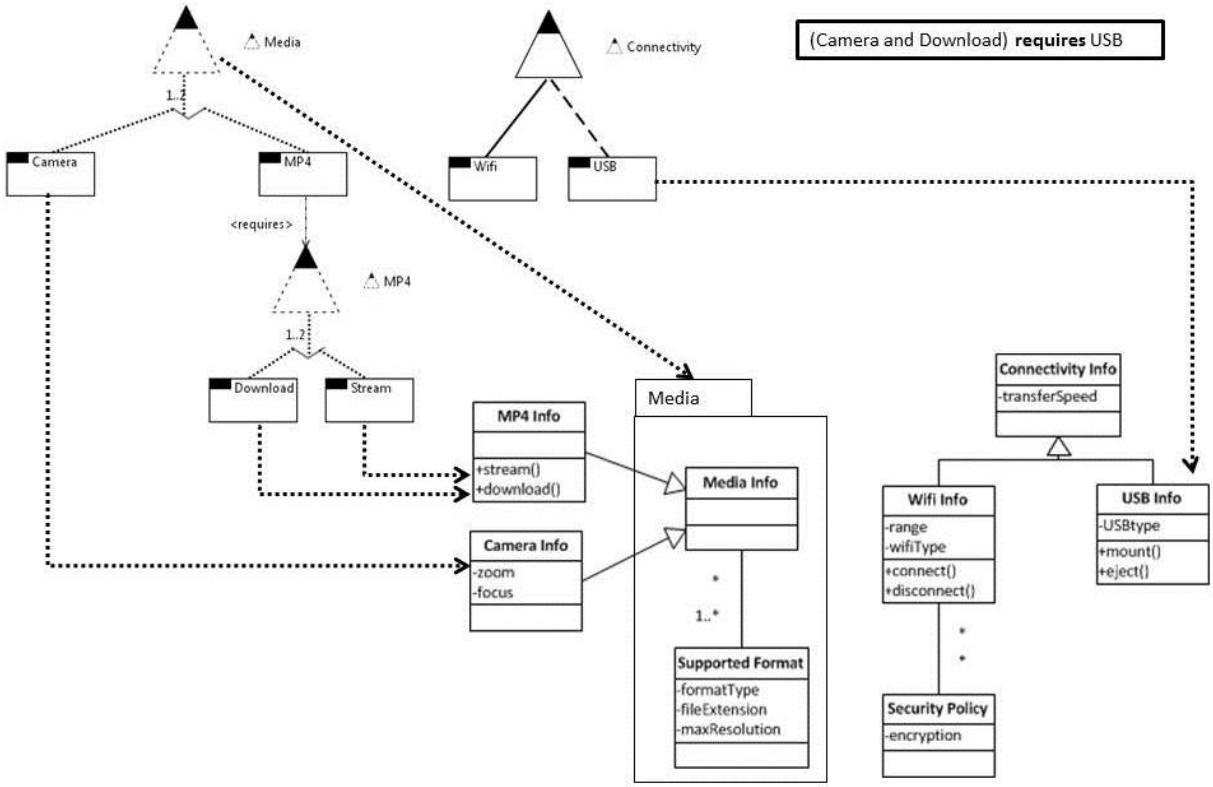

Figure 8. Part of the OVM model of mobile phones: relations to the base model