Deriving Variability Patterns in Software Product Lines by Ontological Considerations

Mohsen Asadi[1], Dragan Gasevic[2], Yair Wand[3], Marek Hatala[1],

[1] Simon Fraser University, Canada, [2]Athabasca University, Canada, [3]University of British Columbia, Canada
{masadi, mhatala}@sfu.ca, dgasevic@acm.org
Yair.wand@ubc.ca

Abstract. Variability modeling is widely used in software product line engineering to support reusability. Specifically, it is used in the derivation of concrete software products from a reusable solution within a family of products. To help manage variability, several modeling languages have been proposed for representing variability within a family of products. The study and evaluation of languages to model variability has so far focused on practical aspects of such languages. Less attention has been paid to more theoretical approaches to the analysis of variability modeling languages. In developing such approaches it would be of particular interest to explore the ability of variability modeling to represent the information about the real world (application) domain for which the product family is designed. In information systems research, evaluation of expressiveness of conceptual modeling languages has been done based on ontological theories. This paper describes a framework for general analysis of types of variability based on Bunge's ontology and derives a variability framework which is used to evaluate variability modeling languages.

Keywords: Variability modeling, ontological theory, software product lines

1 Introduction

Software Product Line Engineering (SPLE) is an approach to develop a set of software systems which satisfy requirements of a specific domain and share common features [1]. The key factor for the success of SPLE is variability which can be categorized into [3]: *essential variability* (i.e., variability from the system usage perspective) and *technical variability* (i.e., variability related to the realization/implementation of essential variability and/or variability related to the IT-infrastructure)[2]. Essential variability is variability in domain knowledge and can be represented by *conceptual models*. Conceptual models are representations of static (e.g., things and their properties) and dynamic (e.g., process and events) phenomena in a domain of interest [7]. Variability languages should be able to represent both essential variability (manifested in the conceptual models) and technical variability (manifested in the implementation models). Several variability languages have been proposed which use approaches such as feature modeling [1] and orthogonal variability modeling [3]. The evaluation and improvement of expressiveness of variability languages have attracted some attention in SPLE research [4][5]. However, theoretical analysis of variability modeling languages with respect to their ability to represent variability in real world domain models has yet not been studied in detail. Theoretical analysis of variability languages help in better understanding of expressiveness of these languages and consequently changes in these languages for proper representation of variability between conceptual models.

In this paper, we investigate the use of ontological theories for theoretical analysis of variability modeling languages. An ontological theory defines constructs required for describing the structure and processes of the world in general. Ontological theories have been used to evaluate modeling languages in terms of correspondence of ontological concepts to modeling constructs. Bunge's ontology (as adapted by Wand and Weber [7]) has been used to evaluate several conceptual modeling languages [7][8]. This ontology includes a set of high level constructs for representing real world phenomena. The evaluation of modeling languages is based on the assumption that an information system is an artifact that represents a real-world domain. Since Bunge's ontology provides concepts for representing real world phenomena, we presume that it is appropriate for analyzing the *essential* variability of software systems which represents variability of real-world domains. Therefore, in this paper, we analyze *essential* variability of products in a software product line by investigating variability between real-world domains they are intended to represent. More specifically, we employ concepts from Bunge's ontology (as adapted for information systems [7]) as the theoretical framework for identifying possible variability patterns among products based on real-world domains variability. Afterward, we develop a theoretical framework for variability and apply the framework for evaluating expressiveness of variability languages.

2 Backgrounds

2.1 Bunge's Ontology

In Bunge's ontology [6], the "world is made up of substantial *things* which possess *properties*". Examples of things are specific instances of person, car, or book and properties are name, color, or weight. A *property* can be either *intrinsic* (i.e., is possessed by one thing) or *mutual* ("meaningful only in the context of two or more things" [7]). "Properties in general are represented by *state functions*, the values of which express individual properties" ([8] p.4). A *functional schema* is formed by a set of state functions that are used to describe similar things. A *state* is "the vector of values for all property functions of a thing" [8]. An event is "a change of a state of a thing" [8]. A state *law* defines any restriction on the individual properties (or attributes) of a thing. The *process* of a thing comprises the events that the thing might undergo and is determined by a set of transition laws [7]. A set of things that possess a common property is termed a *class*. A set of things having several properties in common is called a *kind*. A *natural kind* is a kind of things adhering to the same laws. Bunge's ontology defines more concepts that will be mentioned when are needed in the paper.

2.2 Variability in Software Product Line Engineering

Variability is the central concept in SPLE [2] and have been investigated with different perspectives and assumptions. Mainly, existing points of views on the variability notion can be categorized into two classes:

- Bosch et al. [9] defines variability as "the ability to be changed, customized, configured, or extended for use in a specific context".
- Weiss et al. [10] define variability as "an assumption about how members of a family may differ from each other ".

Mentzger et al. [11] refer to software variability as the first class of variability and product line variability as the second class. Product line variability differs from software variability; product line variability is an explicit decision of product management about what should vary between the systems in a product line and what should not [3];

software variability is an inherent property of the software under development and represents different behavior of the software. Product line variability originates from differences among real-world domains which are represented by the products of a product line. For example, in a shopping domain, real-world shopping systems may vary in their types of payment methods where one may include *credit card* and *debit card* and another may have *debit card* and *cash.* This variation leads to product line variability for their corresponding information systems. But, in a shopping domain, if all shopping systems have all three types of payment methods and hence, the payment methods cannot be varied among different members of the product line; that is, there may be software variability but there is no product line variability.

Since software products are representation of real-world domains, one can analyze the sources and types of variability which exist among real-world domains by using an ontological theory. In this paper, we concentrate on the ontological analysis of essential variability to identify variability patterns that can exist among different products.

2.3 An Illustrative Example

To exemplify the concepts in the remainder of the paper, we use a standard case study commonly used in comparative analyses of information system methodologies [12]. This is the IFIP working conference case study ([12] pp. 8-9). We assume three conference examples named conference A, conference B, and conference C, and we analyze their variability. Table 1 shows these three conferences.

Table 1: Three imaginary conferences based on the IFIP working conference example

	Conference A	Conference B	Conference C
Things	Specific instances of Program Committee, Organizing Committee, Participants, Papers	Specific instances of Program Committee, Organizing Committee, People Involved, Papers, Demos	Specific instances of Program Committee, Organizing Committee, Attendees, Papers, Art Works,
Properties (attributes)	Paper (Submission ID, Title, Author(s) Name, Quality, Type, Status), Authors(Name, Papers(s), Affiliation, Role)	Paper (Paper ID, Title, Author(s) Name, Quality, Category, Status), Authors (AName, Paper(s), Affiliation, Role, Conflict)	Paper (Paper ID, Title, Author(s) Name, Quality, Category, Status), Authors(AName, Papers(s), Affiliation, Role)
Lawful State Spaces	Paper-Status (Submitted, Accepted, Rejected, Short Papers) Paper-type (Experience, Research, Evaluation. Ideas)	Paper-Status (Submitted, Accepted, Rejected, Conditionally Accepted) Paper Category (Theoretical Foundation)	Paper-Status (submitted, Accepted, Rejected, Conditionally Accepted) Paper-Category (Theory, Practice)
Lawful Events	Paper{(Submitted→Accepted) (Submitted → Rejected), (Submitted →Short Paper Accepted)	Paper{(Submitted→Accepted) (Submitted → Rejected), (Submitted → Conditionally Accepted), (Conditionally Accepted→Accepted), (Conditionally Accepted →Rejected)}	Paper {(Submitted→Accepted) (Submitted → Rejected), (Submitted → Conditionally Accepted), (Conditionally Accepted→Accepted), (Conditionally Accepted →Rejected) }

3 Ontological Analysis of Essential Variability

According to the representation premise of Bunge's ontology, we can conclude that every product (i.e. information system) in a product line family is a representation of a real domain. For example, a software system developed for managing conference A is a representation of conference A in the real-world. Thus, the software system's conceptual model represents static and dynamic phenomena of the real-world domain of the system under study (i.e., conference A). As already indicated, essential variability represents variability among knowledge of real-world domains, and can be represented by different conceptual models of the products. Hence, we investigate essential variability by exploring variability of domains which these products are intended to represent. To investigate variability between different real world domains corresponding to different products, we assume that we can map the phenomena in one domain to the phenomena in another domain.

Mapping Premise: We assume that we can establish corresponding mappings between *things*, *attributes, states,* and *events* in one domain and *things*, *attributes, states,* and *events* in another domain.

For example, for the IFIP conference context, we can establish mappings between property *Submission ID* of thing $Paper \in Conference\ A$ with property *Paper ID* of thing $Paper \in Conference\ B$. The mapping is denoted as $A(Paper.Submission\ ID) \leftrightarrow B(Paper.Paper\ ID)$.

To investigate variability among different domains, first we analyze similarity patterns among things in these domains. Next, we consider variability as opposite to similarity and define variability patterns between different real-world systems

3.1 Variability Patterns among Sets of Phenomena

Before introducing variability classes in terms of Bunge's concepts (i.e., thing, property, state, and event), we introduce a set of general variability patterns between two sets of phenomena (By phenomena we refer to any possible observation that can be made about the domain or part of it). Afterwards, considering these variability patterns and Bunge's concepts, a set of variability classes among two domains are defined.

Assume $S = \{s_1, s_2, \dots, s_m\}$ is a set of phenomena belonging to domain D_1 and $T = \{t'_1, t'_2, \dots, t'_n\}$ is a set of phenomena belonging to domain D_2. Now, we have one of following situations with respect to similarity between these two sets.

Definition 1 (Equivalent Sets of Phenomena): *S is equivalent to* T (denoted as $S \equiv T$), if and only if there is a mapping between elements in S and elements in T.

Definition 2 (Similar Sets of Phenomena): *S is similar to* T with respect to p (denoted as $S \cong_p T$) if and only if there is a subset of S (i.e., $S' \subset S$) and of T (i.e., $T' \subset T$) which are equivalent $S' \equiv T'$. p is equivalent subset i.e. $p = S' = T'$.

Definition 3 (Completely Dissimilar Set of Phenomena): *S is completely dissimilar to* T with respect to (denoted as $S \neq T$) if and only if there are no subsets of S(i.e., $S' \subset S$) and of T (i.e. $T' \subset T$) that are equivalent.

Based on Definitions 1-3, we can define the following similarity patterns between two different sets of phenomena:

- *Full similarity double side* – when a set of phenomena S and set of phenomena T *are equivalent* (i.e., $S \equiv T$).
- *Full similarity one side* – when a set of phenomena S and a set of na T are $similar\ (i.e., S \cong_p T)\ and$ when $we\ have$ either $S' \subset S$ and $S' \equiv T$ or $T' \subset T$ and $S \equiv T'$.
- *Partial similarity* – when a set of phenomena S and a set of phenomena T are *similar* (i.e.,$S \cong_p T$) and there is no subset of one functional schema that is equivalent to the other functional schema.
- *Complete Dissimilarity* – when two sets *of* phenomena are *completely dissimilar*.

One of the above similarity patterns may happen between attribute sets of phenomena in different real-world domains. All the above patterns, except *full similarity double side*, represent possible variability between two sets of phenomena. To investigate variability between more than two sets which belong to different real-world domains, we can explore the variability patterns between each set and the rest of the sets.

3.2 Variability among Things of Different Real-World Domains

According to Bunge's ontology, a real-world domain is comprised of things. To investigate variability between domains, we need to explore variability among things in the product domains in terms of the variability of their structure and processes. The structure and processes of things is defined in terms of a combination of their properties, states, laws and events. We analyze variability among things by analyzing variability among their *attributes (properties representation)*, *lawful state space* and *lawful event space*. To investigate variability among two things in terms of their attributes, we form functional schemas which are sets of attributes for defining those things. Table 2 shows different similarity classes of two things in different real-world domains.

Table 2: Similarity classes between things of different product domains

Patterns	Class name	Description
Full similarity	*Full similarity among functional schemas*	Equivalent functional schemas
	Full similarity among lawful state spaces	Equivalent lawful state spaces
	Full similarity among lawful event spaces	Equivalent event spaces
Full similarity one side	*Full similarity one side among functional schemas*	Similar functional schemas and a subset of one functional schema is equivalent of the other functional schema
	Full similarity one side among lawful state spaces	Similar lawful state spaces and a subset of one lawful state space is equivalent of the other lawful state space.
	Full similarity one side among lawful event spaces	Similar lawful event spaces and a subset of one of the lawful event spaces is equivalent of the other lawful event space.
Partial similarity	*Partial similarity among functional schemas*	Similar functional schema and there is no subset of one functional schema that is equivalent to the other one.
	Partial similarity among lawful state spaces	Similar lawful state spaces and no subset of one lawful state space that is equivalent to the other lawful state space.
	Partial similarity among lawful event spaces	Similar lawful event space and there is no subset of one lawful event space that is equivalent to the other event space.
Complete dissimilarity	*Complete dissimilarity among functional schemas*	Dissimilar functional schemas
	Complete dissimilarity among lawful state spaces	Dissimilar lawful state space
	Complete dissimilarity among lawful event spaces	Dissimilar Lawful event space

For instance for IFIP conferences, we have a full similarity double side among functional schema defining paper in conference A and functional schema defining paper in conference C. Moreover, there is a subset of the functional schema defining authors of conference B which is equivalent to the functional schema describing authors of conference A. Hence, there is a full similarity one side between the functional schemas of the authors in the two conferences.

3.3 Variability among Real-World Domains

After exploring variability among *things* of different product domains, we can define variability classes between two *product domains* belonging to a product line. Variability among product domains may occur in both static (structure) and dynamics (processes) of the domains. The structure of a domain is defined based on things in the domain. Considering Bunge's ontology, the structure of a domain can be shown using functional schemas and a lawful state space of the whole domain. Hence, to investigate structural variability between two domains, we need to consider combinations of variability classes defined for functional schemas and lawful state spaces of these two domains. A difference between the functional schemas implies a difference between the conceivable state spaces and lawful state spaces of the two product domains. Therefore, variability between functional schemas of the domains leads to the variability of the state spaces of the domains. However, an equivalence of the two functional schemas means the equivalence of their conceivable state space, but does not mean an equivalence of their lawful states because different laws may govern their properties. Hence, there may be structural variability between domains in terms of their lawful state spaces, even though there is a full-similarity double side between their functional schemas. For instance for IFIP conferences, we have a full similarity double side among the functional schema defining *paper* of *conference A* and the *functional schema* defining the *paper* in the *conference C*, but possible values for *status* of the paper in conference A are *Submitted, Accepted, Rejected, Short Papers* and in the conference C are *Submitted, Accepted, Rejected, Conditionally Accepted.* This shows variability in the lawful state spaces, even though their functional schemas are completely similar.

On the other hand, when considering dynamics of the domain – a process can be defined in terms of a *sequence of changes* (i.e. events) [7]. These changes may happen within things (internal events of the things) and between things (i.e. interactions between things) of a domain. The processes of a domain can be shown using the *lawful event space* of the domain and the *ordering* between these events. Hence, to investigate variability between processes of different product domains, we need to explore variability classes between their lawful event spaces of the domains and their sequences. As an example for variability in ordering of events, assume that determining the program of conference A involves an order *Workshop→ Tutorial→Main Conference* and for conference B *Main Conference → Workshop→ Tutorial*. Although both the conferences contain the same set of events, the ordering of events is different.

Table 3 shows similarity classes between two different product domains. All of the above similarity classes, except full similarity double side, show variability classes among domains. When investigating more than two domains, combination of these classes may exist between one product domain and the rest of product domains.

3.4 Variability Framework Derived From Bunge's Ontology

Having identified variability classes in section 3.3, we describe a framework for evaluating variability languages by using the ontological theory. The evaluation framework is based on the assumption that variability languages must be ontologically expressive and must be able to represent all the variability classes which may happen among the elements of conceptual models, represented in Table 3.

By investigating the variability classes in section 3.3, we specify two main concepts for variability framework: *variability sources* and *variability patterns*. A *variability source* shows elements in which variability may happen (c.f. sections 3.1 and 3.2). A *variability pattern* shows a different recurring type of variability (see section 3.1) between sets of phenomena of different product domains. Considering Bunge's ontology, *structure of a domain* including *things, properties (attributes),* and *lawful state space* and *processes of the domain* including *lawful event space* and *time of occurrence* are variability sources. The common variability patterns for both the structure and process of domains are *full-similarity one side, partial similarity*, and *complete dissimilarity.* Additionally*,* the *ordering* variability pattern is dedicated to the processes of domain and shows another aspect of difference between the sets of lawful event spaces.

Similar to ontological analysis for conceptual modeling languages [7], we identify two evaluation criteria for assessing the capability of languages to model variability – *variability completeness* and *variability clarity*. Variability completeness is concerned with investigating if modeling languages have constructs for representing all the variability patterns and consider variability in all possible sources. Variability clarity means that there is a one-to-one mapping among variability constructs in variability languages and variability patterns of the framework.

Table 3: **Similarity Classes between two product domains**

Similarity among structures of Domains			Similarity among Processes of Domains		
Functional Schema	**Lawful State Space**	**Class Name**	**Lawful Event Space**	**Sequence Difference**	**Class Name**
Full Similarity – Double Side	Full Similarity - Double side	Completely similar structures	Full Similarity – Double Side	No	Full Similarity-Double side among Process
	Full Similarity – One Side	Complete similar macro structure, different micro structure		Yes	Full Similarity-Double side among events and different order
	Partial Similarity		Full Similarity –One Side	No	Full Similarity –one Side among Process
	Dissimilarity	Complete similar macro structure and complete dissimilar micro structure		Yes	Full Similarity –one side among events and different order
Full Similarity – One Side	Full Similarity – one side	High similar macro and micro structure	Partial Similarity	No	Partial Similarity among Process
	Partial Similarity			Yes	Partial Similarity among Process and different order
	Dissimilarity	High similar macro and complete dissimilar micro structures	Dissimilarity	NA	Complete Dissimilarity among process
Partial Similarity	Partial Similarity	Medium similar macro and micro structure			
	Dissimilarity	Medium similar macro and complete dissimilar micro structure			
Dissimilarity	Dissimilarity	Complete Dissimilar Structure			

4 Analysis of Variability Languages with Variability Framework

In this section, we analyze two variability languages, i.e. feature models [1] and Orthogonal Variability Models (OVMs) [3] using the proposed variability framework.
Feature models are widely employed in SPLE to model variability and provide representations for variability relations. A central notion in identifying and modeling variability in feature-oriented software product lines is *feature*, which is defined as follows:

"Important distinguishing aspects, qualities, or characteristics of a family of systems" (Kang et al. [1]); and "a logical unit of behavior specified by a set of functional and non-functional requirements" Bosch [13]. In the feature model, variability is represented through the following variability relations: *Optional feature, Alternative feature group,* and *Or feature group.* OVMs represent variability using the variation points (VP) and variants (V) constructs [3]. A variation point is a representation of variability *subject*, "a variable item of the real-world or a variable property of such item" [3]. A variant is a representation of a variability *object,* a particular instance of a variability subject [3]. For example, a variable subject *color* is a property of a real-world item (e.g. Car) and variable objects for *color* are *green, red*, and *blue*. Then, color is a variation point and green, red, and blue are variants. In OVM, variability is specified using relations defined between variation points and variants. These relations are *optional* and *alternative choice* with cardinality *min..max [*11].
In order to analyze these two languages, based on the evaluation criteria defined in our framework, we derive a research question: *What are the representational shortcomings of feature models and OVM in light of the theoretical variability framework?*

Table 4: Analysis results of feature models and OVM using variability framework (√) match, (×) not match, (±) ambiguity

Concepts in the Framework \ Variability Languages			Feature models		OVM	
				Explanation		Explanation
Variability Source	Structure	Things	√	relating features to natural kind in Bunge's Ontology	√	relating variation point (subject) and variants (object) to natural kind in Bunge's Ontology
		Properties	√		√	
		Lawful state Space	√		√	
	Process	Event Space	√		√	
		Time	√		√	
Variability Patterns	Full Similarity One-side		√	Optional relations	√	Optional relations
	Partial Similarity		±	OR relation	±	Cardinality [m..n]
	Dissimilarity		√	Alternative Relation	√	Cardinality [1..1]
	Ordering Variability		×	Not Considered	×	Not Considered

To answer the research question, we established a mapping between constructs of these languages and the concepts in the variability framework (see Table 4).
With respect to completeness criteria, the variability languages should encompass constructs for presenting variability sources and variability patterns. Hence, we investigate if feature models and OVM can represent variability among all different sources of variability (i.e., things, properties, lawful state space, and lawful event space).

Variability Sources: Variability in feature models is represented in terms of difference among features and in OVM in terms of variation points and variants. According to definitions for features, we can conclude a feature is a particular set of properties or processes of one or more products in a product family. To interpret features based on Bunge's ontology, we can relate features to *natural kinds* because natural kinds are used to define things with a set of common properties that adhere to the same laws including both transition and state laws [8]. Hence, the natural kinds similar to features can be used to represent both the processes (lawful event spaces and time) and structure (things, properties, and lawful state space) of the domain. Considering the ways features and natural kinds can be related, we propose that a "good" set of features is required to represent all sources of conceptual variability including things, properties, lawful state space, and lawful event space. Similarly in OVM, a variation point and a variant can represent both processes (lawful event spaces and time) and structure (things, properties, and lawful state spaces) of the real-world domain. Therefore, the variation point and variant can be related to natural kinds. This means that OVM can represents all sources of conceptual variability.

Variability Patterns: To interpret variability relations in feature models and OVM, using our variability patterns, for simplicity we consider a product line with three products (e.g., conferences A, B, and C). However, the overall discussion and interpretation is applicable for product lines with any number of products. We make two assumptions. First, if a product has a feature (in feature models) or a variant (in OVM)f, then there are things in the real-world domain of the product belonging to natural kind nk_f. Second, we assume that all products involved in a variability relation, contain *thing(s)* in their corresponding real-world domain which are mapped to *thing(s)* in the real-world domain of the other involved products.

Optional Relation: If a feature (variant in the OVM) f is optional then one or two of the products in the example product line contain f. Consequently, things in the real-world domain of products containing feature (variant in OVM) f belong to natural kind nk_f. However, things in the real-world domains of the other products (i.e., products without optional feature or variant f) do not have a set of properties and processes defined by nk_f. This means there is a full similarity one side with respect to nk_f among things which have mapping to the real-world domains of the products belonging to the product line.

Alternative Relation (Alternative choice with cardinality 1..1): In an alternative relation (i.e. XOR group f_1, f_2, f_3), from a group of alternative features (variants in the OVM), each product contains only one of the features (variants in the OVM). This means that there are things in the domain of products involved in the alternative group such that these things belong to natural kinds nk_{f1}, nk_{f2} and nk_{f3}. The mapped things of different products involved in the alternative relation are completely dissimilar with respect to the properties and laws defined in $nk_{f1}, nk_{f2},$ and nk_{f3}.

OR relation (alternative choice with cardinality *min..max*): A group of features connected by the OR relation means that products in the group have one or more features. Similarly, in OVM, a group of variants with cardinality *min..max* refers the selection of at least *min* and at most *max* number of variants. To interpret these patterns, all the variability patterns, including complete dissimilarity, partial similarity, and full-similarity one side may happen (but do not have to) among mapped things of each pair of products. Therefore, an OR group in feature models and alternative choice with cardinality *min..max* in OVM can be mapped into complete dissimilarity, partial similarity, and full-similarity one side variability patterns.

Ordering Variability: Neither feature models nor OVM has constructs for representing ordering variability between features and variants, respectively. Consequently, feature models and OVM cannot represent part of process variability classes shown in table 3.

Based on our analysis, we conclude feature models and OVM have the same representational expressiveness for modeling conceptual variability. Also, both languages have the lack of variability completeness as they do not have any construct for representing *ordering* variability. Finally, due to the ambiguity in OR and alternative choice with cardinality *min..max*, we cannot establish one–to-one mapping between variability patterns in our framework and these variability relations. This is a lack of *variability clarity* with respect to our ontological framework.

5 Related Work

Similar to our work, Reinhartz-Berger et al. [14] conducted an ontological analysis of variability modeling using Bunge's ontology. They analyzed process variability and identified a set of variability patterns in the behavior of products. They used their variability framework to perform a formal operational feasibility analysis. In our study, we have investigated both structural and behavior variability which need to be modeled using variability languages. Moreover, our framework is used to evaluate expressiveness of variability languages for representing types of differences among products.

Several frameworks have been employed for evaluating variability languages from practical points of view. Specifically, two main works have been done to evaluate feature models [4][5]. Diebbi et al. [5] performed a study in an company and established a set of criteria by studying software engineers' main expectations for a variability notation. Heymans et al. [4] developed a formal quality method based on SEQUAL (SEmioticQUALity) for evaluating the expressiveness of feature models. In comparison to these works, instead of practical consideration we employed a theory-based approach which can better reveal complete and non-redundant set of variability types.

6 Conclusion and Future Works

Variability plays a pivotal role in systematic reusability in SPLE. This prompts a need for new methods for detailed and formal analysis of the variability notion. However, existing variability formalizations are mainly practice-oriented and lack theoretical foundation. Trying to address this challenge, our work explores the notion of product line variability using the ontological theory and provides a theoretical analysis of variability modeling. Ontological understanding of variability is beneficial from several aspects. First, based on the results of ontological analysis, software developers can clearly identify sources of essential variability in information systems and consider those sources during domain engineering lifecycle. Second, the possible patterns of variability (i.e., types of differences) among products in terms of structure and processes can be identified. This can support the evaluation of variability modeling languages and the improvement of their expressiveness in representing variability. This work is part of continuing work for developing theoretical foundations for variability modeling and configuration decisions in software product lines. We intend to enrich the variability framework with patterns and sources of technical variability.

References

[1] Kang, Kyo C., Sholom G. Cohen, James A Hess, William E. Novak, and A. Feature-Oriented Domain Analysis (FODA) Feasibility Study, Technical Report CMU, 1990.

[2] Pohl, K., Metzger, A.: Variability management in software product line engineering. Proc. 28th Int'l Conf. Software engineering. pp. 1049–1050 (2006).

[3] Pohl,K.,Böckle,G., van der Linden,F.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[4] Heymans, P., Schobbens, P.Y., Trigaux, J.C., Matulevicius, R., Classen, A., and Bontemps, Y. Towards the comparative evaluation of feature diagram languages. Proc. Software and Services Variability Management Workshop, (2007).

[5] Djebbi, O.,Salinesi, C. Criteria for Comparing Requirements Variability Modeling Notations for Product Lines. 4th Int'l WShComparative Evaluation in Req. Eng. (2006), 20-35.

[6] Bunge, M. (1977) Treatise on Basic Philosophy. Vol. 3. Ontology I: The Furniture of the World. Reidel. Boston, Massachusetts

[7] Wand, Y. and Weber, R. On the deep structure of information systems. Information Systems Journal 5, 3 (1995), 203-223.

[8] Evermann, J. and Wand, Y. Ontology based object-oriented domain modelling: fundamental concepts. Requirements engineering 10, 2 (2005), 146–160.

[9] Van Gurp, J., Bosch, J., and Svahnberg,M.: "On the notion of variability in software product lines," in Proc.Working IEEE/IFIP Conf. Software Architecture, 2001, p. 45-54.

[10] Weiss, D.M., Lai, C.T.R.: Software Product-Line Engineering, A Family-Based Software Development Process. AddisonWesley, 1999

[11] Metzger, A., Pohl,K., Heymans, P., Schobbens, P.-Y., and Saval, G.: "Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis," in RE 2007, Delhi, India, 2007, pp. 243-253.

[12] Olle, T.W., Stuart, A.A.V., Sol, H.G.: Information Systems Design Methodologies; A Comparative Review: Proceedings of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies, (1982).

[13] Bosch, J., Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach. ACM Press / Addison-Wesley, 2000.

[14] Reinhartz-Berger, I., Sturm, A., Wand, Y.: External Variability of Software: Classification and Ontological Foundations. In Proc. Conceptual Modeling Conf. pp. 275-289. (2011).