

Modelling Requirements Variability across Product Lines¹

Stan Böhne, Kim Lauenroth, Klaus Pohl

Software Systems Engineering

Institute for Computer Science and Business Information Systems (ICB)

University of Duisburg-Essen

45117 Essen; Germany

(buehne|lauenroth|pohl)@sse.uni-essen.de

Abstract

The explicit definition of variability in software product lines is a key difference between the development of single software systems and software product line engineering. More and more companies maintain several software product lines which focus on different types of products, market segments, and/or domains. Those product lines typically share commonalities and variability. The companies thus face the problem of managing communality and variability across different product lines.

In this paper we identify essential requirements for the documentation of requirements variability across product lines. We propose a meta model for structuring the variability information, sketch a prototypical realisation for managing variability across product lines in DOORS, and illustrate the use of the meta model in a small example. We further report on experiences made with the proposed variability modelling approach.

1. Introduction

A key difference between the development of single software systems and software product line engineering is the explicit definition of variability (cf. [6]). Companies often face the problem of managing variability and commonalities between different software product lines (cf. e.g., [1,8]). Van Ommering ([9]) suggests that reuse between product lines is necessary, and Thompson and Heimdahl ([12]) observed that *a part of the commonalities and variability can be shared among product lines*. This also reflects our observations in industrial projects.

For example, an electronic company may have product lines for DVD players, hard disk recorders,

and for mp3 players. For each product line common and variable requirements and features are defined, but the different product lines also share common and variable requirements and features. Consider for example, the requirements to play and record mp3 files. Within the DVD player product line, some players offer the functionality to play mp3 files. Within the mp3 player product line, playing mp3 files is mandatory, i.e. all mp3 players offer this functionality. Furthermore, some mp3 players also offer the functionality of recording mp3 files. Within the hard disk recorder product line, some recorders are able to play and record mp3 files (cf. Figure 1). In addition, the three product lines share additional requirements and offer different functionality such as different video algorithms for DVD players and hard disk recorders, or advanced audio features like Dolby Surround, or an audio equaliser.

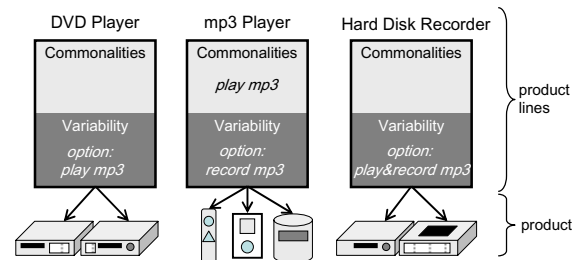


Figure 1: Different product lines within a company

Obviously a company benefits from the ability to manage those commonalities (invariants) and variability across the different product lines. Thus, variability and commonalities have to be managed across product lines.

In this paper we propose a meta model for structuring variability information across product lines,

¹ This work was partially funded by the DFG project 607/01 PRIME, and by the ITEA Project ip02009 FAMILIES Eureka Σ! 2023 Program.

report on a prototypical implementation in DOORS as well as on experiences made with our approach. We focus on the definition and management of variability and commonalities in the requirements artefacts.

In Section 2 we define three essential requirements for documenting variability across product lines. We characterise the state-of-the-art based on those requirements (Section 3). In Section 4 we introduce our variability meta model used to structure variability information. For documenting and managing variability information across product lines, we extended the meta model in Section 5. In Section 6 we sketch a prototypical realisation of the meta model in DOORS. In Section 7 we briefly report on our experience with this approach. Finally, we summarise our contribution and provides an outlook on our future work (Section 8).

2. Requirements for Documenting Variability across Product Lines

For structuring and managing variability information across product lines, we identified three essential requirements:

R₁: Document for which product line(s) a requirement is valid and whether it is common, optional, or alternative. For example, the requirement “record mp3 files” is part of the mp3 player product line as well as the hard disk recorder product line. The requirement “play mp3 files” is obviously a commonality for the mp3 player and the hard disk recorder product lines. It is, in addition, optional for the DVD player product line.

R₂: Ensure a product line-specific recording of variability constraints. Variability constraints can differ from product line to product line. For example, in the hard disk recorder product line, the requirement “play DivX files” requires the requirement “record DivX files” which enables the hard disk recorder to record DivX files which can then be played. The DVD recorder product line also offers the optional requirements “play DivX files” and “record DivX files”. However, in this case the “play DivX files” requirement does not require the “record DivX files” requirement since the DVD recorder can play DivX files from a disk.

R₃: Facilitate the creation of views on the documented variability and constraints. An approach for documenting and managing variability information across product lines has to support selective retrieval of variability and commonality in-

formation, including variability constraints. For example, it should be possible to create a view that includes the requirements that are common for all product lines. Or, it should be possible to retrieve all requirements which are defined as variable in all product lines, or one should be able to retrieve the commonalities and differences for a given set of product lines (e.g. for the product lines DVD player and hard disk recorder).

3. State of the Art

Several research contributions deal with the documentation of variability for one software product line. For example, Sinnema et al. ([11]) suggest a framework for modelling variability in software product lines. Bachmann et al. ([2]) suggest the documentation of variability for software product lines in a central conceptual model. Kang et al. ([7]) suggest a feature-model based documentation of the variability in a software product line. Van Ommering ([9]) focuses on building product populations with software components. His contribution considers more than one product line but does not deal with the documentation of variability in requirements.

Thompson and Heimdahl ([12]) propose a set-based approach to structure commonalities and variability in requirements across product lines. They argue that variability and constraints for variability can be defined within sets across product lines. They illustrate their approach using textual examples. However, the variability information within the sets appears to be unstructured. For example they do not differentiate between variation points and variants nor do they deal with variability constraints. Therefore R_1 and R_2 are not fulfilled. Without structuring the variability information, selective retrieval of the information is hard. There approach does thus not fulfil requirement R_3 . In contrast to Thompson and Heimdahl, we define a meta model to structure variability information and define the variability across product lines as “views” on this structure.

4. The Core Variability Meta Model

For documenting the variability of a product line, we follow the suggestion proposed in [2]:

- a) to explicitly define the variability of the product line in a conceptual model;
- b) to differentiate between variation points, variants, variability and constraint dependencies.
- c) to document the variability information separately (see [2] for an argumentation), i.e. not to define

the variability information within the different requirements artefacts.

Figure 2b² depicts our meta model for structuring the variability information. The meta model is represented using UML 2.0 and based on the meta model proposed in [2]. In the following we use italic to highlight model elements in the text and quotation marks to indicate examples.

A **variation point** represents a variation subject and defines what can vary in a product line. A **variant** represents the variation object and defines a concrete type of variation (cf. [10]). A *variant* is related to a variation point using a *variability dependency*. A variant can be related to one or more variation points (multiplicity 1..n) and a variation point is at least associated with an *variability dependency* to one variant, but can of course be associated with more than one variant (multiplicity 1..n).

The *variability dependency* defines whether a variant is *mandatory* or *optional* for the *variation point*:

- **Mandatory:** A variant assigned to a variation point using a *mandatory dependency* has always to be selected for this variation point.
- **Optional:** A variant assigned to an variation point using an *optional dependency* can be selected for this variation point, but must not be selected.

For the optional variability dependencies, often a selection range has to be specified, e.g. if one wants to ensure that from a set of five optional variants at least two and at most three variants are being selected for product of the product line. We therefore define the *alternative choice* which groups a number of *optional variability dependencies*. An *optional variability dependency* can be part of an *alternative choice* (multiplicity 0..1) and an *alternative choice* has at least two optional variability dependencies (multiplicity 2..n). The attributes *min* and *max* of the *alternative choice* define upper and lower bounds for a possible selection, e.g., if *min*=2 and *max*=3 from the *alternative choice* set two or three variants have to be selected. Obviously, the number of optional variants grouped by the *alternative choice* has to be at least equal to the *max* definition.

A requirements artefact (e.g. a use case, scenario, textual requirements specification, data model) is represented by the element **requirements artefact**. A *requirements artefact* is assigned to the *variant* of the variability model using a *specified by* dependency. By assigning a *requirements artefact* to a *variant*, the requirements artefact becomes variable. If a *require-*

ments artefact is not related to any variant, the artefact represents a commonality for the product line. The multiplicities express that a requirements artefact can be related to several *variants* (multiplicity 0..n), and a *variant* must be related to at least one *requirements artefact* (multiplicity 1..n). The *specified by* assignment between a *variant* and the different *product line requirements* supports the traceability between the variability model and the requirements artefacts. For example, one can trace all affected *requirements artefacts* for a given variant, or all variants that are related to a given requirements artefact, or all scenarios associated to a set of variants, etc.

The **variant constraint dependency** is used to define variability constraints which exist between two variants. A *variant* can constrain an arbitrary number of *variants* and a *variant* can be constrained by an arbitrary number of *variants* (multiplicity 0..n at both ends). We distinguish between:

- The **variant requires variant dependency**, which defines that a variant requires a variant. For example, if, for technical reasons, the variant “record mp3 files” requires the variant “play mp3 files” a *variant requires variant dependency* is introduced between the two variants.
- The **variant excludes variant dependency**, which defines that a variant excludes another variant. For example, if, for legal reasons, the variant “play DivX video files” excludes the variant “play mpeg4 video files” a *variant excludes variant dependency* is introduced between the two variants.

The **variant variation point constraint dependency** defines the relation between the *variant* and the *variation point* and is a directed edge from the *variation point* to the corresponding *variant*. In contrast to feature modelling approaches, we experienced that defining variability information for a product line does not lead to a hierarchical structure, but to a graph (this observation is was made by other researchers, e.g. [11]). We therefore introduce a constraint dependencies between variants and variation points. We distinguish between:

- The **variant requires variation point dependency**, which defines that the selection of a *variant* requires the selection of additional *variants* for the related *variation point*. For example, assume that a variation point “wireless system” offers the variants “IEEE802.1b, IEEE802.1g and Bluetooth™”. and that a *variant requires variation point* dependency between the variant “wireless media connection” and the variation point “wireless system” is defined. This dependency requires

² For reasons of space limitation, we use the following abbreviations: VP = variation point; V = variant.

- The *variant excludes variation point dependency*, which defines that the selection of a *variant* disables the selection of all *variants* of an *variation point*; e.g., the selection of the variant “flash memory” (for file storage) excludes the consideration of the variation point “record video” and thus the selection of any variant defined for this variation point (because the available flash memory is not capable of handling the necessary amount of data).

- The **variation point requires variation point dependency**, which defines that all *variants* of a *variant point* require an arbitrary *variant* of the related *variation point*. For example, all variants of the variation point “record audio files” require a variant of the variation point “audio storage”.

- The ***variation point excludes variation point dependency***, which defines that all *variants* of a *variation point* exclude all *variants* of another *variation point*. For example, all variants of the variation point “LCD display” exclude all variants of the variation point “OLED (organic light emitting diode) display”.

In our industrial cooperation we observed that different product lines often share the same requirements as commonality or as variability. As a consequence, variability information between different product lines is, in part, redundantly defined. This causes the following problems:

- The redundant definition leads to additional effort for each product line.
- The redundant definition bears a high risk of inconsistencies between the definitions.
- Equal or similar variability definitions across the product lines are hard to identify.

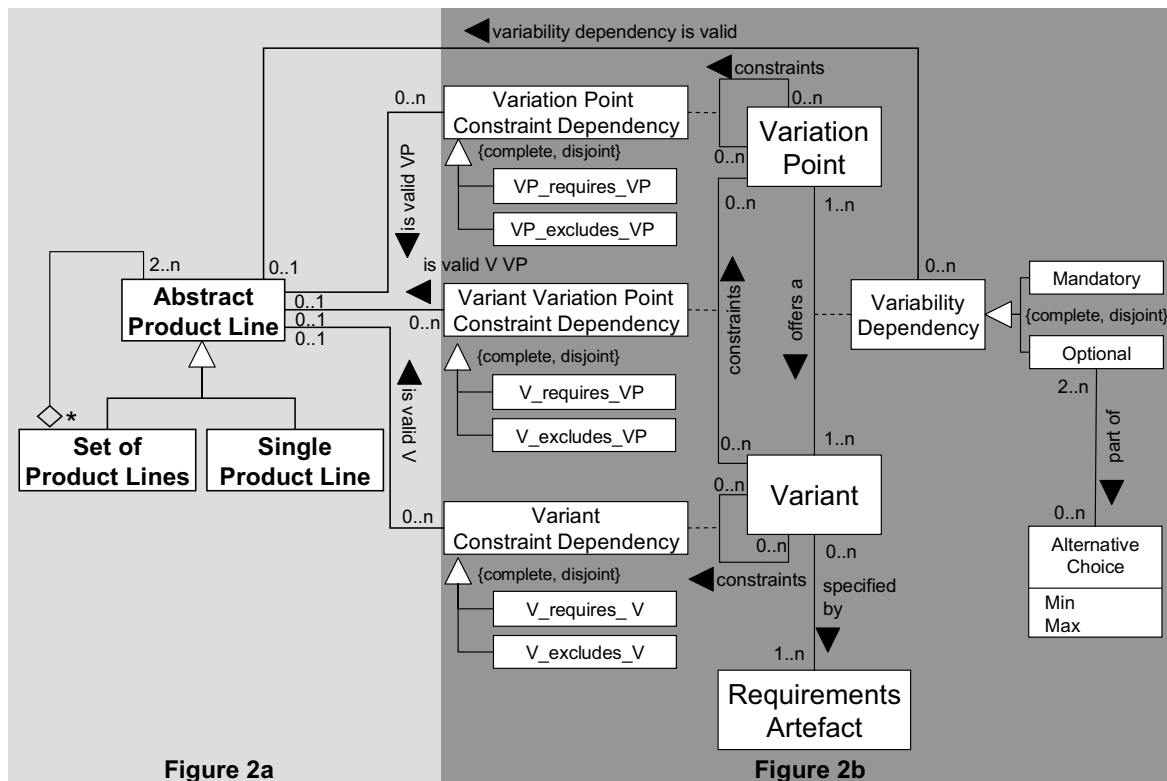


Figure 2: Extended variability meta model²

By assigning variability to different product lines (or set of product lines) it is possible to define for which product line (or set of product lines) a variability information is valid. For example, the requirement “play mp3” is mandatory for the mp3 player product line, but optional for DVD player product line.

For being able to define for which product line or set of product lines a particular variability information is valid, we extend the variability meta model introduced in Section 4 (cf. Figure 2a).

The class *set of product lines* is defined to group single product lines into sets. This grouping enables us to represent that a variability definition is valid for a set of product lines; e.g., that a variability definition is valid for the “DVD player” and “CD player” product lines.

The class *single product line* and the class *set of product lines* are generalised by the abstract class *abstract product line*. The aggregation association between the class *set of product lines* and the class *abstract product lines* enforces that the class *set of product lines* either consists of:

- at least two instances of *single product line*, or
- an instance of *single product line* and an instance of *set of product lines*, or
- at least two instances of *set of product lines*.

The class *abstract product line* is related to the class *variability dependency* through a *variability dependency is valid* relation. This relation defines which variability dependency is valid for which single product line and/or for which set of product lines. Note, that if a *variability dependency* is valid for a



single product line or a set of product line (defined by a *variability dependency is valid* relation), the associated *variation point* and *variant* become valid for the product line or the set of product lines.

Similar to the variability dependency, we introduce “is-valid-dependencies” between the class *abstract product line* and the three types of constraint dependencies (*variant constraint dependency*, *variant variation point constraint dependency*, and *variation point constraint dependency*). More precisely, we introduce

- the *is valid VP* association which defines that a *variation point constraint dependencies* is valid for a *single product line* or *set of product lines*;
- the *is valid V VP* association which defines that a *variant variation point constraint dependencies* is valid for a *single product line* or *set of product lines*;
- the *is valid V* association which defines that a *variant constraint dependencies* is valid for a *single product line* or *set of product lines*.

The multiplicities 0..1 at class *abstract product line* represent that a variability dependency or a constraint dependency may exist without an relation to a *single product line* or a *set of product lines*. The multiplicity 0..n at the variability dependency and the various constraint dependencies represents that a *single product line* or *set of product lines* may have an arbitrary number of variability and constraint dependencies. The same applies for the different *is valid* relations from the class *abstract product line* to the various constraint dependency classes. To document that a variability definition holds for all product lines, one has to define an instance of the class *set of product lines* that contains all product lines and then relate this set with the corresponding “is-valid” relation to the variability definitions.

Given the above definitions, the variability definitions for a *single product line* is defined by

- a) the variability definitions which are defined as valid for the *single product line* (using a *variability dependency is valid*, a *is valid VP*, a *is valid V VP* or a *is valid V* association), and (!)
- b) the variability definitions defined to be valid for a *set of product lines* the single product line is member of.

To avoid conflicting variability definitions for a single product line, the “is valid” associations for a *single product line* must not conflict with the “is valid” associations the single product line is a direct member or indirect member (i.e. the definitions for the set of product lines which includes a set of product lines where the single product line is a member).

For example, a variant that is defined as mandatory for the set of product lines “DVD and CD player” must not be defined as optional for the “DVD player” product line, because the *single product line* “DVD player” is member of the *set of product lines* “DVD and CD player”.

5.2. Example

Assume that we want to express the following variability across the three product lines “DVD player”, “CD player”, and “mp3 player”:

- All product lines offer the mandatory variant “digital rights management”.
- The “CD and DVD player” (set of) product line(s) offers the mandatory variant “play digital audio” from compact disk.
- The “mp3 player” (single) product line offers the mandatory variant “play mp3 files” and the optional variant “play aac files”.
- The “DVD player” (single) product line offers the optional variant “play mp3 files”.

In order to model the variability, we define a variation point “audio” and relate all the variants to this variation point. We define a set of product lines “all”, a set of product lines “CD/DVD player”, and the three single product lines “CD player”, “DVD player”, and “mp3 player” to represent variability across the three product lines.

We relate the set of product lines “all” to the mandatory variability dependency between the variation point “audio” and the variant “digital rights management” to document that this variant is part of all product lines.

We relate the set of product lines “DVD and CD player” to the mandatory variability dependency between the variation point “audio” and the variant “play digital audio” to document that this variant is part of both product lines.

We relate the single product line “mp3 player” to the mandatory variability dependency between the variation point “audio” and the variant “play mp3 files”, and to the optional variability dependency between the variation point “audio” and the variant “play aac files” to document that “play mp3 files” is mandatory for the “mp3 player” product line and that “play aac files” is optional for the “mp3 player” product line.

We relate the single product line “DVD player” to the optional variability dependency between the variation point “audio” and the variant “play mp3 files” to document that “play mp3 files” is optional for the “DVD player” product line.

The complete example together with the instantiation relation to the meta model is illustrated in Figure 3. The upper part shows an excerpt of the variability meta model and the lower part shows an instantiated product line model.

6. Prototypical Realisation in DOORS

To support the requirements engineer in defining variability across product lines, and to reuse the defined variability, the meta model has to be implemented to provide a sufficient support for the definition as well as for the selective retrievals. We have implemented the meta model in DOORS to retain the advantages of a requirements management tool, e.g. history, traceability, user management, etc. (cf. Figure 4).

For the implementation in DOORS we have defined three types of formal modules:

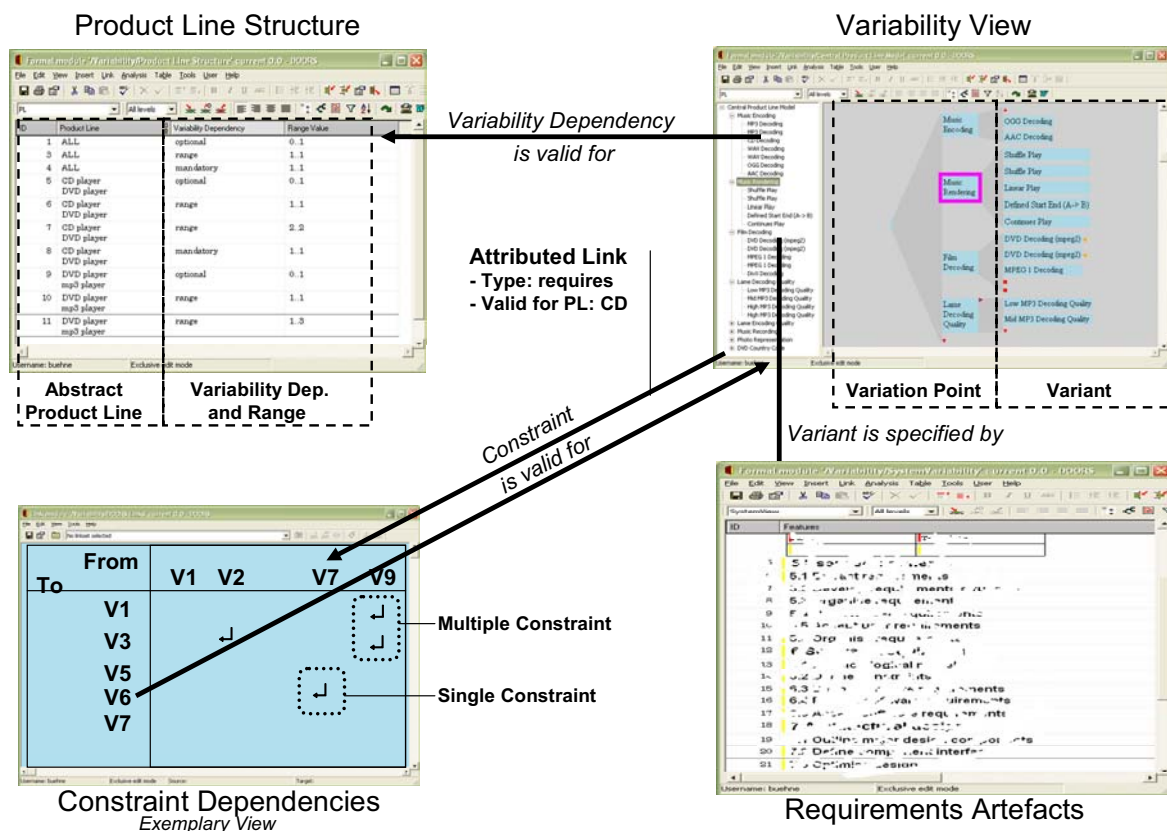
- The formal module *product line structure* documents the necessary abstract product lines (product line set instances and product line instances),

the variability dependency (mandatory, optional, alternative choice), and the corresponding range of the variability dependency (e.g. 0..1 for optional, n..m with $n \leq m$ for alternative choice, 1..1 for mandatory).

- The formal module *variability view* documents all existing variation points and variants for the considered product line without any definition of how the variant is available at the variation point. This module only illustrates which variants belong to which variation points.
- The formal module(s) *requirements artefacts* document the different requirements that specify the abstract variants, for instance, by using scenarios, functional requirements, or quality requirements.

Furthermore, a set of link modules documents the associations between the different modules, that exist in the variability meta model:

- The link module *constraint dependencies* documents the occurring constraints between variation points, variants, and variants and variation points.



To be able to specify the validity of the constraint (cf. *is valid for* in meta model), we attributed the links by the attribute *link type* (requires, excludes), and the attribute *valid for* (which defines the product line (set) that the link is valid for).

- The other link modules store the traces between objects of two modules. The module variability dependency documents the traces between the formal modules *product line structure* and *variability view*. The *specified by* links are documented in the link module *specified by* and documents the traces from variants to the corresponding requirements artefacts, such as scenarios or textual requirements.

The realisation of the variability meta model introduced in Section 5 in DOORS facilitates:

- the **comparison of and the reasoning about variability information** defined for different product lines. For example, overlaps and the differences of the variability of two product lines, such as DVD and MP3 player, can be displayed;
- the retrieval of **variability information for a given variant** defined for different product lines. For example, we can retrieve all product lines which offer the variant *mp3 play*, and whether the variant is mandatory, optional or an alternative choice (represented as range attribute in DOORS) – which is essential in the case of change integration;
- the retrieval of **variability information for a given product line** as well as for a set of product lines. For example, with the attributes *product-line = 'all'* AND variability dependency = “man” we are able to retrieve all variants that are common for all product-lines;
- the retrieval of **variability information for a given variation point**. For a given variation point we can retrieve all variants defined for this variation point in any single product line or set of product lines – including the variability constraints and variability definitions (mandatory, optional, alternative choice) for each particular product line.

Moreover, the separation of variability information and requirements artefact information (as defined in our meta model) facilitates the reasoning about variability information without considering the “realisation” of the variability in the requirements artefacts, i.e. independent of the realisation in the various requirements artefacts (textual requirements, goals, use cases, scenarios, data models, behaviour models).

Figure 5 depicts an example of a retrieval of variability information stored in our DOORS prototype. The DOORS screenshot illustrates the selective retrieval for a “mp3 player” product line. The module represents five columns:

- the first column indicates if the information in this row is about a variation point (VP) or variant (V);
- the second column contains the name of the variant or variation point;
- the third column names the variation point that the variant depicted in this row belongs to;
- the fourth column shows the variability dependency, i.e. if the variant is mandatory, optional or an alternative choice for this variation point;
- the fifth column shows the validity of the dependency, i.e. this attribute defines, for which product lines (set or single) the variant shall be available with the defined variability dependency.

Figure 5 represents the result of a selective retrieval for the “mp3 player” product line, with the columns variant, variation point, variability dependency, and the abstract product line for which the dependency is valid for. Thereby variants for the “mp3 player” are composed of:

- variants that are defined for the “mp3 player”,
- variants that are defined for all product line sets where the “mp3 player” is a part of the set,
- and finally of variants that are valid for all product lines.

Type	Variant	Variation Point	Variability Dep.	abstract PL
VP	Music Rendering		unspec	ALL
V	Shuffle Play	Music Rendering	opt	DVD MP3
V	Linear Play	Music Rendering	man	ALL
V	Defined Start End (A-> B)	Music Rendering	opt	ALL
V	Continues Play	Music Rendering	man	ALL
VP	Film Decoding		unspec	ALL
VP	Lame Decoding Quality		unspec	ALL
V	Low MP3 Decoding Quality	Lame Decoding Quality	man	DVD MP3 HDR
V	Mid MP3 Decoding Quality	Lame Decoding Quality	man	DVD MP3 HDR
V	High MP3 Decoding Quality	Lame Decoding Quality	opt	MP3
VP	Lame Encoding Quality		unspec	ALL
V	Mid MP3 Encoding Quality	Lame Encoding Quality	man	MP3 HDR
V	High MP3 Encoding Quality	Lame Encoding Quality	opt	MP3
VP	Music Recording		unspec	ALL
V	MP3 Encoding	Music Recording	opt	MP3 HDR
VP	Photo Representation		unspec	ALL
V	JPEG Picture Representation	Photo Representation	opt	DVD MP3
VP	DVD Country Code		unspec	ALL

Figure 5: Retrieval of variability information in DOORS.

This information can be generated by the defined traces between the modules *variability view* and

product line structure. Overall, the DOORS realisation supports the requirements engineer to identify:

- where (in which product lines) a variant is used (e.g. the variant *shuffle play* is available in all hard disk recorders and can be available in all mp3 player and DVD players),
- which variants are used in different product lines (e.g. the variant *mp3 decoding* is used in hard disk recorder, mp3player and DVD player) and
- which variants are available in all product lines (e.g. the variant *continuous play* is available in all product lines).

Moreover, due to the documented trace information between variants and requirements artefacts in the module *specified by*, the requirements engineer is able to retrieve all corresponding requirements artefacts related to the variants of the considered product line. Thereby the engineer is supported to check the definition of variability information in different requirements artefacts for consistency. For example, the engineer can retrieve for each variant defined in the variability model the corresponding definition of this variability in all parts of the textual definition. Thus he can easier check if the definitions in those artefacts correspond to the intended variability defined in the variability model.

7. Experiences

We applied the variability modelling approach to several industrial examples, mainly from the automotive industry³. Although a comprehensive evaluation of this approach is still missing, our gathered experiences indicate that the explicit definition of product line structures, and with it, the explicit documentation of variability information across product lines, leads to:

- a *better understanding* for engineers of the variability available within the different product lines – sometimes they recognised that the differences are much smaller than expected;
- a *quality improvement* of the documented requirements, e.g. identified errors can be fixed for all affected product lines and changes can be consistently integrated since all affected parts of the requirements documentation can be retrieved;
- a *reduction of the re-invention* of already existing variability through the overall availability of variability information, i.e. the identification of vari-

ability unknown to engineers of other product lines;

- a *harmonisation of requirements variants*, due to the identification of slightly different variants in several product lines that can be unified, so that only one variant for these product lines has to be provided. For example, the variability “play mp3 files” and “play music files that are encoded with mp3 compression” can be unified to one variability definition and thereby complexity is reduced;
- a *clearly defined delimitation* of product lines, and thus to a better understanding where the provided product lines differ from each other, and where the product lines look alike;
- a *better exchange of knowledge* about variability information, and thus a better use of variants that are available across product lines, e.g. due to the existence of “digital rights management” for DVD player the variant can be easily integrated in all CD player product lines;
- a *higher reuse of variants* for individual products, i.e. due to the fact that all variants of all product lines are available, specific customer needs can be fulfilled by the reuse of such variants.

Moreover, the definition of variability across product lines in a central model supports the definition of a new product line. For example, if a product line for a combined DVD/hard disk recorder should be launched, the engineers are supported in accessing commonalities and variability of the DVD player product line and the video recorder product line. It further supports product management, e.g., the comparison of the own product lines with the products of competitors is supported by providing an overview of the commonalities and the variability of the product lines and by the ability to compare those definitions.

8. Summary and Outlook

To facilitate the definition of variability across product lines, we have proposed a meta model which defines a clear structure for recording variability information and facilitates the definition of “views”, and thereby supports the definition of variability information across product lines. Our variability meta model satisfies all the three requirements stated in Section 2. Our meta model:

- provides a clearly defined structure for recording variability information based on the explicit definition of variation points, variants, variability dependencies, and constraint dependencies;

³ The examples in this paper are taken from the home entertainment domain due to legal problems.

- facilitates the definition of variability dependencies which are valid for a single product line and/or for a set of product lines, and thus enables us to define for which single product line and/or set of product lines a variation point and/or a variant are valid;
- facilitates the definition of constraint dependencies which are valid for a single product line and/or for a set of product lines;
- facilitates the implementation of tool support;

Based on the meta model, we have realised a prototypical implementation in DOORS which facilitates the documentation of variability information across product lines and, based on the clear structure, allows the selective retrieval of variability information. For example, one can retrieve which variants are shared among different product lines, or which requirements artefacts are optional for which product lines, etc.

The realisation in DOORS indicates, that the proposed approach can be implemented as an extension of commercial requirements management tools with relatively small effort. However, our experience also indicates, that more sophisticated tool support is required for: (1) the definition of variability; (2) the maintenance of variability information; (3) consistency checking for all variability definitions across the product line sets; (4) the definition of advanced selective retrievals; and last but not least (5) an appropriated representation of variability for different stakeholders, such as customers, product managers, architects, developers, etc.

Currently, we are developing a more sophisticated tool support, which assists in the definition of product line sets, and the consistent definition and documentation of variability across product lines by providing consistency checks for defined variability dependencies and constraint dependencies across product lines. We further intend to validate our approach and the variability management tool in large scale industrial case studies.

9. Acknowledgements

We thank Ali Botorabi, Christian Dinnus, Günter Halmans, Andreas Metzger, Mark-Oliver Reiser, Ernst Sikora, Matthias Weber, and Thorsten Weyer for fruitful and stimulating discussions.

10. Literature

- [1] M. Aoyama, K. Watanabe, Y. Nishio, and Y. Mori-waki, "Embracing Requirements Variety for e-Governments Based on Multiple Product-Lines Frameworks", *Proceedings of the 11th IEEE International Conference on Requirements Engineering*, Monterey Bay, California, USA, September 2003, p. 285.
- [2] F. Bachmann, M. Goedicke, J. Leite, R. Nord, K. Pohl, B. Ramesh, and A. Vilbig, "A Meta-Model for Representing Variability in Product Family Development", *Proceedings of the 5th International Workshop on Product Family Engineering*, Siena, Italy, November 2003, pp. 66-80.
- [3] J. Bosch, *Design & Use of Software Architectures, Adopting and Evolving a Product-Line Approach*, Addison-Wesley, Boston, May 2000.
- [4] S. Bühne, K. Lauenroth, and K. Pohl, "Why is it not Sufficient to Model Requirements Variability with Feature Models?", *Proceedings of the International Workshop on Automotive Requirements Engineering, co-located at RE'04*, Nanzan University, Nagoya, Japan, September 2004, pp. 5-12.
- [5] S. Bühne, G. Halmans, and K. Pohl, "Modelling Dependencies between Variation Points in Use Case Diagrams", *Proceedings of the 9th International Workshop on Requirements Engineering – Foundation for Software Quality*, Klagenfurt/Velden, Austria, June 2003, pp. 59-70.
- [6] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison-Wesley Longman Reading, Boston, August 2001.
- [7] K. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering", *IEEE Software*, Vol. 19, No. 4, July/August 2002, pp. 58-65.
- [8] R. v. Ommerring and J. Bosch, "Widening the Scope of Software Product Lines - From Variation to Composition", *Proceedings of the 2nd Software Product Line Engineering Conference*, San Diego, CA, USA; Springer, August 2003, pp. 328-347.
- [9] R. v. Ommerring, "Building Product Populations with Software Components", *Proceedings of 24th International Conference on Software Engineering*, Orlando, Florida, September 2002, pp. 255-265.
- [10] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer, Heidelberg, August 2005.
- [11] M. Sinnema, S. Deelstra, J. Nijhuisa, and J. Bosch, "COVAMOF: A Framework for Modelling Variability", *Proceedings of the 3rd Software Product Line Conference*, Springer, Boston, August/September 2004, pp. 197-213.
- [12] J. Thompson and M. P. Heimdahl "Structuring Product Family Requirements for n-Dimensional and Hierarchical Product Lines", *Requirements Engineering Journal*, Vol. 8, No. 1, Springer, London, April 2003, pp. 42-54.