Tool Interview – VaVe

1 INITIAL CONCEPTUAL MODEL

The initial conceptual model describes essential concepts (or a superset of it) for modeling variability of a software system in space and time and shall subsume functionality related it. Additionally, the model unifies those concepts to represent revisions of variable system parts. The conceptual model follows an open-world assumption (descriptive) instead of a closed-world assumption (prescriptive) as metamodels commonly do. In Table 1 we provide a definition of the involved concepts.

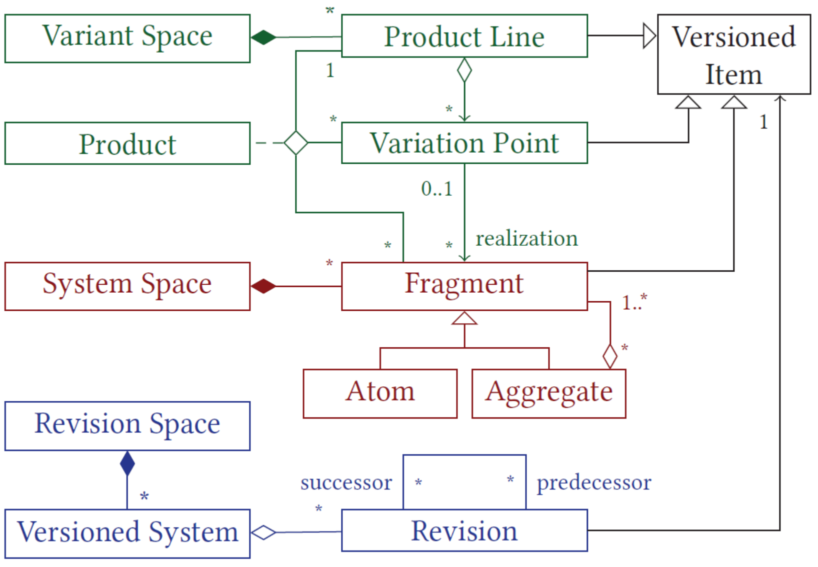

Figure 1: The Initial Conceptual Model with essential and combining Concepts for Variability in Space and Time.

Table 1: Definition of concepts in the Conceptual Model.

Concept	**Direct relation to other Concepts**	**Definition**
Fragment	*Variation Point, Product*	*Fragments* are the essential concept to describe a system on realization level. A *Fragment* can either be an atom or an aggregate, e.g. a single file, character or the node of an AST. A hierarchical structure of containments is not enforced but instead *Fragments* can be composed to various combinations.
Product Line	*Variation Point, Versioned Item*	A *Product Line* represents the configurable space regarding spatial variability and is composed of a system‘s *Variation Points*.
Variation Point	*Product Line, Fragment,*	A *Variation Point* expresses the variability of a system by representing an option set for variation of the *Product Line*.

	Product, Versioned Item	A *Variation Point* can either be explicit (e.g., if-defs or a plug-in system with a compositional variability realization mechanism) or implicit (a reference between a feature module and fragment represents the implicit variation points, therefore the fragment is not aware of its variation e.g., FOP, AOP, delta modeling).
Product	*Product Line, Variation point, Fragment*	A *Product* is fully specified if all existing *Variation Points* in the *Product Line* are bound to *Fragments* or *Variation Points* are not bound explicitly, e.g., if a feature is optional and not selected for product (hence, all to a configuration relevant *Variation Points* are bound to fragments). A partial *Product* does not require the binding of every *Variation Point*.
Revision	*Versioned Item*	A *Revision* of the *Fragment* evolves along the time dimension and is intended to supersede its predecessor by an increment, e.g., due to a bug fix or refactoring.
Versioned System	*Revision*	A *Versioned System* represents the configurable space regarding temporal variability. It is composed of a system‘s revisions.
Versioned Item	*Revision*	The *Versioned Item* represents versioning of the introduced concepts for *Fragment*, *Variation Point* and *Product Line* by putting them under revision control.

Table 2: Particular Relations of the Conceptual Model.

Relation	Direct relation to Concepts	Definition
Realization	*Variation Point, Fragment*	Each *Variation Point* has a set of possible options for variation whereby each option is realized by *Fragments*.
Configuration	*Product Line, Variation Point, Fragment*	A *Configuration* defines one particular *Product* of a *Product Line* by resolving the variability of a *Product Line*, i.e., binding all relevant *Variation Points* of a *Product Line* to *Fragments*.
Branching / Merging	*Revision*	To represent *branching* (which is considered a temporary divergence for concurrent development) along with *merging*, multiple (direct) successors and predecessors relate to a revision. This relation gives rise to a revision graph, which is a directed acyclic graph where each node represents a unique revision.

2 INTERVIEWS

Please inspect

1. If
2. and if yes, how

concepts of the conceptual model are represented by constructs used in your tool. Therefore, the representation of each concept in the tool and their (direct) relation to other constructs is considered separately.

Table 3: Concept Mapping between Conceptual Model and Tool.

Concept	**Representation of Concept in Tool**	**Relation to other Constructs**
Fragment	A fragment can be any type of realization artifact and may span code, models, documentation etc. Due to technical reasons, a prerequisite is that these fragments have a representation based on EMF Ecore, i.e., a meta model that is suitable to represent concrete fragments as models of that meta model. There are no further requirements on, e.g., the structure of the meta model or regarding marking of variation points. Fragments can be arranged to various combinations representing a graph structure.	A fragment may reference another fragment, e.g., an import of a class.
Product Line	A product line is represented by a tool-specific variability model that comprises variation points that contain a set of variants (similar to an Orthogonal-Variability Model). The VaVe variability model is arranged in a tree-structure with additional cross-tree constraints along with delta modules that invasively modify fragments via transformation and a mapping between variants and delta modules.	Variation Point
Variation Point	Explicitly represented in the VaVe-specific variability model. Each Variation Point contains a set of variants.	Fragments

Product	Represented on two levels: A product on the conceptual level is represented by a configuration, i.e., a selection of features in specific versions. A product on the realization level is referenced as a product, i.e., the fragments in the variation representing the selection of the configuration (in other words, the software system resulting from the configuration). A product is represented by fragments of any type of realization, e.g., state machine + java code. A product can must be created from scratch.	
Revision	Represented as a version of a variant, which denotes an implementation of a variant at a particular point in time. Versions are perceived as being incremental, i.e., version 1.1 is a change to the functionality of version 1.0.	Variation Point, Version
Versioned System	The sum of all variants and delta modules represent the versioned system.	
Versioned Item	The variant.	Delta Modules
Realization	A mapping is represented by a realization link between versions of variants and its realizing delta modules.	Delta Modules
Configuration	A configuration on conceptual level is a selection of features with exactly one version elected for each feature.	VaVe variability model
Branching / Merging	Support for Branching & Merging	Version
Remarks	Core concepts of VaVe: • variability model (variation points, variants, cross-tree constraints) • Delta Modules • Configuration • Application order constraints	

3 USE CASES

Please provide an overview of use cases that your tool addresses.

- The VaVe approach copes with variability in space and time in a delta-driven way.
- It aims at embedding variability in space and time in multi-view development
- Additionally, it preserves consistency between views (products / solution space) and between views and the feature model (problem space)

4 PREVIEW: SEMANTICS

The semantics of several concepts is only defined through the mechanisms that operate on them. For example, the configuration of a product from a product line, variation points and fragments is expressed in the conceptual model, but constraints that define which variation points and fragments may be selected have to be ensured by a configuration mechanism. The same applies to the generic concept of the *Versioned Item*. A mechanism that defines how the relation between revisions of product lines, variation points and fragments can be combined has to be defined. Designing such mechanisms, based on the conceptual model, is the next step towards a unifying concept for variability in space and time.
We consider semantics represented by the following mechanisms of a system that deal with variability in space and / or time:

1) *Analyses mechanisms* support the validity of:
    a. the variability model
    b. the configuration
    c. the fragment

2) The *mapping mechanism* that is used to resolve a configuration from a variability model to a set of realization artifacts

3) A *variability realization mechanism* assembles realization artifacts for a configuration in a particular manner (*annotative* variability, e.g. #ifdefs; *compositional* variability, e.g., feature-oriented programming; *transformational* variability, e.g., delta modeling).

In the following, please describe the semantics of your tool regarding the described mechanisms.

Analyses mechanisms
For analyses of the variability model and configuration, analyses mechanisms of FeatureIDE are used.

Mapping model
The VaVe variability model encompasses a mapping between individual feature versions and delta modules.

Variability realization mechanism:

As variability realization mechanism, delta modeling as transformational mechanism is applied.

A. TABLE OF TABLES

Table 1: Definition of concepts in the Conceptual Model. 2
Table 2: Particular Relations of the Conceptual Model. 3
Table 3: Concept Mapping between Conceptual Model and Tool. 4

B. REFERENCES

[1] S. Ananieva, T. Kehrer, H. Klare, A. Koziolek, H. Lönn, S. Ramesh, A. Burger, G. Taentzer and B. Westfechtel, "Towards a conceptual model for unifying variability in space and time," *Proceedings of the 2nd International Workshop on Variability and Evolution of Software-Intensive Systems,* 2019.

[2] G. Guizzardi, L. F. Pires and M. van Sinderen, "An Ontology-Based Approach for Evaluating the Domain Appropriateness and Comprehensibility Appropriateness of Modeling Languages," *Proceedings of the International Conference on Model Driven Engineering Languages and Systems,* 2005.