Variability Analysis of Requirements: Considering Behavioral Differences and Reflecting Stakeholders' Perspectives

Nili Itzik, Iris Reinhartz-Berger, and Yair Wand

Abstract—Adoption of *Software Product Line Engineering* (SPLE) to support systematic reuse of software-related artifacts within product families is challenging, time-consuming and error-prone. Analyzing the variability of existing artifacts needs to reflect different perspectives and preferences of stakeholders in order to facilitate decisions in SPLE adoption. Considering that requirements drive many development methods and activities, we introduce an approach to analyze variability of behaviors as presented in functional requirements. The approach, called semantic and ontological variability analysis (SOVA), uses ontological and semantic considerations to automatically analyze differences between initial states (preconditions), external events (triggers) that act on the system, and final states (post-conditions) of behaviors. The approach generates feature diagrams typically used in SPLE to model variability. Those diagrams are organized according to perspective profiles, reflecting the needs and preferences of the potential stakeholders for given tasks. We conducted an empirical study to examine the usefulness of the approach by comparing it to an existing tool which is mainly based on a latent semantic analysis measurement. SOVA appears to create outputs that are more comprehensible in significantly shorter times. These results demonstrate SOVA's potential to allow for flexible, behavior-oriented variability analysis.

Index Terms—Software product line engineering, variability analysis, feature diagrams, requirements specifications, ontology

1 Introduction

Manufacturers have long employed techniques to create product lines using common factories that assemble and configure parts designed to be reused across similar products. Because software has become an essential part of almost any business, the need to develop software quickly and cheaply has grown. This has led to introducing Software Product Line Engineering (SPLE) for systematically reusing knowledge and software-related artifacts within product families [12]. SPLE has proven successfully in reducing development cost, time-to-market, and improving product quality [31], [34]. It does however require investment to develop reusable artifacts, termed *core assets* or *domain artifacts*. Moreover, effort is required in adapting these artifacts to develop different members of the Software Product Line (SPL) [12]. Hence, SPLE is often not adopted initially, but only after development of successful products in the family [5]. In such cases, software-related artifacts need to be adapted to fit the SPLE framework.

A basic activity of SPLE is variability analysis [39], whose goal is identifying and determining the precise differences among the software products in the family. These differences are usually specified and documented using variability modeling. A notation commonly used to model variability both in research [9] and in industry [5] is feature diagrams. These are tree or graph structures that describe the characteristics of an SPL and the relationships and dependencies among these characteristics [27]. Outcomes of variability analysis have many uses [5], including management of existing variability, product configuration, requirements specification, derivation of products, and design or architecture of products. In all these uses, representing variability accurately and comprehensibly is of high importance to the success of the related tasks.

Several methods have been suggested for analyzing variability of software-related artifacts, mainly within the field of SPLE. Among the different software-related artifacts – software requirements are of special interest because they drive the various development processes including: analysis, design, implementation, and testing. They represent the expectations of the requested system by different stakeholders, such as developers, users, and customers. Requirements analysis is also relevant to many development methods, including agile ones (through concepts such as user stories). It further enables reducing technological considerations and focusing on "essential" variability, namely variability visible to customers and end users [20].

Existing studies analyze variability of requirements based on their syntactic and semantic similarity. These similarities enable grouping of requirements in fixed and predefined views. The approaches in these studies have several deficiencies. First, while using syntax and semantics, they do not necessarily make distinctions between different parts of behaviors as manifested in the requirements. These parts

- *N. Itzik and I. Reinhartz-Berger are with the Department of Information Systems, University of Haifa, Haifa 3498838, Israel. E-mail: nitzik@campus.haifa.ac.il, iris@is.haifa.ac.il.*
- *Y. Wand is with the Sauder School of Business, University of British Columbia, Vancouver, BC V6T 1Z4, Canada. E-mail: yair.wand@ubc.ca.*

Manuscript received 31 Mar. 2015; revised 12 Nov. 2015; accepted 15 Dec. 2015. Date of publication 24 Dec. 2015; date of current version 22 July 2016. Recommended for acceptance by R. Lutz.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSE.2015.2512599

include preconditions for changes, triggers that cause changes, and final conditions. Inadequate distinctions between these parts can result in different requirements that appear similar, or in similar requirements that look different. Second, the fixed and predefined views by which current approaches represent variability may limit the ability of stakeholders to conduct variability-related tasks. For example, sales persons interested in differences between software products may have difficulties in identifying the required variability. These stakeholders may find variability of classes, components, and attributes, but will actually prefer viewing variability of functionality (actions, functions, and services). Third, if similar requirements are expressed in different levels of detail, they may appear semantically quite different although they will fundamentally be the same.

To address the above deficiencies, we propose an approach for analyzing variability using the behavior of software products. The method, called semantic and ontological variability analysis (SOVA) has two main contributions. First, it enables analyzing behavior from an external point of view. This reduces the apparent differences that may arise due to differences in the level of details. As well, it reduces the appearance of semantically similar phrases that specify different parts of the behavior (e.g., "the item is available", which can specify a precondition or a post-condition of a behavior). Second, SOVA takes into account the stakeholders' perspectives by using parameters that reflect their needs and preferences in analyzing variability.

The inputs for SOVA are requirements documents written in plain text. Each requirement may represent a use case, a user story, or any unit describing a single behavior—expected or existing—of a software product. Using Bunge's ontological model [6], [7], we define behavior in terms of the initial state of the system before the behavior occurs (namely, the behavior's preconditions), the external events that trigger the behavior, and the final state of the system after the behavior occurs (i.e., its post-conditions). Behavior is then analyzed using natural language processing (NLP) techniques and semantic measures. The differences between the initial states, external events, and final states of various behaviors are identified. The outcomes are organized in feature diagrams that can be generated according to different perspectives which reflect stakeholders' needs and preferences. The resulting feature diagrams are behavior driven and lay the foundation for flexible variability analysis.

This paper extends earlier papers published on this method. In [41] we concentrated on the similarity measure of SOVA and its performance with respect to a latent semantic analysis (LSA) measurement. In [25], we presented the supporting tool of SOVA, and in [24] we emphasized the flexibility of the method to generate various feature diagrams based on stakeholders' needs and preferences. In the current paper we focus on the considerations of SOVA with respect to variability analysis, and present empirical evaluation of its usefulness in supporting different variability-related tasks. The evaluation demonstrated high comprehensibility of SOVA's outcomes.

The rest of this paper is structured as follows. Section 2 provides the required background and reviews the literature about generating variability models from textual descriptions and organizing variability models. Section 3 provides the motivation for our research, while Section 4 describes how SOVA deals with the concerns of variability analysis. Section 5 presents and discusses SOVA's empirical evaluation. Section 6 concludes by summarizing this research and raising issues for future research.

2 Background and Related Work

Creating SPLs is expensive and complex, so companies typically begin by developing individual products. Only after similar products show success is the adoption of SPLE considered. This adoption often follows the "extractive" approach, which generates common and varying artifacts from individual software systems [5], [30]. In such scenarios, analyzing the variability of existing artifacts has a crucial role in the development of core assets, namely artifacts intended to be (re) used by different members of the SPL.

Prior studies have suggested creating variability models from various development artifacts. Due to our special interest in requirements, Section 2.1 reviews studies that focus on generating variability models from textual descriptions, such as requirements documents and product descriptions. Section 2.2 refers to studies dealing with organizing variability models and especially feature diagrams.

2.1 Generating Variability Models from Textual Descriptions

Recently, a systematic literature review [3] has been conducted on feature extraction from requirements expressed in a natural language. The main conclusions of this review are: (1) most studies use software requirements specifications (SRS) as ***inputs***, but product descriptions, brochures, and user comments are also used due to practical reasons; (2) commonly the ***outputs*** are feature diagrams, clustered requirements, keywords or direct objects; and (3) the ***extraction process*** can be divided into four phases: requirements assessment, terms extraction (using different techniques, such as algebraic models, similarity metrics, and NPL tools), features identification, and model formation.

In the context of our research, Table 1 lists existing studies, focusing particularly on the methods they use for calculating similarity, for structuring hierarchy, and for inferring variability, as well as on the variability perspectives they highlight in the generated outputs. For providing a complete picture, we further mention their input and output formats. Next, we briefly refer to these characteristics in the reviewed studies.

The first group of studies assumes unique terminology, meaning that similar features have the same name. Acher et al. [1] present a semi-automated method in which inputs are tabulated sets of product descriptions. These sets represent valid combinations of features in families of products. The outcomes, namely the generated feature diagrams, represent variability in objects, classes, components, or services of the product family, depending on the provided inputs. The method follows rules of conversion and transformation to structure the hierarchy of diagrams, and graph analysis techniques to infer variability.

TABLE 1
Generating Variability Models from Textual Descriptions

Source	Input Format	Output Format	Similarity metric	Hierarchy Structuring	Variability inference	Variability Perspectives
Acher et al. [1]	Product descriptions (tabular format)	Feature diagrams	Identity	Conversion & transformation rules	Graph analysis	Provided features
Niu and Easterbrook [38]	Requirements documents	OVM	Identity		Expert knowledge, linguistic clues, heuristics	Nouns and verbs
Ferrari et al. [17]	Brochures of vendors	Common & variable features	Identity		Graph analysis	Nouns[1]
Dumitru et al. [16] Hariri et al. [21]	Product descriptions & feature descriptors	Feature clusters	Syntactic	Clustering	Frequent item set graph	Nouns, verbs and adjectives
Darvil et al. [14]	Product descriptions	Feature diagrams	Syntactic	Implication graph & clustering	Graph analysis	Nouns, verbs and adjectives
She et al. (2011) [44]	Features descriptions & propositional formulas	Feature diagrams	Syntactic	Heuristics & Synthesis	Graph analysis	Provided features
Alves et al. [2]	Requirements documents	Feature diagrams	Semantic	Clustering	Grammatical patterns	Entire text
Weston et al. [49]	Requirements documents	Feature diagrams	Semantic	Clustering	Expert knowledge, grammatical patterns	Entire text
Chen et al. [10]	Requirements documents	Feature diagrams	Classification without a metric	Clustering	Graph analysis	Objects and verbs

[1] *Can be extended to verbs, developing a new set of patterns.*

Niu and Easterbrook [38] introduce a semi-automatic method for constructing orthogonal variability models (OVM)[1] [39] from requirements documents. This method extracts functional requirements profiles, represented as "verb-direct object" pairs, using expert knowledge and linguistic clues. Heuristics are further used for inferring variability. Here again, features are considered similar only if their names are identical. No hierarchy is structured, because OVM models are not hierarchical.

Ferrari et al. [17] suggest extracting common and variable features from public brochures using structural part-of-speech patterns. While currently suggesting patterns based on nouns, it is claimed that the method can be extended to represent functionality as well, but this requires new verb-based patterns to be developed. The method uses a NLP approach for the extraction of terms. Based on their occurrences in the different documents, the terms are considered as either domain specific or of general purpose. The method does not support structuring the extracted features into diagrams.

When the terminology is not unique, then similarity becomes a main anchor to analyze variability. Similarity can be calculated using syntactic measures such as cosine similarity[2] [19]. Alternatively, different semantic metrics can be utilized. Those metrics are commonly classified as corpus-based or knowledge-based [19], [36]. Corpus-based measures identify the degree of similarity based on information derived from large corpora (e.g., [8], [32] and [46]). Knowledge-based measures, on the other hand, use information drawn from semantic networks. Many of these methods use WordNet [51] – a lexical database for the English language – for measuring similarity of terms or concepts.

Using cosine similarity, Davril et al. [14] presented a fully automated approach for utilizing publicly available repositories of product descriptions. This work is based on the feature extraction technique presented in [16], [21] that suggests using online product descriptions for creating frequent item set graphs and product-by-feature matrices. These outcomes are utilized to create feature diagrams by

1. OVM represents variability separately (orthogonally) from the development artifacts, which can be specified using different modeling languages such as UML. Variability is modeled in terms of variation points (locations at which variability occurs) and variants (ways to realize variability). Dependencies between variation points and variants, as well as trace links to development artifacts, are also specified in OVM.

2. This similarity measures the cosine angle between two vectors of an inner product space.

using a clustering algorithm and conditional probabilities between occurrences of features. In all three studies TF-IDF[3] is used to assign weights to each term in the similarity calculation. Nouns, verbs and adjectives are utilized to extract features and infer variability.

She et al. [44] introduce a tool-supported approach for reverse engineering of feature diagrams from textual feature descriptions and feature dependencies. The feature descriptions are extracted from documentation, processor symbols, or code comments, whereas the feature dependencies are specified in propositional formulas. The similarity metric applied is syntactic and can handle very short descriptions where one shared word is significant. Feature hierarchy is created using ranking heuristics to identify the most likely parent of each feature. Variability is inferred by constructing and analyzing an undirected graph, with features such as vertices and edges denoting mutually exclusive relations between features. The features are provided as inputs, and hence the view of variability is fixed.

Alves et al. [2] suggest creating feature diagrams from textual requirements, using LSA [32] and vector space model (VSM). LSA is a well-known corpus-based semantic method that analyzes the statistical relationships among words in a large corpus of text. VSM is used to represent documents as vectors of words. The weight of a word is represented by the frequency of the word in the document, and cosine similarity between vectors is calculated. The hierarchy is structured using a clustering approach. Variability is inferred by searching for grammatical patterns and words denoting differences (e.g., "optionally," "alternatively," "at least one of"). The entire text of the requirements is used for analyzing variability. Weston et al. [49] propose an extension to the method of Alves et al., named ArborCraft. The tool uses NLP techniques as well as mining techniques for finding potential variable elements within the input documents. This tool creates feature diagrams by grouping similar requirements using LSA and a hierarchical agglomerative clustering algorithm. Feature variants are then identified using an XML-based language called requirements description language (RDL). RDL enriches the requirements specification with semantic information (such as actions, objects, and relations) derived from the semantics of the natural language. Here again the entire requirements text is used for variability analysis.

The last group of works does not use specific similarity metrics. Chen et al. [10] suggest analyzing the relationships between individual requirements, based on the resources that store the data accessed (modified or read) by requirements. In this method, tightly related requirements are clustered into features and the features are described as objects and verbs.

To summarize, the studies reviewed above use unique terminologies, employ syntactic or semantic similarity measures, or do some classifications. They further generate outcomes that are derived from pre-defined perspective of variability (most notably using nouns, verbs, or verbs and direct objects). As we will show later, these characteristics may impose limitations when analyzing variability of functional requirements.

3. TF-IDF is a statistical measure used to evaluate the importance of a word for a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the words in the corpus.

2.2 Organizing Variability Models

Many variability models, such as feature diagrams, are structured hierarchically. How to organize such hierarchies has been the focus of several studies where some of these studies suggest using different views and integrating them in order to address scalability issues. In this context, a view is a simplified representation of a feature model tailored for a specific stakeholder, role, or task [23]. Hubaux et al. [23] suggest techniques to specify, visualize, and verify the coverage of a set of views. To this end, they define a multi-view feature model as a set of views, each of which consists of a set of features. Coverage means that all features appear in at least one view and if not—the decisions on the features that do not appear in any view can be inferred from the decisions made on the features that are part of the view. Clarke and Proenca [11] refer to a view as a disjoint set of features and abstractions (namely, sets of features "hidden" behind a label meaningful to the user). They formally define compatibility properties between views and their combinations. Czarnecki et al. [20] refer to feature models as a "notational subset of ontologies" or as specific views on ontologies. This perspective suggests two potential approaches to feature modeling: (1) view projection, in which a comprehensive ontology is available or being constructed using ontology-oriented domain analysis; (2) view integration, in which feature models are created more or less independently, with some ontology in the mind of each developer. An ontology can then be used to align the views.

Some studies focus on organizing variability models according to stakeholders' needs and preferences. Early studies mainly distinguish between user-oriented and technical features, e.g., [28]. Czarnecki et al. [13] propose the concept of stage-configuration, in which the customization of feature models is partitioned into several stages. At each stage, different stakeholders make their own configuration decision on the feature model, generating a more specified feature model for the next customization stage. Zhao et al. [53] suggest clustering information scattered in a feature model into separate views according to different stakeholders' viewpoints. The suggested process includes three steps: scoping views for stakeholders, defining features in views, and organizing features in view.

All these studies deal with specification, visualization, and integration of perspectives (views) and not with their extraction from textual requirements. Herrejon-Lopez et al. [22] suggest assessing search-based techniques for reverse engineering feature models. Using the different feature models that can be reversed engineered from descriptions of individual variants, three standard techniques (evolutionary algorithms, hill climbing, and random search) are compared. The focus is on precision and recall of these techniques and not on usefulness of the retrieved models. Bécan et al. [4] define a generic, ontology-aware synthesis procedure that computes the likely siblings or parent candidates for a given feature. They develop six heuristics for clustering and weighting the logical, syntactical and semantical

TABLE 2
Examples of Requirements from E-Shop Applications

#	Requirement pairs	LSA	MCS
1.	• A customer should be able to purchase products from different stores via the web application. • The web application should be able to store different products that a customer purchased.	0.95	0.91
2.	• The system will allow introducing different products to the catalog. • Different products are allowed to be added to the product catalog.	0.56	0.65
3.	• After reviewing the order details, the supplier needs to confirm the order. • When a supplier reviews the order details, the system verifies the provided authorization code. If the code is valid, the system generates an order details report. Only after the supplier confirms the order, the system changes the order status.	0.43	0.73

relationships between feature names. This procedure can be used when synthesizing a feature model from a set of features' dependencies.

We next exemplify the drawbacks of the above approaches, which mainly rely on textual and linguistic aids, and motivate the need for behavioral variability analysis of functional requirements as well as for the organization of the corresponding outcomes according to behavioral characteristics.

3 Motivation

To demonstrate and motivate the need for improvement, consider the pairs of requirements of e-shop applications presented in Table 2. For each pair, we further present the values of two different semantic similarity measures: LSA[4] [32] and MCS[5] by Mihalcea, Corley, and Strapparava [36].

The first pair of requirements shows high semantic similarity (0.95 and 0.91 for LSA and MCS, respectively). It is clear, however, that these requirements are quite different. The first requirement represents a behavior triggered by an external user (a customer) who needs support for purchasing products from different stores. The second requirement represents an internal behavior that probably aims at tracking purchase activities of customers.

The second pair of requirements may seem to be more different from the first pair, both in LSA and MCS (resulting in values of 0.56 and 0.65, respectively). However, the two requirements represent very similar domain behaviors regarding insertion to a catalog.

The third pair of requirements tackles the observation that requirements can be described in different levels of detail, whereas the stakeholder may be interested in analyzing the variability from an external point of view. From this external point of view the two requirements can be considered very similar, as they both require reviewing the order details and confirming the order. The semantic similarities of the requirements result in confounding similarities: 0.43 for LSA and 0.73 for MCS. This happens primarily due to the differences in the levels of detail of intermediate actions.

Besides the limitations in similarity calculations demonstrated above, existing methods commonly generate a single or a set of very similar variability models for a given set of inputs. Thus, the generated models may be less suitable for performing different variability-related tasks. As an example, consider Figs. 1 and 2. Both diagrams represent variability of the same set of requirements of e-shop applications. Fig. 1 focuses on the various actions in the applications, such as purchasing, paying, order cancelation, and product return. We refer to such a view as functional. Based on this diagram, it is relatively easy to answer questions such as, "What types of shipment are supported/mandatory?" and "When can an order be cancelled?" Fig. 2, on the other hand, refers primarily to objects, e.g., catalog, shopping cart, payment type, and inventory. We refer to such a view as structural. This diagram can help answer questions about the types of payment that are supported in the applications, and which catalogs are supported. It would be very difficult, if not impossible, to answer function-related questions via a diagram that represents a structural view and vice versa. We therefore suggest improving the existing corpus of literature by introducing a behavioral method of variability analysis that reflects different stakeholders' perspectives based on behavioral characteristics.

4 Semantic and Ontological Variability Analysis

The proposed variability analysis method is called SOVA. SOVA introduces ontological considerations to extract different parts of behavior specification, and semantic measures to calculate the similarity of those parts. It further allows controlling the views (called perspectives) of variability models based on behavioral characteristics. SOVA is composed of three main stages depicted in Fig. 3: text parsing, behavioral similarity calculation, and feature diagram construction. The first stage parses the inputs, which are textual descriptions representing behaviors, i.e., functional requirements of the products. Each input document (or description) represents a different product in the SPL and consists of statements (i.e., paragraphs) describing requirements. Each statement, which may be composed of one or more sentences, outlines a different behavior of the product. Parsing is done mainly by extracting the pre-conditions, triggers, and post-conditions of a behavior utilizing an ontological model and taking into account semantic roles in the requirement text. The parsed

4. As noted, LSA is a well-known corpus-based semantic method. Sentence similarity is computed as the cosine of the angle between the vectors representing the sentences' words. We used LSA implementation that can be accessed via http://lsa.colorado.edu/.

5. Combining corpus- and knowledge-based semantic approaches, MSC calculates sentence similarity by finding the most similar words in the same part of speech class. The derived word similarity scores are weighted with the inverse document frequency scores that belong to the corresponding word. We used MCS implementation that is part of the SEMILAR (a semantic similarity toolkit) online demo at http://www.semanticsimilarity.org/.

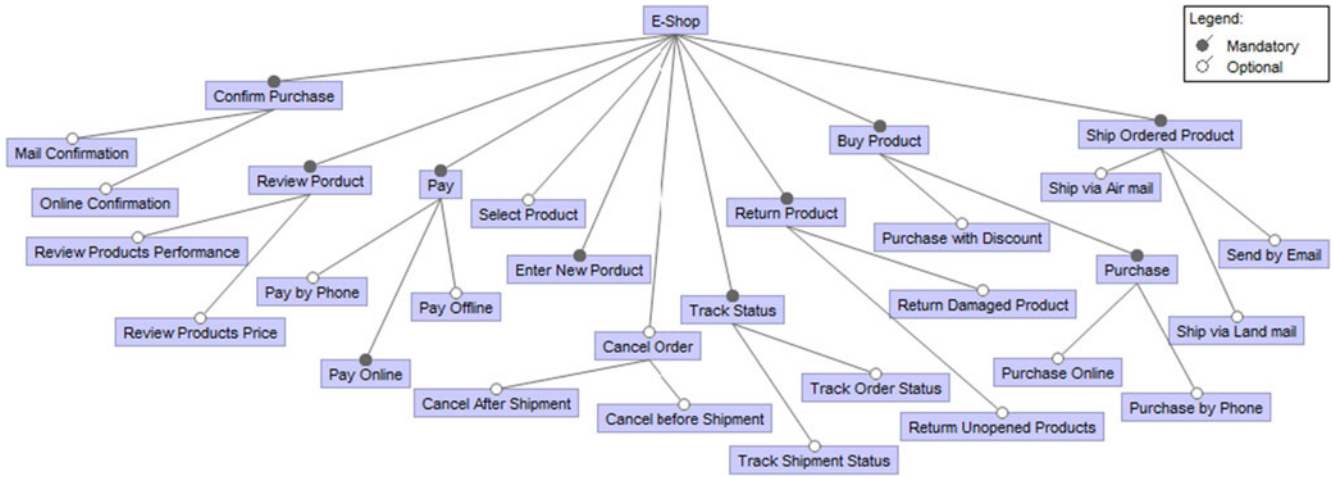

Fig. 1. An e-shop feature diagram—a functional perspective.

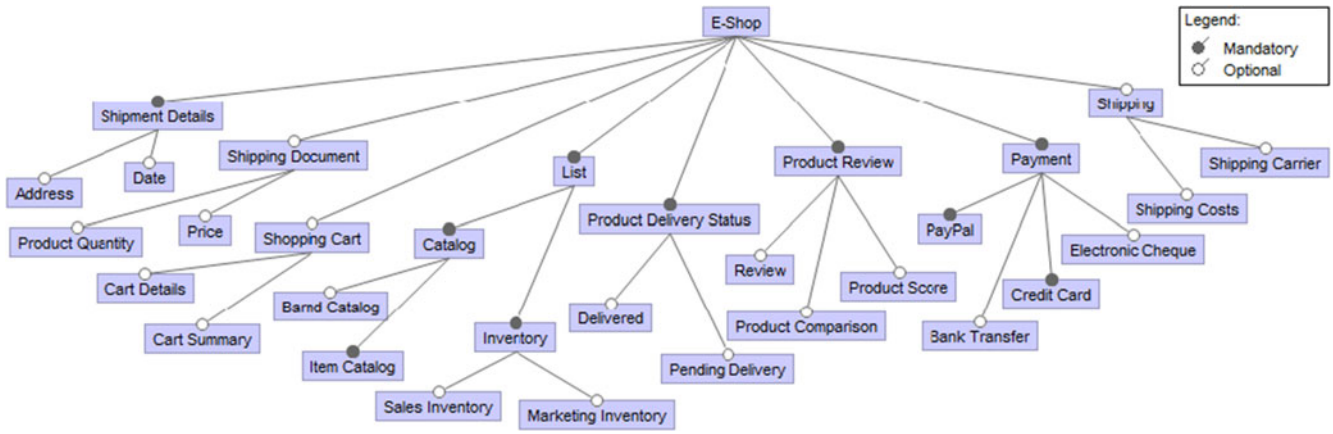

Fig. 2. An e-shop feature diagram—a structural perspective.

text is then used to calculate the behavioral similarity, following different perspective profiles. These profiles include parameters and weights that reflect the variability-related task and the stakeholder's preferences. Finally, the outputs (the generated feature diagrams) are structured on the differences among the input requirement documents and consider the perspective profiles. The following sub-sections detail each stage, elaborating on the variability analysis considerations. Formal versions of the definitions for this section are provided in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2015.2512599.

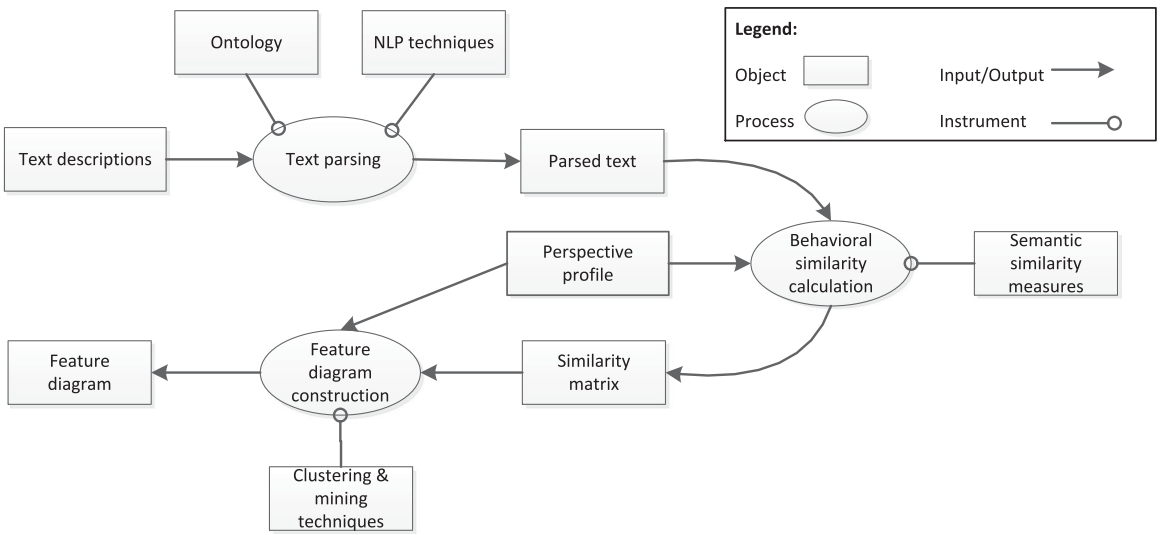

Fig. 3. An overview of the process in SOVA.

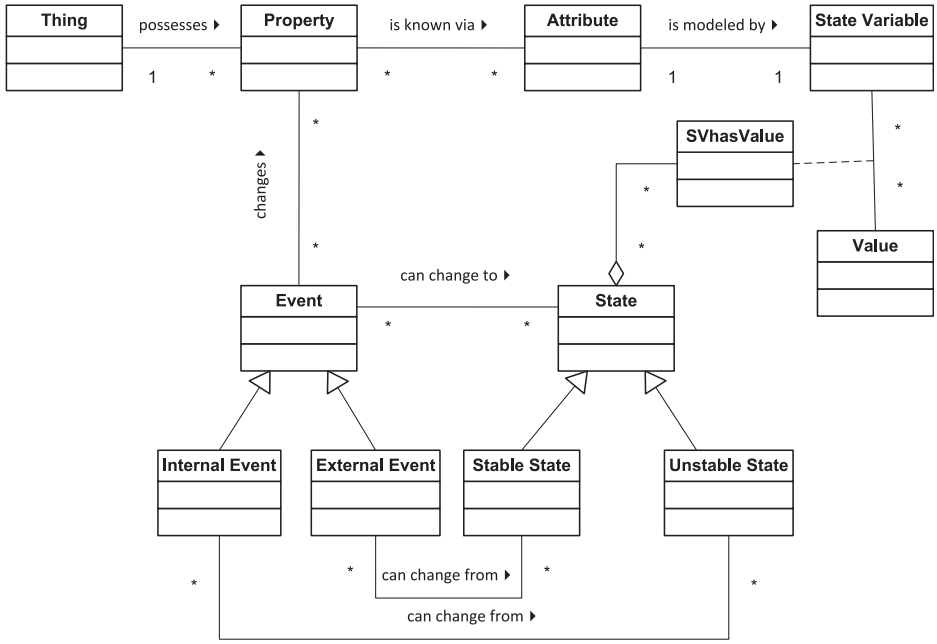

Fig. 4. Bunge's ontological model.

4.1 Text Parsing

To parse the behavioral parts of the requirements, we use Bunge's ontological model [6], [7]. We chose Bunge's ontology because it places an emphasis on the notion of systems, defines behavior-related concepts such as states and events in a well-formalized axiomatic way, and has been adapted to conceptual modeling in the context of systems analysis and design [47], [48]. According to this model, the world is made of *things* that possess *properties*. Properties are known via *attributes*, which are characteristics assigned to *things* by humans. *State variables* are functions that assign values of specific ranges to attributes of things, at given points of time. A thing may exist in different states, but at a specific point in time a thing will be in exactly one *state*, defined as the vector of values of state variables of the thing at that point in time. A state can be stable or unstable. A *stable state* can be changed only by other things. An *unstable state* is changed by the thing itself. An *external event* is a change in the state of a thing as a result of an action of another thing. Fig. 4 summarizes Bunge's ontological model.

The emphasis that Bunge's ontological model places on the notions of states and events makes it appropriate for modeling requirements behavior. In [42] and [43], a behavior is defined as a triplet $(s_1, \langle e_i \rangle, s^*)$, where s^* is the first stable state the thing reaches when it is in the stable state s_1 and the sequence of external events $\langle e_i \rangle$ occurs. s_1 is termed the *initial state* of the behavior and s^*—the *final state* of the behavior. Based on this definition depicting an external view of behavior, or how behavior is perceived from a user point of view, eight (external) variability classes were introduced in [43]. Those classes are summarized and exemplified in Table 3. The notation $s_1 \xrightarrow{\langle e_i \rangle} s^*$ corresponds to the behavior $(s_1, \langle e_i \rangle, s^*)$.

In order to identify the different components of a behavior (initial state, external events, and final states) and analyze their variability, we utilize semantic role labeling (SRL) [18]. This technique enables detection of the semantic arguments associated with the predicate or the verb of a sentence and their classification into specific roles. We used six semantic roles that have special importance to functionality and thus can be used to extract the different components of a behavior. Those roles are listed and explained in Table 4.

To exemplify the different roles, consider the following sentence: "When supplying an order, the supplier has to inform the customer on each item if it is available." The phrase "when supplying an order" represents a modifier emphasizing the temporal condition of the action (AM-TMP). The words "supplier" and "customer" respectively represent the agent (A0) and object (A1) of the action ("inform"), while the phrases "on each item" and "if it is available" respectively represent the action's instrument (A2) and adverbial modifier (AM-ADV). Note that SOVA uses SRL rather than methods based on parts of speech (e.g., nouns and verbs) in order to be able to differentiate among the same parts of speech that appear in various semantic roles. For example, a noun that serves as an agent versus the same noun that serves as an object.

Each phrase of a sentence in the input documents is considered a behavioral vector, either an action or a non-action vector.

Definition 1. *An* action vector *represents an activity (identified by a verb) in the behavior. It is based on a verb or a predicate and potentially combines agents, objects, and instruments.*

Definition 2. *A* non-action vector *originates from a (temporal or adverbial) modifier. It is not associated with a verb or a predicate, and describes temporal constraints and other conditions (see last two rows in Table 4).*

Definitions 1 and 2 in Appendix A, available in the online supplemental material, formalize the concepts of action and non-action vectors. To demonstrate these concepts, consider the following requirement that describes a typical use case of ordering in e-shop applications.

> *When the products list is displayed, a registered customer can order from the list an item which is on sale. She*

TABLE 3
External Variability Classes Based on System Behaviors

#	s_1	<e_i>	s^*	Class Name	Examples from the domain of e-shop applications	Comments
1.	~	~	~	Completely similar behaviors	$ready \xrightarrow{<order\ product>} ordered$ $ready \xrightarrow{<order\ product>} handled$	In both cases orders are handled.
2.	~	X	~	Similar cases and responses, different interactions	$ready \xrightarrow{<online\ order>} ordered$ $ready \xrightarrow{<phone\ order,\ fax\ approval>} ordered$	In both cases orders are successfully processed even though the events are different (online order vs. phone order + fax approval).
3.	~	~	X	Similar triggers, different responses	$ready \xrightarrow{<order\ product>} ordered$ $ready \xrightarrow{<order\ product>} failed$	Although having the same starting point and interactions, the behaviors end differently (ordered vs. failed).
4.	~	X	X	Similar cases, different behaviors	$ready \xrightarrow{<online\ order>} ordered$ $ready \xrightarrow{<phone\ order,\ fax\ approval>} failed$	Behaviors start similarly, but different sets of events (online order vs. phone order + fax approval) yield different responses (ordered vs. failed).
5.	X	~	~	Different cases, similar behaviors	$not\ shipped \xrightarrow{<cancel\ order>} canceled$ $shipped \xrightarrow{<cancel\ order>} canceled$	Behaviors end similarly with the same set of events occurring in different initial states (shipped vs. not shipped).
6.	X	X	~	Different triggers, similar responses	$not\ shipped \xrightarrow{<cancel\ order>} canceled$ $shipped \xrightarrow{<return\ product>} canceled$	Behaviors end similarly with different sets of events (cancel order vs. return product) occurring in different initial states (shipped vs. not shipped).
7.	X	~	X	Different cases and responses, similar interactions	$not\ shipped \xrightarrow{<cancel\ order>} canceled$ $shipped \xrightarrow{<cancel\ order>} failed$	Behaviors react to the same set of events that occur at different initial states (shipped vs. not shipped) and yields different responses (canceled vs. failed).
8.	X	X	X	Completely different behaviors	$not\ shipped \xrightarrow{<cancel\ order>} canceled$ $shipped \xrightarrow{<return\ product>} failed$	Behaviors react to different sets of events that occur at different initial states and yield different responses.

$\sim = similar, X = different.$

provides payment details after she enters the shipping preferences. If the payment details are valid, the system generates an order confirmation note.

Table 5 presents the six behavioral vectors derived from this requirement. Vectors 1-4 and 6 are action vectors, while vector 5 represents a non-action vector, namely, an adverbial precondition for the last action vector (6). Note that the first vector is an action vector whose source is a temporal modifier (specifying when the action occurs), and thus it also represents a precondition. Both preconditions in this example (vectors 1 and 5) have unknown agents as they are phrased as passive.

Besides the SRL used for semantically labeling the requirements, three additional NLP techniques are used in order to complete the parsing before comparing the behaviors and their components.

(1) Replace pronouns with their anaphors (i.e., the nouns to which they refer) in order to enable accurate comparison of vectors that refer to the same noun. This can be done using algorithms such as the one suggested in [40]. This algorithm changes the agent "she" in the example above to "a registered customer."
(2) Eliminate descriptive parts in order to abstract away "minor variations." This is done by using techniques

TABLE 4
The Semantic Roles Used in SOVA

Label	Role	Assigned to	Relations to requirements	Examples from the sentence
A0	Agent	Agents, causers, or experiencers	Who performs?	Supplier
	Action	Predicate or verb	What is performed?	Inform
A1	Object (patient)	Undergoing state change or being affected by the action	On what objects is it performed	Customer
A2	Instrument	Instruments, attributes	How is it performed?	On each item
AM-TMP	Temporal modifier	Time indicators specifying when an action took place	When is it performed?	When supplying an order
AM-ADV	Adverbial modifier	Temporally related (modifiers of events), intentional (modifiers of propositions), focus-sensitive, or sentential (evaluative, attitudinal, viewpoint, performative)	In what conditions is it preformed?	If it is available

TABLE 5
Examples of Behavioral Vectors

#	Agent	Action	Object	Instrument	Modifier	Source
1	[1]	is displayed	the products list			AM-TMP
2	a registered customer	Orders	an item	from the list		
3	[a registered customer][2]	Provides	payment details			
4	[a registered customer][2]	Enters	the shipping preferences			
5	[1]				the payment details are valid	AM-ADV
6	The system	Generates	an order confirmation note			

[1]*Note that because we are interested in an automated analysis, we cannot incorporate here assumptions about what causes these actions (e.g., the system or an external user).*
[2]*Replacement of a pronoun by the relevant noun is indicated with [noun].*

such as the text chunker in [37]. This tool divides a given text into syntactically correlated segments by recovering phrases constructed by certain part-of-speech tags (the first noun phrase is considered the main element in the phrase). The text chunker truncates, for example, the object "an item which is on sale" and leaves it as "an item." This results in a higher similarity to requirements that involve items whether on sale or not, but in the same time it abstracts away differences between requirements that involve items on sale and those that involve items that are not on sale.

(3) Order the vectors according to their actual occurrence in the real world and not necessarily according to their appearance in the requirement. This can be done by following a temporal ordering algorithm such as the one suggested in [33]. This algorithm is based on six types of temporal relations derived from the text: Y before X, Y immediately before X, X begins Y, X ends Y, X includes Y, and X simultaneous Y. This algorithm will swap vectors 3 and 4 in Table 5, because entering the shipping preferences precedes providing payment details (due to the word "after" that appears in the sentence).

The behavioral vectors are then classified into initial states, external events, and final states. This classification is done by analyzing the agent and the action parts of the vectors as follows (see Table 6 for summarization and examples).

Definition 3. *Given a requirement R and its derived ordered list of behavioral vectors, sorted as mentioned above (according to their actual occurrence in the real world), the* behavior associated with *R is defined by the following external events, initial states, and final states.*

- **External events**
  - Action vectors whose ***agents*** are ***external*** and their ***actions*** are ***active***.
  - Action vectors whose ***agents*** are ***internal*** and their *actions* are *passive*.
- **States**
  - Action vectors whose ***agents*** are ***internal*** and ***actions*** are ***active***.
  - Action vectors whose ***agents*** are ***external*** and ***actions*** are ***passive***.
    The decision whether the vector is classified as an initial state or a final state is done according to the order of the vector with respect to other vectors classified as external events.
- **Initial states**
  - Action vectors classified as states that appear ***before*** the first external events in the requirements.
  - Non-action vectors that appear before the first external events in the requirements.

TABLE 6
Classification of Behavioral Vectors into Initial State, External Events, and Final States

Case #	Vector	Agent	Action	Order with respect to external events	Classification	Example
1	Action	External	Active		External event	The customer can buy items.
2a	Action	External	Passive	Before first	Initial state	The customer receives a receipt (meaning the system create receipts).
2b				After last	Final state	
3a	Action	Internal	Active	Before first	Initial state	The system updates the inventory.
3b				After last	Final state	
4	Action	Internal	Passive		External event	The system receives the purchase details (meaning someone has to provide those details).
5	Non-action			Before first	Initial state	If the purchase details are valid... If the customer is new...

TABLE 7
The Resulted Behavioral Vectors of the Requirement Pair

		Agent	Action	Object	Instrument	Classification
R1	bv_1	A customer	Purchase	Items	From the catalog	External Event
	bv_2	The system	Update	The number of available item		Final state
R2	bv'_1	A customer	Buy	Products		External Event
	bv'_2	The system	Generate	A confirmation note		Final state

- **Final states**
  - Action vectors classified as states that appear ***after*** the last external events in the requirements.

Vectors whose agents are missing are classified into multiple behavioral elements (as if they were internal and external), enabling their comparison with all possible combinations. Vectors that cannot be classified following the aforementioned rules are ignored at this stage, as they represent intermediate outcomes.

Definition 3 in Appendix A, available in the online supplemental material, formally expresses those cases and their classification. Returning to the example in Table 5, vectors 2-4 can be classified as describing external events (performed by a registered customer). Vector 6 in that table is an example of a final state vector (performed by the system after the last external event). Vector 1 has multiple classification options. If it is assumed that the system displays the product list then the corresponding vector should be classified as an initial state. If not done by the system the vector should be classified as an external event. Vector 5 is not classified at all as it represents intermediate outcomes.

4.2 Behavioral Similarity Calculation

At this stage, the behavioral similarity of requirements is calculated, taking into consideration the stakeholders' preferences and needs in the form of "perspective profiles." These profiles define weights for components (initial states, external events, and final states) and roles (agents, actions, objects, and instruments). Stakeholders who are interested in variability of *functionality*, for example, will assign high weights to *actions and external events*. Stakeholders who are interested in variability of *structure*, on the other hand, will assign high weights to *objects and states*.

Definition 4. *A perspective profile is a vector of weights given to the similarities of initial states, external events, final states, agents, actions, objects, and instruments of the requirements. Notation:* $\langle w_{s1}, w_E, w_{s^*}; w_{agent}, w_{action}, w_{objec}, w_{instrument}\rangle$.

A formal version of this definition is provided in Appendix A, available in the online supplemental material. In [24] we claim that many feature diagrams follow either structural or functional perspectives. A structural perspective highlights classes (or objects), components, and attributes. It can be achieved, for example, when setting the weight of the final state to 1 and the weight of the object role to 1. In other word, this profile can be specified by the vector $\langle 0, 0, 1; 0, 0, 1, 0\rangle$. This profile will analyze the variability of the outcomes of the different functionalities. Alternatively, another structural perspective may set weights of 0.5 for the initial state, 0.5 for the final state, and 1 for the object role, i.e., be represented by the vector $\langle 0.5, 0, 0.5; 0, 0, 1, 0\rangle$. This perspective, as opposed to the former one (that assigns weights of 1 to objects and final states), considers all internal objects whose states influence or are influenced by the behavior. A functional perspective, on the other hand, focuses on processes, services, and functions. It can be achieved, for example, when setting the weight of the external events to 1 and the weights of the actions and the objects on which they are performed to 0.7 and 0.3[6] respectively (the vector $\langle 0, 1, 0; 0, 0.7, 0.3, 0\rangle$).

Following a selected perspective profile, we define now similarity of behavioral vectors, similarity of behavioral components (namely, initial states, external events, or final states), and similarity of requirements. Examples are provided immediately afterwards, whereas the formal versions of these definitions appear in Appendix A, available in the online supplemental material.

Definition 5. *Given a perspective profile, the* similarity of two behavioral vectors *is calculated as the weighted average of the similarities of their corresponding roles.*

Definition 6. *Given a perspective profile and a behavioral component (namely, initial state, external events, or final state), the similarity of behavioral components is calculated as the average of the maximum paired similarities[7] of the behavioral vectors when classified as the same behavioral component.*

Definition 7. *Given a perspective profile, the* overall behavioral similarity *of two requirements is calculated as the weighted average of similarities of their initial states, external events, and final states.*

Following those definitions, the same pair of requirements may result in different values following various perspective profiles. As an example consider the following requirements.

R_1 When a customer purchases items from the catalog, the system updates the number of available items.
R_2 When a customer buys products, the system generates a confirmation note.

The behavioral parsing of those requirements is presented in Table 7.

These requirements result in fixed similarity values of 0.79 and 0.64 for LSA and MCS respectively. The meaning of these values is vague: medium similarity. Following a functional perspective in SOVA, in which the weights of external events, actions, and objects are 1, 0.7, and 0.3 respectively (and all the other weights are 0), we obtain a similarity of 0.91. This means that the two requirements handle highly similar functions: buying products. Taking a

6. We observed in [24] that features describing functionality are commonly composed of actions (verbs) and target objects, e.g., order items.

7. The rational of choosing the average of maximum pairs is to find and match the most suitable vectors from each behavioral component. It is mainly done to address situations in which multiple vectors of a requirement are classified as the same behavioral component.

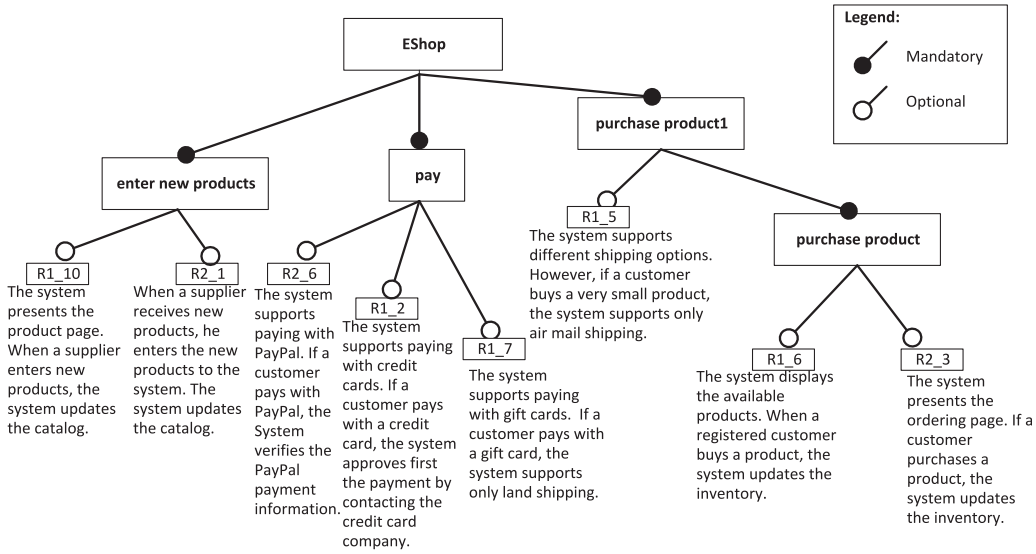

Fig. 5. An example of a feature diagram created by SOVA—using a functional perspective.

structural perspective, on the other hand, in which the weights of final states and objects are 1 (and all the other weights are 0), we obtain a similarity of 0.5. This means that the requirements differ in the behavior's outcomes. In R_1 the number of available items is updated, while in R_2 a confirmation note is generated.

4.3 Feature Diagram Construction

In the third and last stage, a feature diagram that represents the variability found in the input textual descriptions, following the selected perspective profile, is automatically constructed. The construction procedure consists of three sequential stages: requirements clustering, variability identification, and feature naming.

The *requirements clustering* stage applies a variant of the hierarchical agglomerative clustering algorithm [15], which iteratively merges clusters whose average requirement similarities are the highest, till a predefined threshold is reached. Note that, at each merge iteration, several clusters may be simultaneously merged if the differences between the similarities of their requirements are less than a predefined threshold. This kind of clustering process yields a general tree (not necessarily a binary one) with similar requirements as siblings.

In the *variability identification* stage, the tree of clusters is enhanced with variability information. Three constructs commonly used in feature diagrams notations, namely, optionality, OR and XOR relations[8], are deduced by examining the appearance of the different requirements in the input requirements documents.

- If at least two requirements that are grouped under the same parent node appear in the same input document, then the corresponding requirements are OR-grouped, to indicate that one or more requirements from a cluster appear in the same input document.
- If requirements originating from different input files are grouped under the same parent node, we consider the corresponding requirements as XOR-grouped, to indicate that exactly one requirement from a cluster can appear in the same input document.
- The parent node is optional, if it does not include requirements from all input files.

Finally, during *feature naming*, the perspective profile information is used. To compose feature names, SOVA uses up to two roles whose weights are the greatest (e.g., action + object for the functional perspective). In addition, a keyword extraction tool, such as that presented in [35], is utilized for composing the names of inner nodes of feature diagrams. That tool identifies phrases that are prominent in a given text and filters out the most significant ones based on frequency statistics (such as term frequency, inverse document frequency, and TF-IDF) and occurrence positions in the document text (e.g., beginning and end, spread of occurrences). Perceiving each requirement as a paragraph that is composed of several sentences describing the product's behavior, SOVA is able to extract the key terms (words or phrases) of a single behavior (one requirement) as well as the key terms of a set of behaviors (a cluster of requirements). These key terms are compared with the terms that constitute the behavioral elements of the followed perspective. As an example, consider the two requirements from the previous section. The key terms that are extracted for a cluster including only these requirements are items, customer, and system. When taking into consideration a functional perspective, the relevant terms that constitute external events are "when a customer purchases items from the catalog" and "when a customer buys products." The first phrase includes two key terms (customer and items), while the second phrase includes only one term (customer). Therefore the name of the cluster is composed from the action and the object of the first phrase, namely, purchase items.

Figs. 5 and 6 demonstrate two partial feature diagrams generated by SOVA for the same set of requirements (appearing in two separate documents) following different functional and structural perspectives. The leaves of these diagrams include the requirements, where the prefixes rX_Y indicate the input file number (X) and the requirement within this file (Y). The names of the inner nodes use the key terms extraction tool described above. In some cases the feature names are repeated at higher levels, because the same behavioral

8. OR and XOR relations are considered only if the number of input files is relatively high.

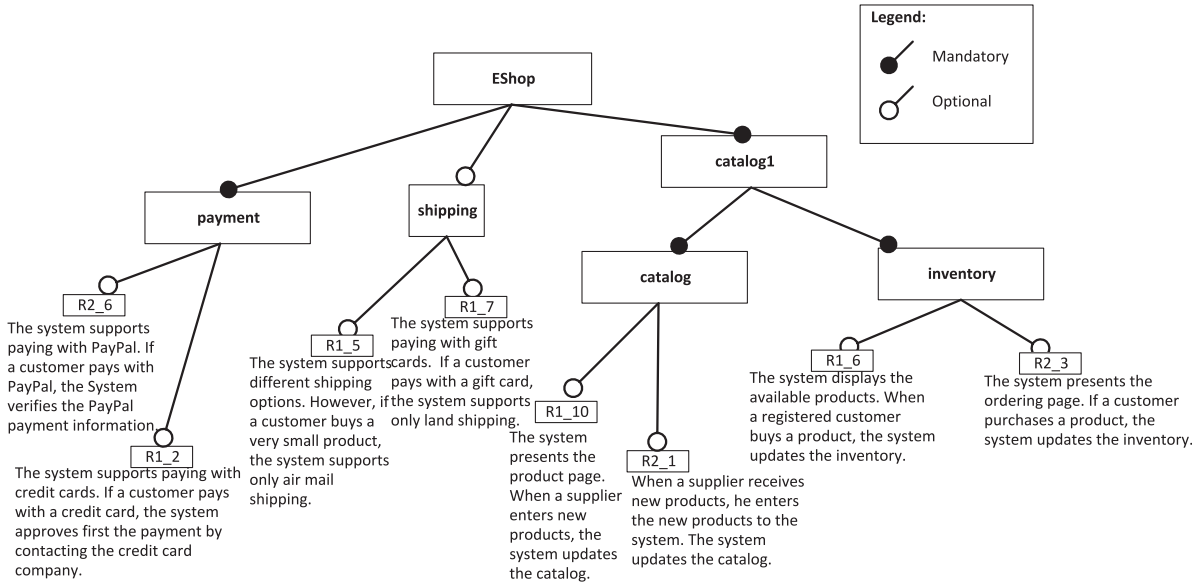

Fig. 6. An example of a feature diagram created by SOVA—using a structural perspective.

phrases are extracted as key terms (e.g., the features 'purchase product' and 'purchase product 1' in Fig. 5). This suggests a need for future research. Note that "purchase product" was extracted rather than "buy product" because the former term was ranked higher by the extraction tool based on frequency statistics and occurrence positions.

As can be seen, the two partial feature diagrams are quite different. Consider as an example requirement r1_5. Taking a structural perspective, the requirement is grouped with requirements r1_7 to represent the variability of shipping. Shipment can be done via land (in r1_7) or air (in r1_5). From a functional perspective, requirement r1_5 is more similar to requirements r1_6 and r2_3, which deal with buying products, while requirement r1_7 is more similar to requirements r1_2 and r2_6, which concern paying.

As can be observed, in both perspectives the same pairs of requirements can be grouped similarly (e.g., requirements r1_10 and r2_1). This situation implies that the requirements are similar in both their external events and final states. Both r1_10 and r2_1 refer to entering a new product from the functional perspective, and affect the catalog from the structural perspective.

5 Evaluation and Results

To evaluate the usefulness of SOVA we conducted an experiment that primarily examined the ability to answer different variability-related questions using SOVA-generated diagrams. The experimental goal, design, hypotheses, results, conclusions, and threats to validity are reported next.

5.1 Experimental Goal

The *goal* of the experiment was to analyze the usefulness of SOVA-generated feature diagrams with the *purpose* of answering variability-related questions. The *quality focus* is on ensuring whether the diagrams have a sufficiently high level of comprehensibility for requirements analysis or not. The *researchers' perspective* is on comparing the usefulness of the different variability analysis outcomes by answering different variability-related questions.

5.2 Participants

The experiment was executed at the University of Haifa in the fall semester of 2014-2015. The participants were 70 students in a 3rd and final undergraduate course dealing with software engineering. Two thirds were information systems (IS) students and one third were computer science (CS) students. All but one of the students were undergraduates. One was a CS graduate student with average results, and therefore similar to the other participants. All participants had studied modeling and requirements engineering in the software engineering course. Some participants (20 percent) had industrial experience in software engineering or modeling. The participants were not expected to be familiar with feature modeling prior to the experiment.

5.3 Experimental Material

The materials used in the experiment were based on two partial sets of requirements of e-shop applications (see Appendix B, available in the online supplemental material). The requirements were written in plain text and do not follow a predefined format or pattern. We automatically generated two feature diagrams from this set of requirements. The first one, generated by SOVA and presented in Appendix E, available in the online supplemental material, follows a functional perspective with a 70 percent emphasis on actions and 30 percent on objects of the external events. We used a functional perspective, perceiving functionality as the primary, pure perspective of SOVA, as opposed to the structural perspective that may involve objects appearing in different behavioral components. Due to the number of participants and the experimental design, we included only one SOVA diagram. While not limiting SOVA to a specific similarity measure, we used MCS [36] to calculate the similarity between phrases, and the Wu and Palmer measure [52] to calculate the similarity between words.[9] Note that we could use corpus-based,

9. This measure considers paths and distances between words in WordNet.

knowledge-based, or other semantic metrics for this purpose as well, but these two metric—MCS and Wu and Palmer—provided reasonable outcomes.

The second feature diagram, presented in Appendix D, available in the online supplemental material, was automatically constructed using ArborCraft [49], a tool that uses inputs and produces outputs similar to those in SOVA. However, SOVA and ArborCraft differ in several respects. First, ArborCraft calculates similarity following a purely semantic approach using LSA, while SOVA computes behavioral similarity taking into account not only semantic but also ontological considerations. Second, ArborCraft uses the entire requirement text to calculate similarity, while SOVA refers to the extracted behavioral vectors and the weights set in the perspective profile. As a result, ArborCraft-generated feature diagrams using the same set of inputs are quite similar, while the feature diagrams generated from SOVA enable different perspectives.

The ArborCraft-generated diagram turned out to be similar to the selected SOVA-generated feature diagram in terms of height and width. Because ArborCraft refers to a fixed number of sentences in each requirement (given as a parameter), we used requirements that are composed of exactly two sentences. Note that SOVA does not restrict the number of sentences in each requirement.

In the available version of the ArborCraft tool, the automatic naming option is not supported. Instead, the nodes in the output get dummy names like feature1, feature2, and so on. Thus, two researchers manually assigned names to nodes in the ArborCraft-generated diagram. They further improved SOVA's (automatically generated) feature names, as the method currently lacks the ability to generalize names in the upper levels of the hierarchy.[10]

Finally, to avoid visualization differences, and to incorporate the requirements text into the generated feature diagrams, we manually created versions of the automatically generated feature diagrams in Microsoft Office Visio, for both SOVA and ArborCraft. The final diagrams appear in Appendixes E and D, available in the online supplemental material, respectively.

5.4 Tasks

The participants were asked to answer 15 variability-related questions, using an online questionnaire. Because the requirements in this experiment were functional, the questions referred to different aspects of application's behaviors.

(1) Function-related questions (see Q1, Q2, and Q4 in Appendix C, available in the online supplemental material) dealt with the assessment of variability related to functions or actions supported by the e-shop applications.
(2) Options (see Q3, Q5, and Q9 in Appendix C, available in the online supplemental material) dealt with the assessment of differences in the options to perform actions in the e-shop applications.
(3) Object-related questions (see Q10, Q11, and Q15 in Appendix C, available in the online supplemental material) dealt with the assessment of variability related to objects involved in different e-shop actions.
(4) Preconditions (see Q6, Q12, and Q13 in Appendix C, available in the online supplemental material) dealt with the assessment of differences related to preconditions of different e-shop actions.
(5) Stakeholders-related questions (Q7, Q8, and Q14 in Appendix C, available in the online supplemental material) dealt with the assessment of differences between stakeholders in triggering various e-shop actions.

Note that only function-related questions directly correspond to the SOVA-generated feature diagram that followed the functional perspective. The other questions have some indirect relationship with functionality.

Some of the questions were open, requiring listing of elements from the diagrams or the requirements. Others were closed but offered either a single or a multiple answer choice. To prevent guessing, the participants were allowed to select the "I don't know" option in closed questions, and provide the code 0 in open questions.

To prevent order effect, we created 12 versions of the main questionnaire.[11] In each variant, five questions referred to the textual requirements (without any model) and ten questions referred to the requirements incorporated within a feature diagram (generated by either ArborCraft or SOVA). The requirements were used for examining difficulties that can be attributed to comprehension of automatically generated feature diagrams. Questions from all types appeared in each part, but in different order. The order of parts also differed among the 12 variants of the questionnaire.

Besides answering each question, the participants were required to grade their agreement with the following two statements.

(1) The question was difficult to answer.
(2) I feel very confident in my response.

5.5 Hypotheses, Parameters, and Variables

Following our goal of analyzing the usefulness of SOVA-generated diagrams to answer variability-related questions, we phrased two main research questions. The research questions and the corresponding sets of null hypotheses and variables are presented in Table 8.

For RQ1, regarding the ability to answer variability-related questions using SOVA-generated feature diagrams compared to textual requirements, we followed a within-subject design. For RQ2, regarding the ability to answer variability-related questions using SOVA-generated feature diagrams compared to ArborCraft-generated diagrams, we followed a between-subject design. In both cases the dependent variables are calculated as follows.

- Correctness is computed using F-measure, which is defined as the harmonic mean of precision and recall: F-measure $= 2 * \text{precision} * \text{recall}/(\text{precision} + \text{recall})$. Precision and recall are calculated with

10. As noted, in some cases the feature names are repeated at higher levels, because the same behavioral phrases are extracted as key terms.

11. A full variant of the questionnaire appears in http://mis.hevra.haifa.ac.il/~iris/research/SOVA/usefulnessQue.pdf

TABLE 8
Research Questions and the Corresponding Set of Null Hypotheses

Research question	Null hypotheses	Independent variables	Dependent variables
RQ1. Are variability-related questions answered "better" (in terms of correctness, time required to complete task, difficulty to answer, and confidence in the response) when using SOVA-generated feature diagrams than when using only (textual) requirements?	$H0_{1\text{-corr}}/H0_{1\text{-time}}/H0_{1\text{-diff}}/H0_{1\text{-conf}}$: There is no difference, in terms of correctness/time required to complete task/difficulty to answer/confidence in response, between responses answered using only (textual) requirements and those answered using SOVA-generated feature diagram.	Representation type: Requirements SOVA diagram	Correctness Time required Difficulty Confidence
RQ2. Are variability-related questions answered "better" (in terms of correctness, time required to complete task, difficulty to answer, and confidence in the response) when using SOVA-generated diagrams than when using ArborCraft-generated diagrams?	$H0_{2\text{-corr}}/H0_{2\text{-time}}/H0_{2\text{-diff}}/H0_{2\text{-conf}}$: There is no difference, in terms of correctness/time required to complete task / difficulty to answer/confidence in response, between answers given using SOVA-generated diagrams and those given using ArborCraft-generated diagram.	Model type: SOVA ArborCraft Question type: Functions Options Objects Preconditions Stakeholders	Correctness Time required Difficulty Confidence

respect to the expected answer.[12] Higher F-measure results indicate answers that are more correct.

- Time to complete task was directly recorded by the online questionnaire in seconds.
- Difficulty in answering and the confidence in response were reported by the participants on a scale of 1 to 5, where 1 indicates "strongly disagree" and 5, "strongly agree."

5.6 Experimental Design

A few days prior to the experiment we asked a graduate student, who was the teaching assistant in the course in which the experiment took place, to fill the questionnaire. The aim was two-fold: to assess the comprehensibility of the task and questions, and to confirm sufficient time (about 1.5 hours). Following the received feedback, we changed the questions from completely open to multiple choice questions.

The participants were randomly divided into 12 groups, each receiving a variant of the main questionnaire, differing in order of parts and questions. In addition to the main questionnaire we used a preliminary questionnaire to collect background information on the participants, and a post-test questionnaire for feedback about difficulties. The preliminary questionnaire had two parts. The first collected general information on the participants' subjects of study and working experience. The second tested understanding of the notation of feature diagrams. It included a simple feature diagram of mobile phones, with four variability statements (each referring to a different feature diagram construct, namely, mandatory, optional, OR, and XOR). Participants had to choose whether statements were correct or not. To prevent guessing, two additional options were given: "cannot be answered from the model," and "I don't know." The post-test questionnaire included one open-end question to obtain qualitative insights about the participants' difficulties in the task.

5.7 Analysis Procedure

We analyzed[13] the data using the Shapiro-Wilk normality test [50]. If the data was distributed normally, we used a T-test for either independent samples or paired samples, according to the relevant hypotheses. When the data was not distributed normally, we adopted the Mann-Whitney test for independent samples and the Wilcoxon signed ranks test for dependent samples [50]. Moreover, when comparing requirements to feature diagrams (RQ1) we used one-tailed statistical tests, assuming that using feature diagrams would be more comprehensible than using requirements. When comparing ArborCraft and SOVA (RQ2), we used two-tailed statistical tests due to the non-directionality of the hypotheses. In all the statistical tests, we decided to accept a probability of 5 percent of committing a Type I error [50], i.e., rejecting the null hypothesis when it is actually true. Finally, we analyzed the effect of other factors, namely, the order of parts in the questionnaire (representing the order of exposure to requirements) and the major subject of study of the participants, on the dependent variables (correctness, time, difficulty and confidence), by using multi-way ANOVA [50]. The results of the other factors' analysis are presented in Appendix F, available in the online supplemental material.

5.8 Execution (Preparation and Deviations)

The experiment took place during the lesson hours of a software engineering course. The experiment was presented as a lab exercise. Participants who took part in this exercise got one extra point to their final grade in the course. To ensure high motivation, the participants were promised higher scores based on their performance. Nevertheless, they were informed that they could quit the experiment at any time

12. In open questions, pre-coding has been done by a researcher before calculating precision and recall.

13. All the data analyses were done using SPSS 19.

TABLE 9
Raw Statistics Regarding Representation Type (RQ1, Requirements versus SOVA Diagrams)

	Requirements			SOVA Diagrams			
	Mean	Median	Sd	Mean	Median	Sd	P-value (1-tailed)
Correctness	74.4	74.93	14.36	75.32	78.95	11.83	0.344
Time	**188.24**	**187.2**	**72.22**	**126.73**	**127.6**	**43.45**	**<0.001**
Difficulty	**2.74**	**2.6**	**0.76**	**2.32**	**2.1**	**0.64**	**0.002**
Confidence	**3.6**	**3.8**	**0.79**	**3.85**	**4.0**	**0.696**	**0.045**

without negatively affecting their grades (the lecturer of the course was not a researcher in this study).

Before the experiment, we gave a half-hour introduction about SPLE and feature modeling (including notation). Then, each participant got a link to a questionnaire variant and a hard copy of slides explaining feature diagram notation. After completing the preliminary questionnaire, each participant got a hard copy of the relevant artifact, either the textual requirements or a diagram, depending on the assigned questionnaire variant. The participants were presented with the artifact on the screen and one question at a time. The participants had to answer the question before proceeding to the next question, and they could not return to previous questions (this constraint was explained to the participants before the experiment began). After completing the first part of the main questionnaire (that referred to the requirements or to one of the diagrams), the participant had to replace the hard copy of the first part artifact (requirements or a diagrams) with that of the second part artifact (a diagram or requirements), and then insert a provided code for continuing. Finally, but before closing the online questionnaire, participants had to answer the post-test question.

The differences between the experimental groups (requirements versus diagram for RQ1 and SOVA versus ArborCraft for RQ2) were analyzed, by examining the questions about the feature diagram in the preliminary questionnaire. The differences were found not significant (resulting in p-values of 0.51 and 0.76, respectively).

5.9 Results

5.9.1 Plain Requirements versus SOVA-Generated Feature Diagrams (RQ1)

In this comparison, we had only 33 participants (those who got a questionnaire variant with the SOVA-generated diagram). Table 9 summarizes the results of RQ1 regarding the differences between plain requirements and SOVA-generated diagrams. As noted, we used the Wilcoxon signed ranks test (1-tailed test) for all the hypotheses related to RQ1. With respect to correctness, the mean and median values are slightly higher when answering questions based on SOVA-generated feature diagrams than for plain requirements (75.32 and 78.95 for SOVA and 74.4 and 74.93 for requirements, respectively). However, this difference is not significant, resulting in a p-value of 0.344. The time to complete task is significantly shorter when using SOVA-generated diagrams than when using only textual requirements (126.73 versus 188.24 seconds on the mean value and 127.6 versus 187.2 seconds on the median, respectively). Additionally, it was perceived as significantly more difficult to answer questions, and the confidence in responses is significantly lower using only requirements compared to using SOVA-generated diagrams. Based on this analysis, we can reject hypotheses $H0_{1\text{-time}}$, $H0_{1\text{-diff}}$ and $H0_{1\text{-conf}}$, while hypothesis $H0_{1\text{-corr}}$ cannot be rejected.

5.9.2 SOVA versus ArborCraft (RQ2)

In this comparison, we had all 70 participants. Table 10 summarizes the results of RQ2, applying the T-test (2-tailed) to independent samples when the data distributed normally and the non-parametric Mann-Whitney test when not distributed normally.

With respect to correctness, we can see that the mean and median values are higher for SOVA than for ArborCraft (75.32 and 78.95 for SOVA, and 69.67 and 73.84 for ArborCraft). This difference is insignificant though. With respect to time to complete task, answering questions using ArborCraft-generated diagrams lasted significantly longer than answering questions using SOVA-generated diagrams (p-value of 0.049). With respect to difficulty to answer and confidence in response, a slightly lower difficulty and a slightly higher confidence were reported with SOVA-generated diagrams. However, these differences are not significant. Based on this analysis, we can only reject hypothesis $H0_{2\text{-time}}$.

To gain deeper insights, we analyzed the correctness of responses according to the question type, utilizing the non-parametric Mann-Whitney test (results are presented in Table 11).

TABLE 10
Raw Statistics Regarding Model Type (RQ2, SOVA versus ArborCraft)

	SOVA			ArborCraft				
	Mean	Median	sd	Mean	Median	sd	Stat. Test*	P-Value (2-Tailed)
Correctness	75.32	78.95	11.83	69.67	73.84	16.1	M	0.12
Time	**126.73**	**127.6**	**47.73**	**148.6**	**142.7**	**43.45**	**T**	**0.049**
Difficulty	2.32	2.1	0.64	2.54	2.4	0.72	T	0.173
Confidence	3.85	4.0	0.7	3.82	3.8	0.75	M	0.646

* *T- Independent samples T-test, M - Mann-Whitney Test.*

TABLE 11
Raw Statistics Regarding Question Type (SOVA versus ArborCraft)

	SOVA correctness			ArborCraft correctness			
	Mean	Med.	sd	Mean	Med.	Sd	P-Value 2-Tailed)
Functions	**81.40**	**83.65**	**12.59**	**68.77**	**75.00**	**18.97**	**<0.001**
Options	77.73	83.33	26.34	77.70	83.33	27.13	0.961
Objects	**53.83**	**50.00**	**26.11**	**68.50**	**75.00**	**23.83**	**0.024**
Preconditions	**89.44**	**90.00**	**13.23**	**70.05**	**75.00**	**27.06**	**<0.001**
Stakeholders	**74.19**	**83.33**	**21.05**	**63.33**	**69.70**	**21.63**	**0.023**

For questions related to functions, preconditions, and stakeholders, SOVA significantly outperformed ArborCraft (p-values of <0.001, <0.001, and 0.023, respectively). For object-related questions, ArborCraft significantly outperformed SOVA (p-value of 0.024), as expected. For option-related questions, ArborCraft and SOVA showed similar results.

5.9.3 Analysis of Post-Test Questionnaire Feedback

Qualitative analysis of the responses in the post-questionnaire showed that 48 participants explicitly referred to preferences and/or difficulties in the task. Many of them (36) stated that it was hard to answer the questions using the requirements alone. Among the reasons they mentioned, we found statements like "it takes a lot of time to find the answer in the requirements," "required a lot of reading," "requirements are not organized," and "hard to find the requested information in a long text." They further mentioned that it was easier and quicker to use the diagrams because "they were organized," and in the diagrams it was "easy to locate the main functions of the application." Among those who reported in the feedback question that they preferred feature diagrams, we found that more than two thirds used SOVA and less than a third used ArborCraft.

Only four participants claimed that it was easier to use the requirements without the diagram. All of them used a questionnaire variant that included the ArborCraft-generated diagram. They justified their claims with statements like "reading the whole paragraph is much easier," "too much text was compressed into the diagram and it was hard to read," and it was "hard to find the relevant information in the diagram."

Eight participants had no preference for the representation type, claiming that it is difficult to answer variability-related questions in any case. They particularly mentioned that it is difficult to compare two applications and to answer function-related questions when having only the requirements. On the other hand, answering focused questions that referred to specific elements was easier to answer if based only on the requirements. Not surprisingly, the feature diagrams helped when the questions referred to requirements jointly grouped in the feature diagrams under nodes with names meaningful to the task at hand.

5.10 Discussion

As reported above, there are significant differences between performing variability-related tasks using SOVA diagrams and textual requirements. When using SOVA, tasks are completed faster and more easily, and confidence is higher. Due to the explicit specification of variability in the provided diagrams, as well as the organization and grouping of features into hierarchical, tree-like diagrams, it is easier and faster to identify the relevant requirements and analyze variability. We believe that, due to the relatively low number of requirements—20 overall, we found no significant results with respect to correctness. The participants could read the whole set of the requirements for each variability-related task, identifying the relevant requirements and analyzing variability. This caused them to take more time in completing the task. However, when the number of requirements increases, this strategy may become difficult, if not impossible, to follow. Only further research with larger sets of requirements could confirm or deny this hypothesis. It is also important to note that the participants were not familiar with feature diagrams prior to the experiment. Prior training with the notation and its semantics may also improve the performance of variability-related tasks.

We also found significant differences between the usefulness of SOVA and ArborCraft that mainly uses a latent semantic analysis measurement to identify similar requirements. The average and median correctness scores were more than five points higher when using SOVA than when using ArborCraft, while the time required was significantly shorter. The reasons for these results may be the use of the specific SOVA-generated diagram (the one using a functional perspective) and the interest in variability of functional requirements. The semantic similarity of those requirements was not high enough to group them together, although they represent similar behaviors. Hence, introducing ontological considerations that directly refer to initial states (preconditions), to external events (stakeholders, functions and options to perform actions), and to final states (objects), improved the ability of participants to represent correctly the behavioral variability as perceived from external points of view.

As an example, consider the question "when can orders be searched?" This was classified as relating to preconditions. It was more correctly answered using SOVA, because SOVA grouped the related requirements R1_1 and R2_1, which referred to the same functionality—"search items"—under the same node, while ArborCraft had no feature referring to search items. ArborCraft, on the other hand, grouped requirements R1_1 with R1_4, R1_10 and R1_2 due to their semantic similarities, while R2_1 appeared directly under the root (i.e., no similar requirements exist). Additionally ArborCraft had no feature referring to search items. Similarly, the question "who can add product review into the application?", which was classified as relating to stakeholders, required finding first the actions referring to

product review and then identifying the relevant stakeholders. The relevant requirements, R2_4 and R2_9, were grouped together in SOVA under "review products," while in ArborCraft R2_4 is secondarily associated with R2_8 and R2_10, which refer to product additions. Requirement R2_9 is not associated with any other requirements (appearing directly under the root).

On the other hand, using a functional perspective, objects become secondary residents and we indeed saw significantly fewer correct answers with respect to object-related variability tasks. The object-related questions referred to "confirmation note," "payment cards," and "shopping carts." ArborCraft properly grouped the related requirements in two out of the three cases (confirmation note and payment cards), due to the semantic similarity of the corresponding requirements. The SOVA-generated diagram properly grouped only the payment-related requirements.

In [24], we showed that different perspective profiles of SOVA are useful for performing different variability-related tasks. In cases where we are interested in analyzing variability that refers to objects, it may be more reasonable to generate a feature diagram following a structural perspective. We could not do this in the experiment due to the large number of variants and the relatively low number of participants. Further research is therefore needed on this issue.

With respect to option-related tasks, which referred to the options to perform actions in the e-shop application, we found no significant differences between SOVA and ArborCraft, although these option tasks relate to actions too. The relevant questions referred to the options to pay, confirm orders and ship products. Further investigation of the related requirements and their different grouping in both diagrams showed that in SOVA the three requirements that referred to payment, R1_6, R1_7, and R2_6, were grouped together, while in ArborCraft only requirements R1_7 and R2_6, which referred to paying by cards, were grouped together. ArborCraft, on the other hand, more accurately grouped the requirements related to order confirmation, requirements R1_3 and R2_5. In SOVA, requirement R2_5 was grouped with requirements referring to product removals and requirements referring to shipment and cancelation. Requirement R1_3, on the other hand, was grouped with requirement R1_4. These two requirements refer to order confirmation (or order finalization) and item shipment, and thus SOVA found them more similar. With respect to product shipment, ArborCraft grouped two out of the three related requirements (R1_3 and R2_2), and SOVA grouped two different requirements (R1_3 and R1_4).

To summarize, it seems that SOVA-generated diagrams, and specifically those following a functional perspective, are useful to understand variability related to functional requirements. The flexibility of SOVA to generate diagrams tailored to specific needs and tasks, as well as its reference to ontological considerations, seem to be beneficial both in comparison with textual requirements and with a semantic tool—ArborCraft.

5.11 Threats to Validity

The validity of our study is subject to several threats. We report these threats and the actions taken to minimize them, following the suggestion in [50].

Construct validity threats concern the relationships between theory and observation. They are mainly due to the method used to assess the outcomes of tasks. We used an online questionnaire to assess the usefulness of SOVA. The correctness of answers was evaluated using an information retrieval-based approach in order to avoid subjective evaluation as much as possible. In addition, the diagrams and questions used as experimental material were checked by three researchers. Time was automatically recorded by the online questionnaire. Difficulty and confidence were collected using the common Likert scale.

Internal validity threats concern external factors that may affect a dependent variable. They may be due to learning and fatigue effects experienced by the participants, or to their background. Those effects are mitigated by the multiway ANOVA reported in Appendix F, available in the online supplemental material. This analysis revealed that the order of exposure affected only the time to complete the tasks (learning) and not the correctness (fatigue effects). Additionally, the participants' knowledge and skills in using feature diagrams might also impact the results. Hence, we included questions in the preliminary questionnaire that examined whether the participants comprehended the notation of feature diagrams or not. We found that most participants (85 percent) received high scores on those questions, and that the participants were uniformly distributed among the experimental groups, verified by the Mann-Whitney test.

External validity threats concern the ability to generalize the results. The main threats in this area stem from the single domain we used, the simple tasks, and from the type of participants. Due to the complexity of the design and the large number of groups used in the experiment, we could not use additional application domains. In the future, it will be interesting to replicate our experiment with different domains and different perspective profiles for the SOVA-generated diagrams. Regarding the tasks, we claim that these were sufficiently realistic and carefully chosen such that they can be answered in the different settings (requirements, SOVA, and ArborCraft). As noted, the teaching assistant in the course in which the experiment took place verified the comprehensibility of the questions. Based on her feedback, hints on the expected answers were introduced (e.g., by changing open questions to multiple choice questions). Although the participants were undergraduate students with little experience in requirements engineering, they had the required knowledge and training. The use of students in experiments like ours, that are not designed for experts, is nevertheless acceptable [29]. Specifically in the context of requirements engineering, it was shown in [45] that students have a good understanding of the way industry behaves, and may work well as participants in such empirical studies as ours. Further studies may confirm whether or not our results can be generalized to more experienced participants.

Finally, *conclusion validity concerns* refer to the relationship between the treatment and outcomes. The statistical analysis was performed using T-tests (independent and paired-sample tests), and non-parametric test (the Mann-Whitney and Wilcoxon signed ranks tests for independent and dependent samples, respectively). These tests are also well suited

for use on small samples, such as the 70 participants in our experiment (resulting in a little more than 30 participants in each experimental group). Moreover, multi-way ANOVA was used mainly to detect possible interactions between co-factors (e.g., order of exposure to diagrams and participants' major subjects of study) and the independent variables (model type). Even if all the assumptions/conditions for using ANOVA were not valid (i.e., the samples were not normally distributed), this test is quite robust [26].

6 Summary and Future Research

SPLE practices have been successful in reducing development costs and enhancing time-to-market, as well as improving product quality. However, adoption of SPLE is a very demanding process and is usually done when artifacts of different products already exist. In those extractive scenarios, analyzing and representing the variability among the different products is a prerequisite to successful SPLE adoption.

In this work we focus on the existence of functional requirements of software for individual products and using them for analyzing and representing variability, as the first step for deciding whether and how to adopt SPLE.

Existing studies suggest analyzing and extracting variability from textual artifacts using syntactic and semantic considerations. They further present variability in models that follow fixed, predefined perspectives inherited from the characteristics of inputs or methods. As we showed, using syntactic and semantic considerations may limit the ability to reuse decisions in SPLE adoption. It can also limit adjusting variability related to functionality as perceived externally by different stakeholders. Furthermore, variability models need to be appropriately adapted for the stakeholders' preferences and the given tasks. Thus, single variability perspective approaches may provide limited usefulness.

In this research we proposed a method, named SOVA to enhance existing variability approaches. The contribution of the method is twofold. (1) It uses an ontological model for extracting behavioral components from textual requirements and analyzing the variability of software products based on these components. (2) It introduces flexibility in the construction of feature diagrams, enabling the stakeholders to control the perspective appropriate to their needs and preferences.

The results of evaluating the usefulness of SOVA-generated diagrams demonstrate the method's benefits in representing variability. Variability-related tasks performed with SOVA were better and significantly faster, compared to those with plain text requirements and compared to those with diagrams generated following a semantic approach (ArborCraft). Following the functional perspective of SOVA, the answers to questions related to variability in functions, stakeholders and preconditions were significantly favorable. This evaluation demonstrated SOVA's potential to lay the foundation for flexible behavioral variability analysis.

Future research can include several directions. First, the method needs to be evaluated in different scenarios and domains, with different tasks and stakeholders. Second, the method currently associates each requirement with a single feature. Extending the method to extract several features from the same requirement, if needed, may improve analysis of variability. If, for example, a certain requirement involves several external events (e.g., confirming an order and shipping products), we may want to separately analyze the variability of each part with respect to other requirements. In other words, we expect to find such a requirement under the features "order confirmation" and "product shipment." A third direction may be extending the similarity calculation by considering domain-specific vocabulary and web sources to better represent acronyms and domain-specific terms. A fourth direction relates to the ability to align with "good" perspective profiles to given tasks and stakeholders. This can be done according to general guidelines and analysis of given requirements. Additionally, it would be also interesting to explore merging several perspectives into a single variability model, and to support more sophisticated perspectives. Examples of such extensions could include perspectives that structure the features following predefined patterns, such as clustering first according to objects and then according to actions. Alternatively, clustering first according to stakeholders (agents) and then according to actions. Additional improvements can be also made with regard to feature naming, and generalization of names in the higher levels of hierarchy using summarization tools. Finally, the method needs to be extended to support more complex requirements statements, which represent "swarms" of behaviors (including branches and loops) and not just single ones. This will enable us to analyze relationships between requirements (like hierarchies) and not just individual requirements.

References

[1] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, and P. Lahire, "On extracting feature models from product descriptions," in *Proc. 6th VaMoS Workshop*, 2012, pp. 45–54.

[2] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, and A. Rummler, "An exploratory study of information retrieval techniques in domain analysis," in *Proc. 12th Int. Softw. Product Line Conf.*, 2008, pp. 67–76.

[3] N. H. Bakar, Z. M. Kasirun, and N. Salleh, "Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review," *J. Syst. Softw.*, vol. 106, pp. 132–149, 2015.

[4] G. Bécan, M. Acher, B. Baudry, and S. Ben Nasr, "Breathing ontological knowledge into feature model synthesis: An empirical study," *Empirical Softw. Eng.*, pp. 1–48, 2015, Doi: 10.1007/s10664-014-9357-1.

[5] T. Berger, R. Rublack, D. Nair, J. M. Atlee, M. Becker, K. Czarnecki, and A. Wąsowski, "A survey of variability modeling in industrial practice," in *Proc. 7th Int. Workshop Variability Model. Softw.-Intensive Syst.*, 2013, pp. 7:1–7:8.

[6] M. Bunge, "Treatise on basic philosophy," in *Ontology I: The Furniture of the World*, vol. 3. Boston, MA, USA: Reidel, 1977.

[7] M. Bunge, "Treatise on basic philosophy," in *Ontology II: A World of Systems*, vol. 4. Boston, MA, USA: Reidel, 1979.

[8] C. Burgess, K. Livesay, and K. Lund, "Explorations in context space: Words, sentences, discourse," *Discourse Processes*, vol. 25, nos. 2/3, pp. 211–257, 1998.

[9] L. Chen and M. A. Babar, "A systematic review of evaluation of variability management approaches in software product lines," *Inf. Softw. Technol.*, vol. 53, pp. 344–362, 2011.

[10] K. Chen, W. Zhang, H. Zhao, and H. Mei, "An approach to constructing feature models based on requirements clustering," in *Proc. 13th IEEE Int. Conf. Requirements Eng.*, 2005, pp. 31–40.

[11] D. Clarke and J. Proenca, "Towards a theory of views for feature models," in *Proc. 1st Int. Workshop Formal Methods Softw. Product Line Eng.*, 2010, pp. 91–98.

[12] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Reading, MA, USA: Addison-Wesley, 2001.
[13] K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged configuration through specialization and multi-level configuration of feature models," *Softw. Process Improvement Practice*, vol. 10, no. 2, pp. 143–169, 2005.
[14] J. M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans, "Feature model extraction from large collections of informal product descriptions," in *Proc. 9th Joint Meeting Found. Softw. Eng.*, 2013, pp. 290–300.
[15] W. H. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *J. Classification*, vol. 1, no. 1, pp. 7–24, 1984.
[16] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli, "On-demand feature recommendations derived from mining public product descriptions," in *Proc. 33rd IEEE Int. Conf. Softw. Eng.*, 2011, pp. 181–190.
[17] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta, "Mining commonalities and variabilities from natural language documents," in *Proc. 17th Int. Softw. Product Line Conf.*, 2013, pp. 116–120.
[18] D. Gildea and D. Jurafsky, "Automatic labeling of semantic roles," *Comput. Linguistics*, vol. 28, no. 3, pp. 245–288, 2002.
[19] W. H. Gomaa and A. A. Fahmy, "A survey of text similarity approaches," *Int. J. Comput. Appl.*, vol. 68, no. 13, pp. 13–18, 2013.
[20] G. Halmans and K. Pohl, "Communicating the variability of a software-product family to customers," *Softw. Syst. Model.*, vol. 2, no. 1, pp. 15–36, 2003.
[21] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher, "Supporting domain analysis through mining and recommending features from online product listings," *IEEE Trans. Softw. Eng.*, vol. 39, no. 12, pp. 1736–1752, Dec. 2013.
[22] R. Herrejon-Lopez, L. Linsbauer, J. Galindo, J. Parejo, D. Benavides, S. Segura, and A. Egyed, "An assessment of search-based techniques for reverse engineering feature models," *J. Syst. Softw.*, vol. 103, pp. 353–369, 2015.
[23] A. Hubaux, M. Acher, T. T. Tun, P. Heymans, P. Collet, and P. Lahire, "Separating concerns in feature models: Retrospective and support for multi-views," in *Domain Engineering: Product Lines, Languages, and Conceptual Models*, I. Reinghartz-Berger, et al. Eds. New York, NY, USA: Springer, 2013, pp. 3–28.
[24] N. Itzik and I. Reinhartz-Berger, "Generating feature models from requirements: Structural vs. functional perspectives," in *Proc. 18th Int. Softw. Product Line Conf.: Companion Volume Workshops, Demonstrations, Tools*, 2014, pp. 44–51.
[25] N. Itzik and I. Reinhartz-Berger, "SOVA—A tool for semantic and ontological variability analysis," in *Proc. CAiSE 2014 Forum*, 2014, pp. 177–184.
[26] G. R. Iversen and H. Norpoth, *Analysis of Variance*, 2nd ed. Newbury Park, CA, USA: Sage, 1987.
[27] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Carnegie-Mellon Univ., Pittsburgh, PA, USA, SEI Tech. Rep. CMU/SEI-90-TR-21, 1990.
[28] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, "FORM: A feature-oriented reuse method with domain-specific reference architectures," *Ann. Softw. Eng.*, vol. 5, pp. 143–168, 1998.
[29] B. A. Kitchenham, S. Lawrence, P. Lesley, M. Pickard, P. W. Jones, D. C. Hoaglin, and K. E. Emam, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, Aug. 2002.
[30] C. Krueger, "Easing the transition to software mass customization," in *Software Product-Family Engineering*. Berlin, Germany: Springer, 2002, pp. 282–293.
[31] C. W. Krueger. (2006). Software Mass Customization. BigLever Software, Inc. [Online]. Available: http://www.biglever.com/extras/BigLeverMassCustomization.pdf
[32] T. K. Landauer, P. W. Foltz, and D. Laham, "Introduction to latent semantic analysis," *Discourse Processes*, vol. 25, pp. 259–284, 1998.
[33] I. Mani, M. Verhagen, B. Wellner, C. M. Lee, and J. Pustejovsky, "Machine learning of temporal relations," in *Proc. 21st Int. Conf. Comput. Linguistics/44th Annu. Meeting Assoc. Comput. Linguistics*, 2006, pp. 753–760.
[34] J. D. McGregor, D. Muthig, K. Yoshimura, and P. Jensen, "Guest editors' introduction: Successful software product line practices," *IEEE Softw.*, vol. 27, no. 3, pp. 16–21, May/Jun. 2010.
[35] O. Medelyan and I. H. Witten, "Domain-independent automatic keyphrase indexing with small training sets," *J. Am. Soc. Inf. Sci. Technol.*, vol. 59, no. 7, pp. 1026–1040, 2008.
[36] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *Proc. 21st Natl. Conf. Artif. Intell.*, 2006, vol. 1, pp. 775–780.
[37] T. Morton, J. Kottmann, J. Baldridge, and G. Bierner. (2005). OpenNLP: A java-based NLP toolkit. [Online]. Available: http://opennlp.sourceforge.net
[38] N. Niu and S. Easterbrook, "Extracting and modeling product line functional requirements," in *Proc. 16th IEEE Int. Requirements Eng. Conf.*, 2008, pp. 155–164.
[39] K. Pohl, G. Böckle, and F. van der Linden, *Software Product-Line Engineering: Foundations, Principles, and Techniques*. New York, NY, USA: Springer, 2005.
[40] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning, "A multi-pass sieve for coreference resolution," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2010, pp. 492–501.
[41] I. Reinhartz-Berger, N. Itzik, and Y. Wand, "Analyzing variability of software product lines using semantic and ontological considerations,"*Proc. 26th Int. Conf. Adv. Inf. Syst. Eng.*, 2014, pp. 150–164.
[42] I. Reinhartz-Berger, A. Sturm, and Y. Wand, "Comparing functionality of software systems: An ontological approach," *Data Knowl. Eng.*, vol. 87, pp. 320–338, 2013.
[43] I. Reinhartz-Berger, A. Sturm, and Y. Wand, "External variability of software: Classification and ontological foundations," in *Proc. 30th Int. Conf. Conceptual Model.*, 2011, pp. 275–289.
[44] S. She, R. Lotufo, T. Berger, A. Wasowski, and K. Czarnecki, "Reverse engineering feature models," in *Proc. 33rd Int. Conf. Softw. Eng.*, 2011, pp. 461–470.
[45] M. Svahnberg, A. Aurum, and C. Wohlin, "Using students as subjects—An empirical evaluation," in *Proc. 2nd ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Kaiserslautern, Germany, 2008, pp. 288–290.
[46] P. Turney, "Mining the web for synonyms: PMI-IR versus LSA on TOEFL," in *Proc. 12th Eur. Conf. Mach. Learning*, 2001, pp. 491–502.
[47] Y. Wand and R. Weber, "On the deep structure of information systems," *J. Inf. Syst.*, vol. 5, no. 3, pp. 203–223, 1995.
[48] Y. Wand and R. Weber, "An ontological model of an information system," *IEEE Trans. Softw. Eng.*, vol. 16, no. 11, pp. 1282–1292, Nov. 1990.
[49] N. Weston, R. Chitchyan, and A. Rashid, "A framework for constructing semantically composable feature models from natural language requirements," in *Proc. 13th Int. Softw. Product Line Conf.*, 2009, pp. 211–220.
[50] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering—An Introduction*. Norwell, MA, USA: Kluwer, 2000.
[51] WordNet. [Online]. Available: http://wordnet.princeton.edu/
[52] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proc. 32nd Annu. Meeting Assoc. Comput. Linguistics*, 1994, pp. 133–138.
[53] H. Zhao, W. Zhang, and H. Mei, "Multi-view based customization of feature models," *J. Frontiers Comput. Sci. Technol.*, vol. 2, no. 3, pp. 260–273, 2008.

Nili Itzik received the BSc degree in applied mathematics from the Technion—Israel Institute of Technology, and the MSc degree in information systems from the University of Haifa, Israel. She has gained extensive experience in developing and managing software projects in industry. Additionally, she has specialized in huge billing databases and data conversion between systems. Her current research interests include automatic variability analysis from textual requirements.

Iris Reinhartz-Berger received the BSc degree in computer science and applied mathematics and the MSc and PhD degrees in information management engineering, all from the Technion—Israel Institute of Technology. She is a faculty member at the Department of Information Systems, University of Haifa, Israel. Her research interests include conceptual modeling, domain analysis, modeling languages and techniques for analysis and design, and systems development processes. She co-organized a series of domain engineering workshops, in conjunction with the CAiSE conference, and co-edited a book entitled *Domain Engineering: Product Lines, Languages, and Conceptual Models*.

Yair Wand received the BSc degree in physics from the Hebrew University, Jerusalem, the MSc degree in physics from the Weizmann Institute, Israel, and the DSc degree in operations research from the Technion—Israel Institute of Technology. He is a CANFOR professor of MIS at the Sauder School of Business, The University of British Columbia, Canada. His research interests include theoretical foundations and methods for information systems analysis and design. In particular, he has worked on ontological approaches to information systems, on theoretical and empirical methods in conceptual modeling, on the application of classification principles in modeling and design of information systems, and on business process modeling. He has published about 100 referred papers. He is currently on the editorial boards of six journals.