Improving the management of product lines by performing domain knowledge extraction and cross product line analysis

Iris Reinhartz-Berger *, Ora Wulf-Hadash
Department of Information Systems, University of Haifa, Haifa 31905, Israel

A R T I C L E   I N F O
Article history:
Received 2 March 2014
Received in revised form 20 November 2014
Accepted 25 November 2014
Available online 5 December 2014

Keywords:
Software product line engineering
Variability management
Domain analysis
Feature modeling

A B S T R A C T
Context: Increase in market competition is one of the main reasons for developing and maintaining families of systems, termed Product Lines (PLs). Managing those PLs is challenging, let alone the management of several related PLs. Currently, those PLs are managed separately or their relations are analyzed assuming explicit specification of dependencies or use of an underlying terminology. Such assumptions may not hold when developing the PLs in different departments or companies applying various engineering processes.

Objective: In this work we call for utilizing the knowledge gained from developing and maintaining different PLs in the same domain in order to recommend on improvements to the management of PLs.

Method: The suggested approach conducts domain knowledge extraction and cross PL analysis on feature diagrams – the main aid for modeling PL variability. The domain knowledge is extracted by applying similarity metrics, clustering, and mining techniques. Based on the created domain models, the approach performs cross PL analysis that examines relations in the domain models and generates improvement recommendations to existing PLs and overall management recommendations (e.g., merging or splitting PLs).

Results: The approach outcomes were evaluated by humans in a domain of mobile phones. The evaluation results may provide evidence that the outcomes of the approach in general and its recommendations in particular meet human perception of the given domain.

Conclusion: We conclude that through domain knowledge extraction and cross PL analysis the suggested approach may generate recommendations useful to the management of individual PLs, as well as to the overall management of different PLs in the same domain.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Different systems and software products may be developed in the same domain. When developed in the same company, it may be effective and efficient to develop and maintain systems that share a common, managed set of features as families rather than as individual systems [8,23]. Those families are termed Product Lines (PLs).

The technical and organizational benefits of PLs development [8,19] and especially the increased market agility may yield the existence of several PLs under the same roof (i.e., the developing company). These PLs may be developed in the same department or in different departments of the same company, e.g., to specify an (ultra-)large-scale system whose components are based on heterogeneous technological platforms [16]. Alternatively, such PLs may originally be developed in different companies but end up under the same roof due to mergers or acquisitions of companies or participation in a consortium of companies that develop PLs in the same domain.

The management of a PL is challenging, let alone the management of several PLs. Different studies have examined the relationships between PLs or various aspects of the same PL usually expressed as feature diagrams [17] – the main aid for modeling PL variability. These studies aim at supporting scalability [27], increasing modularity and reuse [2,6], synchronizing versions [18,32], or supporting multi PL management [16]. They mainly suggest comparing, matching, merging, and composing feature diagrams [1]. Furthermore, they assume using the same terminology...
in the input feature diagrams or having explicit specification of the relations between similar terms. While these assumptions are perfectly suitable for handling feature diagrams that represent different aspects of the same PL or different PLs that use the same underlying terminology, it is not enough for performing commonality and variability analysis of different PLs that have potentially been developed in different locations (departments or companies) and employed different development processes.

As an example to such a situation, consider an imaginary scenario in which the two leading mobile device manufacturers, Samsung and Apple, decide to merge. Each company has its own PLs, e.g., Galaxy S of Samsung and iPhone of Apple. The merged company, Samsung–Apple, wishes to examine whether to merge these PLs, reorganize them differently to increase reuse, or close one PL leaving the other one with small modifications. However, this aim may be difficult to be achieved as similar features may be differently named in the two original companies and the features may be differently structured. For example, the feature “Audio and Video” of Samsung may be represented by three separate features in Apple’s PLs: “Audio Calling”, “Video Calling”, and “Video Recording”. The feature “Memory” of Samsung may be similar to the feature “Capacity” of Apple. The feature “Physical Specification” of Samsung may be similar to the feature “Weight and Dimension” of Apple. Moreover, similar high level features may turn to be different in the lower level of abstraction. “Display”, for example, may be characterized by “Size” and “Resolution” in both companies. However, Samsung may further specify the display’s “Technology”, “Color Depth”, and “Pen Support”, while Apple may just additionally specify the display’s “PPI” (pixel-per-inch).

In this work we suggest using various PLs in the same domain in order to extract domain knowledge and recommend on improvements to the management of individual PLs and the whole set of PLs. To this end, we suggest an approach that gets as an input a set of feature diagrams, each representing the variability within a different PL in the same domain. The approach uses similarity metrics to compare features and their variations, adopts clustering techniques for grouping similar features, and utilizes feature mining techniques for generating domain models. Based on the generated domain models, the approach generates recommendations to the management of individual PLs and the whole set of PLs. Examples of such recommendations may be introducing relevant features that are not supported in certain PLs, merging too similar PLs, or splitting PLs in order to potentially increase reuse. As the approach involves different PLs for extracting domain knowledge and generating improvement recommendations, it is considered “cross PL analysis”.

The current paper extends previously published papers on this approach which focused on clustering similar features [35] and constructing the domain knowledge [34]. Here we concentrate on deriving improvement recommendations from the knowledge extracted on the domain of discourse. The evaluation of the approach is also elaborated in the current paper to examine more complicated cases and to validate the correctness and usefulness of the generated outcomes to humans (domain experts and users).

The rest of this paper is organized as follows. Section 2 provides the needed background on feature diagrams and reviews the relevant literature. Section 3 describes and exemplifies the domain knowledge extraction part, while Section 4 concentrates on the recommendation generation through cross PL analysis. Section 5 presents the evaluation of the approach and its outcomes, as well as discusses threats to validity. Finally, Section 6 concludes and refers to future research.

2. Background and literature review

2.1. Feature diagrams and their analysis

A feature diagram captures the characteristics of a given PL, as well as the relationships and constraints among these characteristics (features) [17]. Feature diagrams are considered relatively simple, as they are technology-independent and include characteristics that are visible to stakeholders, and especially to end-users. A feature diagram is usually represented as a tree (or a directed-acyclic graph), where the nodes denote features and the edges – relationships and constraints (dependencies). Different modeling notations and formalizations have been suggested over the years to model feature diagrams. Although no standard feature modeling language has been emerged yet, four types of relationships and two types of constraints are commonly defined in most feature diagram notations: (1) mandatory features – features that must be selected for any product in the line; (2) optional features – features that may be selected to introduce some added value to the product; (3) XOR-grouped features – exactly one feature among the alternatives has to be selected for a particular product; (4) OR-grouped features – any combination of the composing features that includes at least one feature is valid; (5) requires’ constraints – inclusion of a feature implies the inclusion of another feature; and (6) ‘excludes’ constraints – inclusion of a feature forbids the inclusion of another feature.

Fig. 1 depicts two feature diagrams in the domain of mobile phones. In the upper feature diagram (a), the PL supports utility functions (namely, voice calls and messaging services), screens that may be basic, color, high resolution, or a combination of them, and optional extras that may include a camera, mp3, mp4, or a combination of them. The second PL, presented in Fig. 1(b), supports calls, sending utilities, displays, and optional media capabilities in the form of a camera, mp3, or both. Note that both PLs are structured as trees, where the child features are connected to the parent features with mandatory, optional, OR, or XOR (alternative) relations. Cross-relations also exist in the form of “require” and “exclude” constraints. In Fig. 1(a), for example, camera requires high resolution screen and in Fig. 1(b) camera excludes low resolution display.

Many studies suggest automated analysis of feature diagrams. Benavides et al. [4] found about 30 analysis operations that check different quality attributes. Examples of such operations are: (1) Anomalies detection which returns information about dead features, false optional features, wrong cardinalities, and redundancies; (2) Homogeneity which returns how similar the products in the line are; and (3) Variability factor which returns the ratio between the actual number of products and the overall potential number of products. These operations work on individual feature diagrams and do not analyze relations between feature diagrams in order to improve quality.

2.2. Analyzing collections of feature diagrams

Analysis of collections of feature diagrams is also studied in the literature for different purposes. First, the merge of feature diagrams is studied for supporting scalability of PLs. It is recommended to divide feature diagrams that represent large PLs with high degrees of variability [25]. In these cases, the separate feature diagrams need to be merged in order to create models of the whole PLs. Segura et al. [27] suggest using graph transformations for automating the merge of feature diagrams. They present a catalogue of visual, technology-independent rules that describe how to build a feature diagram including all the products represented by two given feature diagrams. The main assumption in this
A third group of organizational approaches is discussed in [16]. This group refers to the management of PL teams or interdependencies of business units, e.g., [5]. This group is out of the scope of our study.
In order to generate relevant improvement recommendations, the domain knowledge needs to be extracted first. This is done in a three step process (see Fig. 2). First, during the Feature Similarity Analysis step, the set of input feature diagrams is analyzed using linguistic techniques for finding similar features that can be clustered together in the next step. In the second step, Feature Clustering, a variation of an agglomerative clustering technique is used for creating groups (clusters) of similar features that may represent variants of the same features. Finally, in the third step, named Domain Model Creation, the relationships between the clusters are analyzed, generating the domain model from which improvement recommendations can be derived. This section briefly reviews these steps, while elaborations can be found in [34,35].

3.1. Feature similarity analysis

Following studies from the fields of ontology matching [7,28] and data integration [31,33], we use semantic and structural similarity to measure features (and PLs) similarity. Semantic similarity measures the difference between the feature names. Semantic metrics are commonly divided into knowledge-based or corpus-based [13,21]. Corpus-based measures identify the degree of similarity based on information derived from large corpora, while knowledge-based measures use information drawn from semantic networks, such as WordNet. Our approach can be used with any semantic similarity measure whose values range between 0 (semantically different) and 1 (semantically identical), or can be normalized to this range. Returning to our Samsung–Apple example and applying semantic measures, we can find some degree of similarity between the features “audio and video”, “audio calling”, “video calling”, and “video recording”, but it would be difficult to find similarity between “memory” and “capacity”, or between “physical specification” and “weight and dimension”. In those cases, we may benefit from examining the structural similarity of features.

Structural similarity takes into account the immediate context of a feature, namely its child features and parent feature. The similarity of child and parent features is taken into account only if it is greater than the semantic similarity of the corresponding features.
and thus can increase their similarity\(^3\). The child features of “physical specification” in Samsung’s comparison utility are “dimension” and “weight”, and the child features of “weight and dimension” in Apple’s comparison utility are “height”, “width”, “depth”, and “weight”. The high similarity between the child features will increase the similarity of their parent features, resulting in overall similarity higher than the semantic similarity for “physical specification” and “weight and dimension”.

The following definition is used for calculating feature similarity, taking into account both semantic and structural similarities.

**Definition 1 (Feature similarity).** The feature similarity of features \(f_1\) and \(f_2\) is the average of their semantic similarity, the similarity of their child features, and the similarity of their parent features. Formally expressed:

\[
\text{Sim}(f_1, f_2) = \frac{\text{NSim}(f_1, f_2) + \sum_{f'} \max(\text{Sim}(f_1, f'), \text{Sim}(f', f_2)) + \max(\text{Sim}(f_1, f), \text{Sim}(f, f_2))}{m + 2}
\]

where:
- \(\text{NSim}(f_1, f_2)\) is the semantic (name) similarity of features \(f_1\) and \(f_2\).
- \(f_1, f_2\) are child features of \(f_1, f_2\), respectively (in terms of mandatory, optional, OR, and XOR relations).
- \(m\) is the number of \(f_1\) child features multiplied by the number of \(f_2\) child features.
- \(f_1, f_2\) are parent features of \(f_1, f_2\), respectively.

Four important characteristics of the above formula (Definition 1) are: (1) the value of similarity is always in the range of 0 (completely different) to 1 (identical); (2) if the features have no similar child features neither similar feature parents, the overall similarity equals their name similarity; (3) the similarity of features increases proportionally to the degree of similarity of their descendants; and (4) the similarity of features increases proportionally to the degree of similarity of their descendants. Note that for performing the actual calculation of feature similarity two passes on the feature diagram are required: one from the leaves to the root (taking into account the feature’s semantic similarity and the descendants’ similarities) and another pass in the opposite direction (for percolating ascendants’ similarities).

As an example to feature similarity calculation consider the two portions of feature diagrams of mobile phone PLs in Fig. 1. As can be seen, these two PLs differ in the features they support (e.g., the second PL does not support mp4), the ways they structure the features (e.g., calls appear in the first PL under utility functions, whereas in the second PL it appears directly under the diagram root), and the terminologies they use (e.g., ‘extras’ vs. ‘media’, ‘screen’ vs. ‘display’). Adopting Dao and Simpson’s measure \(^{10}\), which is a knowledge-based measure, the semantic similarity of the features “messaging” and “sending utility” is 0.64. The feature similarity of the two features increases to 0.72 due to their structural similarity, namely both include message-related child features (text and voice messages in the first PL and SMS, MMS, and EMS in second PL) and have “mobile phone” parent features.

### 3.2. Feature clustering

For grouping “similar enough” features, we apply a variation of the agglomerative hierarchical clustering technique \(^{20}\). This algorithm is a bottom-up approach that starts with putting each object (feature in our case) in a separate cluster. Then, in each iteration, the algorithm agglomerates (merges) the closest pair of clusters by calculating the distance between the different clusters. We specifically use the complete-link type to measure the distance between clusters. This distance type, which measures the distance (similarity) between the two farthest (least similar) features of the clusters, is especially suitable to our case, since we aim to create a refined division of features to clusters and avoid merging features that are not similar enough to the same cluster. The algorithm continues merging the closest clusters until the distance between the closest clusters is not larger than a pre-defined similarity threshold (see simth in Appendix A). To determine the similarity threshold, sensitivity and specificity analysis are conducted \(^{3}\). These statistical measures assess the performance of a binary classification test: sensitivity measures the proportion of actual positives which are correctly identified as such (in our case, the percentage of similar features which are correctly identified), while specificity measures the proportion of negatives which are correctly identified as such (in our case, the percentage of non-similar features which are correctly identified). The correct identification is checked with respect to domain experts’ classification. Details on how we calculated this similarity threshold for the domain of mobile phones can be found in the evaluation section (Section 5.1).

The feature clustering step may create the following clusters in our example of the two PLs of mobile phones (Fig. 1): \(C_1\) = [voice call, calls]; \(C_2\) = [High resolution, Low resolution, color, colour, basic]; \(C_3\) = [screen, display]; \(C_4\) = [media, extras]; \(C_5\) = [camera, mp3, mp4]. Note that some of the clusters were mainly created due to the high semantic similarity of the features, potentially indicating on synonyms (e.g., cluster \(C_2\)), whereas some clusters emerged due to similar feature contexts (e.g., \(C_4\)). In other cases, both semantic and structural similarity contributed to the cluster creation.

### 3.3. Domain model creation

The generated clusters are the nodes of the domain model. The edges (relationships) between clusters are induced by the input feature diagrams: if a relationship exists in an input feature diagram between two features which are clustered into separate clusters, the same relationship will appear in the domain model between the corresponding clusters. In case of OR or XOR (alternative) relationships in which all the child features are grouped into the same cluster, the type of relationship between the corresponding clusters is changed to “mandatory” (as one of the features that belong to the child cluster must be selected). Note that there may be relationships of different types between the same pair of clusters, potentially in opposite directions, and the domain models may include cycles. Since constraints may be relevant only to the particular features they connect and not to the whole clusters, we percolate the constraints into the domain model as they are, without generalizing them to clusters.

After creating the nodes (clusters) and the edges (relationships) of the domain model, we associate to each edge its local and global strengths, which can be later used for generating improvement recommendations. The local and global strengths are defined as follows:

**Definition 2 (Local strength).** The local strength of a relationship of type \(rel \in \{\text{mandatory}, \text{optional}, \text{XOR}, \text{OR}\}\) from cluster \(C_1\) to cluster \(C_2\) is defined as the ratio between the number of PLs involved in this type of relationship and the total number of PLs whose features appear in at least one of the clusters (\(C_1\) or \(C_2\)). Formally expressed:

\[
\text{Strength}_{\text{local}}(C_1, C_2, rel) = \frac{|\{pl | \exists f_1, f_2 \in pl \text{ such that } f_1 \in C_1, f_2 \in C_2, rel_{f_1, f_2} \in pl\}|}{|\{pl | f \in pl \text{ such that } f \in C_1 \cup C_2\}|}
\]
Definition 3 (Global strength). The global strength of a relationship of type rel ∈ {mandatory, optional, XOR, OR} from cluster C1 to cluster C2 is defined as the ratio between the number of PLs involved in this type of relationship and the total number of input PLs. Formally expressed:

\[
\text{Strength}_{\text{global}}(C_1, C_2, \text{rel}) = \frac{|\{pl \mid f_1 \in C_1, f_2 \in C_2, \text{rel}(f_1, f_2) \in pl\}|}{|pl|}
\]

Fig. 3 is part of the domain model generated for mobile phones, using nine different PLs: seven of these diagrams were taken from S.P.L.O.T. – an academic feature diagrams repository [29]. The two additional ones were specifically modeled to introduce some challenges to better evaluate the approach. In particular, we added synonyms and antonyms and we modeled the hierarchies of features using different nesting strategies. As can be seen, some of the clusters include features whose names can be considered synonyms (semantic similarity), e.g., game and play, display and screen. The reasons for the emergence of other clusters can be found in their similar descendants, e.g., connectivity and communication, or ascendants, e.g., basic, color, high resolution, and low resolution. It is worth paying a special attention to the way the connectivity-related features are clustered in this domain model. Bluetooth and USB are grouped together (and separately from the other connectivity-related features), since their overall similarity, which takes into account their semantic and structural similarities, was higher than the predefined similarity threshold. This mainly happened since Bluetooth and USB appeared under connectivity (or communication) in 5 out of the 9 input feature diagrams (as can be concluded from the global strength of the corresponding relation). The other connectivity-related features appeared in this location only in one feature diagram. Similarly, SMS, MMS, and EMS are clustered together partially due to the semantic similarity of their terms and partially due to the fact that they appear under the feature “message” or “messaging” in 4 out of the 9 input feature diagrams.

4. Recommendation generation

We now turn to the recommendations generated from the domain model. As the domain model is a directed graph which may include cycles, we traverse the domain model using Edmonds’ algorithm for finding minimal (or actually maximal) spanning trees [12]. We use the local strengths (rather than the global strengths) as weights, since they take into consideration only the PLs that explicitly refer to the relevant features (and relationships). Furthermore, in case of several relationships between the same pair of clusters in the same direction, we recalculate their “aggregated” local strength in order to avoid preference of a relationship over several slightly “weaker” relationships. The “aggregated” local strength follows the local strength formula without taking into consideration the relationship type, namely, it is calculated as the ratio between the number of PLs having a relationship between features grouped into the clusters and the total number of PLs whose features appear in at least one of the clusters.

While traversing the domain model using Edmonds’ algorithm, we generate recommendations that can be primarily divided into refinement and management recommendations. Refinement recommendations suggest improvement to particular PLs, and thus are of importance to PL engineers. Management recommendations offer changes that may affect the management of the entire set of PLs (e.g., recommending on merging or splitting of PLs). As such, management recommendations are of interest to PL managers and high-level managers who are usually responsible on several PLs. Fig. 4 presents the types of recommendations generated by the approach, while the rest of this section elaborates on those types. In order to make the approach in general and the recommendation generation in particular flexible, the approach uses different thresholds that can be tuned by managers, PL engineers, or domain experts. These thresholds are mentioned whenever appropriate and are further listed and explained in Appendix A. Appendix B summarizes the types of generated recommendations and their characteristics.

4.1. Refinement recommendations

Refinement recommendations aim at suggesting the addition of aspects or constraints that exist in the domain model and may be relevant to the specific PLs\(^\dagger\). The refinement recommendations are further divided into local and global additions, as well as restrictions.

\(^\dagger\) Note that the approach recommends only on additions and not on deletions or modifications of existing features or relationships, assuming that the input feature diagrams represent different and unique PLs, and preserving these differences is highly important.
such that there is no feature under the feature be, is further be features in using rel. The refinement of feature 2 formally defines local addition recommendations.

The non-included ‘different enough’ child features are rec-
ed features in the domain model. In particular, it identifies situations
in which (1) a parent feature is included in a PL (connectivity in our
example), (2) none of its child features is included (e.g., GPS and its child fea-
tures), and (3) the global strength of the relationship between the
parent feature’s cluster and the child feature’s cluster is greater than a
predefined local threshold (see locth in Appendix A), and (4) there
are “different enough” features in the child feature’s cluster that
are not included in the PL (Bluetooth in our example). In these situa-
tions, the non-included “different enough” child features are re-
commended for inclusion in the given PL. The following definition
(along with Fig. 5) formally defines local addition recommendations.

To avoid false recommendations in which features are recommended to PLs that already exhibit very similar features, we refer here to features whose similarity is greater than a predefined equivalence threshold (see eqth in Appendix A) as equivalent features. We do not recommend on adding features to PLs that already include equivalent features. Likewise the similarity threshold, the equivalent threshold is calculated using sensitivity and specificity analysis.

5 To avoid false recommendations in which features are recommended to PLs that already exhibit very similar features, we refer here to features whose similarity is greater than a predefined equivalence threshold (see eqth in Appendix A) as equivalent features. We do not recommend on adding features to PLs that already include equivalent features. Likewise the similarity threshold, the equivalent threshold is calculated using sensitivity and specificity analysis.

4.1.1. Recommending on local additions

Local additions aim to refine aspects that already exist to some extent in the given PL. For example, assume a mobile phone PL which has only USB connectivity and no Bluetooth. Based on the domain model in Fig. 3, we may want to recommend to our PL engineer to consider adding Bluetooth connectivity to that PL to improve its competitiveness, and potentially its compatibility to the other PLs in the domain.

To this end, the approach refers to the local strengths of relationships in the domain model. In particular, it identifies situations in which (1) a parent feature is included in a PL (connectivity in our example), (2) its child feature (USB in our example) is also included, (3) the local strength of the relationship between the parent feature’s cluster and the child feature’s cluster is greater than a predefined local threshold (see locth in Appendix A), and (4) there are “different enough” features in the child feature’s cluster that are not included in the PL (Bluetooth in our example). In these situations, the non-included “different enough” child features are recommended for inclusion in the given PL. The following definition (along with Fig. 5) formally defines local addition recommendations.

Note that according to the domain model depicted in Fig. 3, Bluetooth is already specified as an optional child of connectivity in 5 out of the 9 input PLs. Thus, this recommendation will not be generated for those PLs, but for the 4 other input PLs.

4.1.2. Recommending on global additions

Global additions aim to introduce new domain-related aspects to existing PLs. For example, assume a mobile phone without GPS. Although this lack may be deliberate in order to reduce prices, we may want to highlight this lack by recommending adding the GPS feature and its child features (not appearing in Fig. 3), e.g., speed detection and auto-correction of positions (A-GPS).

An example of such a recommendation may be: consider adding in PL#1 Bluetooth under Connectivity using an optional relation. Note that according to the domain model depicted in Fig. 3, Bluetooth is already specified as an optional child of connectivity in 5 out of the 9 input PLs. Thus, this recommendation will not be generated for those PLs, but for the 4 other input PLs.


definition 4 (Recommendations for local additions). Let C1, C2 be different clusters in the domain model such that Streng\(\text{h}_{\text{local}}(C1, C2, \sim) \geq \text{locth}\) and \(f_1 \subset C_1, f_2 \subset C_2\) be features in product line PL. For each \(f_2' \in C_2\) such that there is no feature \(f_2'\) in PL satisfying \(\text{sim}(f_2', f_2) \geq \text{eqth}\) (namely, no equivalent feature already exists in PL), recommend on a local addition of the form: consider adding in PL the feature \(f_2'\) under the feature \(f_1\) using rel.

An example of such a recommendation may be: consider adding in PL#1 Bluetooth under Connectivity using an optional relation. Note that according to the domain model depicted in Fig. 3, Bluetooth is already specified as an optional child of connectivity in 5 out of the 9 input PLs. Thus, this recommendation will not be generated for those PLs, but for the 4 other input PLs.


definition 5 (Recommendations for global additions). Let C1, C2 be different clusters in the domain model such that Streng\(\text{h}_{\text{global}}(C1, C2, \sim) \geq \text{glbth}\) and \(f_1 \subset C_1\) be a feature in product line PL satisfying \(C_1 \cap \text{PL} = \emptyset\). For each \(f_2 \subset C_2\), recommend on a global addition of the form\(^6\): consider adding in PL the feature \(f_2\) under the feature \(f_1\) using rel. The refinement of feature \(f_2\) is further recommended using globally strong relationships.

\(^6\) Note that since \(C_2 \cap \text{PL} = \emptyset\), we do not recommend on features equivalent to features that already exist in PL (otherwise those features would be clustered in \(C_2\).
An example of such a recommendation may be: consider adding in PL#j GPS under mobile phone using an optional relation. Consider further adding Speed Detection under GPS using mandatory relation. Consider further adding Auto-Correction of Positions under GPS using optional relation.

4.1.3. Recommending on restrictions

As noted, constraints are defined in the feature level and are not generalized to constrain clusters. However, we can recommend on restrictions for equivalent features, i.e., features whose similarity is greater than the predefined equivalence threshold. As an example, consider the constraint that a ‘camera’ requires a ‘high resolution’ screen. This constraint can be recommended to PLs that include camera, refer to screen resolution (even if it is termed display resolution), but do not restrict the allowable configurations. However, PLs that include only low resolution screens cannot benefit from this restriction, even though they refer to screen resolution.

Locally strong constraints, namely, constraints whose variants (with different equivalent features) appear in at least a certain number (local threshold, locth in Appendix A) of PLs can be percolated to equivalent features in other PLs that include the involved features (or equivalent counterparts). Furthermore, a global recommendation may also yield a restriction recommendation as it introduces new aspects that may be related to locally strong constraints. The following definition (along with Fig. 7) formally defines restriction recommendations.

**Definition 6** (Restriction recommendations). Let \( f_1, f_2 \) be features in product line PL and \( C_1 = \{f \text{ in the domain model} \mid \text{sim}(f, f_1) \geq \text{eqth}\} \). \( C_2 = \{f \text{ in the domain model} \mid \text{sim}(f, f_2) \geq \text{eqth}\} \). Let \( \text{Strength}_{local}(C_1, C_2, \text{require}) \geq \text{locth} \). Then:

1. If \( \text{Strength}_{local}(C_1, C_2, \text{require}) \geq \text{locth} \), generate a restriction recommendation of the form: consider adding in PL that feature \( f_1 \) requires feature \( f_2 \) (if this restriction is not included).
2. If \( \text{Strength}_{local}(C_1, C_2, \text{require}) \geq \text{locth} \), generate a restriction recommendation of the form: consider adding in PL that feature \( f_1 \) excludes feature \( f_2 \) (if this restriction is not included).

An example of such a recommendation may be: consider adding in PL#k the restriction that Camera requires High Resolution Display.

4.2. Management recommendations

PL managers, as well as high-level managers who are responsible on several PLs, may be interested in questions relevant to the whole set of PLs and not just to the improvement (through refinement) of specific PLs. Examples of such questions may be: When are different PL candidates for being merged? When is a PL candidate for being split? Which features deserve high development priority, as they are in the basis of several PLs, or alternatively they are unique and may provide competitive benefits? To this end, the generated domain model may further assist in recommending on improvements to the management of the entire set of PLs.

4.2.1. Recommending on feature-level improvements

Feature-level improvements refer to recommendations on handling similar, core, and unique features. Similar features, namely, features with similar names which appear in similar contexts, may call for their co-development and co-maintenance. As depicted in the following definition, similar features can be extracted from the clustering data as each cluster is a group of similar features.

**Definition 7** (Similar features recommendations). For each cluster \( C = \{f_i\}_{i=1 \ldots n} \) in the domain model, generate a management recommendation of the form: consider co-developing and co-maintaining the features \( f_i \).

An example for such a recommendation, generated based on the domain model in Fig. 3, may be: consider co-developing and co-maintaining the features Short Message Service (SMS), Multimedia Message Service (MMS), and Enhanced Message Service (EMS).

Similar features that appear in many PLs can be considered core. Their development and maintenance need to be prioritized as different PLs exhibit them. We use the global threshold (see glbth in Appendix A) in the following definition for determining the minimal number of PLs for considering a feature as a core feature.

**Definition 8** (Core features recommendations). For each feature \( f \) such that the number of PLs in which \( f \) or an equivalent feature \( f’ \) satisfying \( \text{sim}(f, f’) \geq \text{eqth} \) exists is greater than \( \text{glbth} \), generate a management recommendation of the form: feature \( f \) and all the equivalent features \( f’ \) are core features; consider prioritizing their development and maintenance.

An example for such a recommendation, based on the domain model in Fig. 3, may be: feature Connectivity and all the equivalent features (Communication) are core features; consider prioritizing their development and maintenance.

Unique features, on the other hand, are those features that appear only in a low number of PLs. They may represent old (depreciated) features which are no longer in use in the domain and thus their deletion should be considered, or innovative features which are not yet included in other PLs. They can also indicate on particular features that are relevant to a specific PL and are not part of the domain knowledge. Innovative features should be recommended to be introduced into other PLs and a recommendation to concentrate on the development of these features may be relevant and result in competitive benefits. By highlighting such features, the manager can further investigate them and decide whether they are relevant to other PLs or not.

In order to define unique features, the approach uses a unique threshold (see unqth in Appendix A). Features that appear in less than the unique threshold percentages of the input PLs are considered unique. The following definition formally defines this kind of management recommendations.

**Definition 9** (Unique features recommendations). For each feature \( f \) such that the number of PLs in which \( f \) or an equivalent feature \( f’ \) satisfying \( \text{sim}(f, f’) \geq \text{eqth} \) exists is less than \( \text{unqth} \), generate a management recommendation of the form: feature \( f \) and all the equivalent features \( f’ \) are unique features; determine whether they are old, innovative, or specific features.

As an example, consider the eye tracking capability of Samsung Galaxy S4 (not included in Fig. 3), which enables navigating the phone with the eyes. A manager may be interested to find that this
is a unique (innovative) feature that deserves further development in order to achieve some competitive benefits. Thus, the following recommendation may be of great interest: feature Eye Tracking is a unique feature; determine whether it is an old, innovative, or specific feature.

4.2.2. Recommending on PL-level improvements

Besides the feature-level improvements that can be recommended, PLs management can benefit from PL-level improvements. In particular, too similar PLs can be merged to save development and maintenance efforts. In addition, PLs that treat aspects that appear in several PLs (potentially in different variants) can be split and handled separately, increasing modularity and reusability.

In order to identify PLs that are candidate to be merged, we calculate the difference between the clusters they are involved in. To this end, we represent each PL as a binary vector whose length is the number of clusters in the domain model. If the PL has at least one feature that belongs to a specific cluster, then the corresponding cell gets the value 1. Otherwise the value 0 is assumed. The degree of similarity between PLs is calculated as 1 minus the differences between the corresponding vectors. High degree of similarity between PLs means that the two PLs share many similar features, and thus may be candidates for merge. In other words, if the similarity between two PLs is higher than a predefined merge threshold (see $mr_{th}$ in Appendix A), then a recommendation on merging the PLs has to be generated. Note, however, that highly similar PLs does not necessarily imply identical PLs, so the manager needs to decide whether merging the PLs is indeed beneficial in the given situations. The following definition formally defines this kind of management recommendations.

**Definition 10 (Recommendations on PLs merge).** Let $PL_1$, $PL_2$ be two product lines, such that $\text{sim}(PL_1, PL_2) = 1 - (V(PL_1) - V(PL_2)) > mr_{th}$, where $V(PL_i)$ is a vector in which the $j$th component is 1 if a feature from cluster $j$ exists in $PL_i$ and 0 otherwise and $mr_{th}$ is the merge threshold. In this case, generate a management recommendation of the form: consider merging $PL_1$ and $PL_2$.

In the context of merging PLs, the similarity of Samsung Galaxy S and Google Nexus may justify their merge in a scenario of companies merge or acquisition.

In order to identify PLs that are candidate to be split, we search for sub-trees in the domain model that are “significant” enough (in terms of the number of features, see $min_{th}$ in Appendix A) and appear in several PLs (see $split_{th}$ in Appendix A). We recommend merging such sub-trees as separate PLs in order to increase reusability and avoid redundancy. The following definition formally defines this kind of management recommendations.

**Definition 11 (Recommendations on PLs split).** Let $S$ be a sub-tree of the domain model, such that $|S| > min_{th}$, namely the number of features in $S$ is greater than the minimal threshold that justifies a separate PL. For each such sub-tree satisfying $|S| > split_{th}$, namely the number of PLs including sub-trees homomorphic to the given sub-tree $S$ is greater than a predefined threshold, generate a management recommendation of the form: consider splitting $S$ and managing it as a separate product line.

As an example, consider media properties which include camera, mp3 and mp4, and their characteristic (not fully presented in Fig. 3). These features appear in many mobile phone PLs, as well as tablets and other mobile devices. Thus, they should be considered to be handled separately, so all the PLs will benefit from this co-management. The corresponding recommendation may be: consider splitting media properties and managing them as a separate product line.

5. Evaluation of the approach

In order to evaluate the approach, we implemented a prototype which receives a set of feature diagrams in the Simple XML Feature Model (SXF) format [29], analyze them and generate a textual output including the domain model in a SXFM format and a set of recommendations. The prototype was developed on UNIX platform and was written in Perl using WordNet::Similarity::wup [22] for calculating the semantic similarity of features.

The prototype’s outcomes were evaluated in two steps. First, in order to examine the feature clustering, which is the basis for constructing the domain model and generating the recommendations, we measured the difference between the clusters generated by the approach and expert advice. Second, in order to evaluate the usefulness of the generated recommendations, we requested experienced users to rank their degree of agreement with different statements on the domain of discourse. These statements were derived from the recommendations generated by the approach.

We used two different data sets in the evaluations, both within the domain of mobile phones. The first data set was mainly taken from the academic repository S.P.L.O.T. [29]. This repository includes different feature diagrams, potentially in the same domain, which were independently created by different modelers using different terminologies. Furthermore, the criteria for including feature diagrams in the repository, as published in S.P.L.O.T. web site, guarantee that all the models are consistent (contain at least one valid configuration), do not contain dead features, and identify their authors (or related literature) for providing some contact information. The feature diagrams used in the academic data set are listed in Table 2.

The second data set included feature diagrams created from different comparison web sites of mobile phones. These web sites present the products (mobile phones) and their families. The features of the different products are hierarchically structured. Thus, the creation of feature diagrams from these web sites was straightforward: mandatory features are those that appear in all products in the line, optional features are those that appear in several products in the line but not in all of them, alternative features are those presented as single-choice (value) properties, and OR-grouped features are those presented as multi-choice properties. The created feature diagrams were much larger (in terms of features) than those in the academic data set, their variability was higher, and they describe “real” products (mobile phones). However, these feature diagrams did not include constraints, which were not explicitly presented in the corresponding web sites. The feature diagrams used in this data set, termed “commercial,” are listed in Table 2.

The rest of this section elaborates on the evaluation procedure, its results, and the threats to validity.

5.1. Evaluation of the domain model’s clusters

In order to calculate the thresholds listed in Appendix A in general and the similarity threshold in particular, we created a list including all the features in the input diagrams of the academic set. We further calculated the semantic similarity between each pair of features in the list. We then sampled 100 pairs whose values of semantic similarity ranged from low (almost 0) to high (almost 1). Six human graders, who have strong technical
Table 2

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Creator</th>
<th># of features</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Academic</td>
<td>Mobile-Phone</td>
<td>IR</td>
<td>10</td>
<td>model_20120110_139114401.xml</td>
</tr>
<tr>
<td></td>
<td>Mobile Phone</td>
<td>SS</td>
<td>20</td>
<td>model_20100322_955726153.xml</td>
</tr>
<tr>
<td></td>
<td>Phone</td>
<td>AU</td>
<td>25</td>
<td>model_20101119_1472596180.xml</td>
</tr>
<tr>
<td></td>
<td>Mobile Phone</td>
<td>M</td>
<td>10</td>
<td>model_20101111_1790887308.xml</td>
</tr>
<tr>
<td></td>
<td>Example Mobile phone</td>
<td>LE</td>
<td>10</td>
<td>model_20120110_1719396361.xml</td>
</tr>
<tr>
<td></td>
<td>Cell Phone</td>
<td>R</td>
<td>10</td>
<td>model_20120110_1094246588.xml</td>
</tr>
<tr>
<td></td>
<td>Mobile Phone 1</td>
<td>SO</td>
<td>15</td>
<td>model_20100308_1032655961.xml</td>
</tr>
<tr>
<td></td>
<td>Mobile Phone 2</td>
<td>Self</td>
<td>21</td>
<td><a href="http://www.samsung.com/us/mobile/cell-phones">http://www.samsung.com/us/mobile/cell-phones</a></td>
</tr>
<tr>
<td></td>
<td>Galaxy Note: Galaxy Note II, Galaxy Note 10.1, Galaxy Note 8.0</td>
<td>Self</td>
<td>222</td>
<td><a href="http://www.samsung.com/us/mobile/galaxy-note">http://www.samsung.com/us/mobile/galaxy-note</a></td>
</tr>
<tr>
<td></td>
<td>Mobile Phone AU</td>
<td>Self</td>
<td>25</td>
<td>model_20101119_1472596180.xml</td>
</tr>
<tr>
<td></td>
<td>Mobile Phone SS</td>
<td>Self</td>
<td>20</td>
<td>model_20100322_955726153.xml</td>
</tr>
<tr>
<td></td>
<td>Mobile Phone SO</td>
<td>Self</td>
<td>15</td>
<td>model_20100308_1032655961.xml</td>
</tr>
<tr>
<td></td>
<td>Mobile Phone IR</td>
<td>Self</td>
<td>21</td>
<td><a href="http://www.samsung.com/us/mobile/cell-phones">http://www.samsung.com/us/mobile/cell-phones</a></td>
</tr>
</tbody>
</table>

1 Note that the web sites are dynamically changing and thus may not include the information utilized to generate the feature diagrams used in our evaluation. The relevant evaluation material can be found at http://mis.hevra.haifa.ac.il/~iris/research/crossPLanalysis/.

2 This is a configurable product and thus can be considered an SPL by its own.

Fig. 8 shows a histogram of the number of pairs classified by humans as similar or different for each range of semantic similarity values. As can be noticed, many of the feature pairs whose similarity value falls into the range of 0.7–0.8 were classified as different by human judges. Only around a similarity value of 0.9, all pairs of features whose similarity value is above this value were classified by human judges as similar. Thus, we further examined the range 0.8–0.9 using sensitivity and specificity analysis [3], in order to determine the exact similarity threshold. As shown in Table 3, for similarity threshold higher than 0.87, we got the highest sensitivity (predicting all similar pairs as similar) and specificity (not predicting different pairs as similar) values. We therefore used a similarity threshold of 0.88 in all the evaluations.

The evaluation of the domain model’s clusters was done on the larger (commercial) data set which included a total number of 306 distinct features. We compared the clusters generated by our prototype (using a similarity threshold of 0.88 and the other thresholds values defined in Appendix A) with the clusters suggested by two experts in the domain of mobile phones. Both experts have worked as architects in a leading company in the communication sector and had between 9 and 11 years of professional experience in the field. The experts clustered the features independently.

In order to measure the differences between the clusters generated by the approach and those suggested by the experts, we used Rand index [24], which receives three input files: a list of all unique objects (features in our case) and two sets of clusters, U and V, on these objects. The algorithm counts the number of pairs that are placed in the same cluster in U and V (type a), the number of pairs that are placed in different clusters in U and in different clusters in V (type b), the number of pairs that are placed in different clusters in U and in the same cluster in V (type c), and the number of pairs that are placed in different clusters in V and in the same cluster in U (type d). Types a and b are interpreted as agreement in clustering, while types c and d represent disagreements. The overall Rand index is calculated as follows:

$$R = \frac{a+b}{a+b+c+d}$$

where \( n \) is the number of unique objects (features).

The higher Rand index is the better correspondence between the two compared sets of clusters is. Steinley [30] suggested the following heuristics for determining the quality of clustering: (a) values greater than 0.90 are viewed as excellent clustering; (b) values greater than 0.80 are considered good clustering, (c) values greater than 0.65 reflect moderate clustering, and (d) values smaller than 0.65 reflect poor clustering.

Table 4 presents the results of executing Rand index algorithm in our case. Our approach resulted in 170 clusters, each of which included 1–25 features. The resultant Rand index values in both cases (the two experts who independently performed the clustering) were very high (about 0.98 in both cases), indicating an excellent correlation between the expert advice and the output of our approach. Despite the high Rand index, we further measured the correlation between the clusters of the two experts in order to ensure that the experts’ advice are not too subjective. The received Rand index was very high – about 0.98 too – indicating on excellent agreement on the domain terminology.

5.2. Evaluation of the generated recommendations

In order to evaluate the generated recommendations we used the academic data set whose feature diagrams were created by different creators and included explicit constraints[3]. The approach
resulted with 29 clusters, each of which included between 1 and 19 features. In addition, 47 recommendations were generated: 21 refinement recommendations and 26 management recommendations. Merging recommendations which refer to the same features (for different PLs) and filtering too "trivial" recommendations (e.g., co-managing very similar features) or recommendations that refer to "old" features (e.g., winCE and Symbian operating systems), we ended with 12 recommendations: 6 - management recommendations and 6 - refinement recommendations. We phrased these recommendations as statements which were supposed to be more comprehensible to mobile phone users. For example:

(1) A mobile phone which supports messaging services is likely to support SMS (short message service).
(2) mp3, mp4 and camera are all different types of media capabilities.

We then asked 50 mobile phone users with technical background (they all were information systems students) to grade their degree of agreement with each statement, on a scale of 'completely agree', 'partially agree', and 'disagree'. The respondents could also mark that they have no sufficient information on the subject and hence cannot decide on their degree of agreement with the statement ("don't know").

Analyzing the respondents' background, we found that four respondents worked in service departments of mobile phone companies for several years. The other 46 respondents were familiar with the domain as users for 10 years on the average. Most respondents evaluated their familiarity with the domain as 'very good' (18) or 'good' (22), and only a few of them claim to have 'poor' familiarity (5) or being 'unfamiliar' (1) with the domain.

The analysis of the responses reveals that overall in about 80% of the cases the respondents agreed with the statements (in 59% of the cases they completely agreed and in 20% they partially agreed). Only in 16% of the cases the respondents disagreed with the specified statement. Dividing the questions according to their recommendation type, we found similar degrees of agreement in statements based on management recommendations and in statements derived from refinement recommendations (see Fig. 9).

Analyzing the degree of agreement, we noticed a low degree of agreement ("partially agree") on four specific statements. The first statement referred to 'screen' and 'display' as synonyms in mobile phones. While this is the case in our input feature diagrams (name similarity of 0.95 and very similar contexts), many respondents probably interpreted 'screen' as the physical device that is characterized by resolution, size, etc., while 'display' was interpreted as referring to the way the mobile phone visualizes applications (e.g., using different drivers). The second statement claimed that two common settings of a mobile phone are its operating system and its support for Java. Here we believe that many respondents tend to refer to settings as features (or parameters) that can be controlled by users, but are features relevant to the configuration of mobile phones. The third statement claimed that mobile phones with basic functions are most likely to have games. Here we believe that the respondents only partially agreed with this statement taking into consideration different kinds of mobile phones, the simple of which do not include games or game support at all. Finally, the fourth statement claimed that high resolution, low resolution, color, and basic are all characteristics of mobile phone screens. The respondents may expect to see two characteristics of screen: resolution (high, low) and type (basic, color). Another explanation may be that this statement was the first one in the questionnaire. Respondents usually tend to be more doubtful and cautious at the beginning of unknown tasks. A possible evidence for this explanation may be the low degree of disagreement with this statement.

5.3. Threats to validity

Although our results are promising, the evaluation of the approach has to be discussed under the following threats to validity. First, the evaluation focused only on one domain – mobile phones. Additional, more complicated domains need to be examined, as well as different sources of feature diagrams need to be explored. Second, we mainly examined a single point of view – that of mobile phone users. Other points of view, and especially those of developers and maintainers, need to be explored as well. Third, we used only feature diagrams as inputs for the approach. Additional artifacts of the PLs, e.g., requirements, feature specifications, design models, and code may differently contribute to the domain knowledge extraction and influence the generated recommendations.
Thus, future examination of these artifacts and their influence on the generated recommendations needs to be explored. Forth, we restricted our evaluation of the generated recommendations to 6 management recommendations and 6 refinement recommendations. In particular, we have not examined restriction recommendations (based on constraints) and management recommendations calling to split PLs. We further rephrased the recommendations into statements that may be clearer to end-users but may introduce inaccuracies with respect to development and maintenance. Replicating the evaluation with different recommendations of various types and additional types of stakeholders is advised. Finally, we could not assess the reasons for the differences in the degrees of agreements and especially between ‘partially agree’ and ‘completely agree’. Qualitative research on this subject may complement the conclusions currently drawn.

6. Summary and future research

As the complexity and variety of systems have increased and the need to develop systems quickly and cheaply has grown, companies develop and maintain various Product Lines (PLs). In this work we call for increasing reusability and improving management of PLs by sharing knowledge and artifacts from different PLs in the same domain. To this end, we introduce an approach for extracting domain knowledge and generating domain models from feature diagrams, each of which represents a different PL (aspect). The resultant domain models do not simply merge the input feature diagrams, but consolidate similar features into the same cluster and analyze cluster relationships. Based on the domain models and conducting cross PL analysis, refinement and management recommendations are generated for locally or globally introducing new aspects to PLs, restricting existing PLs, highlighting similar, core, and unique features, and recommending on merging or splitting of PLs. Evaluation results point that the approach’s outcomes may reflect human perception of the examined domain.

Future research can include several directions. First, the calculation of feature similarity can be improved by examining additional techniques for measuring semantic similarity, particularly those that are based on Wikipedia or on other dynamically growing web sources. A second direction may be expansion of the recommendation types to include other PL management concerns. Third, introducing AI and machine learning algorithms to the recommendation generation process may increase the accuracy and correctness of the given recommendations. This can be done by taking into consideration experts’ feedback. Fourth, additional information on features, such as documentation and code, may help predict future trends of similar features and thus improve the management of different PLs in the domain. Finally, additional quantitative and qualitative research needs to be done to empirically generalize the results and conclusions of this work to different settings.

Appendix A. Summary of thresholds used in the approach

<table>
<thead>
<tr>
<th>Threshold name</th>
<th>Abbreviation</th>
<th>Description</th>
<th>Value</th>
<th>Explanation on value selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similarity threshold</td>
<td>Simth</td>
<td>A threshold for considering two features similar enough to be included in the same cluster</td>
<td>0.88</td>
<td>Based on 100 pairs of features classified by six human judgers and conducting sensitivity and specificity analysis</td>
</tr>
<tr>
<td>Local threshold</td>
<td>Locth</td>
<td>A threshold determining whether local additions should be recommended</td>
<td>0.5</td>
<td>Based on the assumption of representative PLs (at least half of the involved PLs include the relevant features)</td>
</tr>
<tr>
<td>Global threshold</td>
<td>Glbth</td>
<td>A threshold determining whether global additions should be recommended</td>
<td>0.5</td>
<td>Based on the assumption of representative PLs (at least half of the input PLs include the relevant features)</td>
</tr>
<tr>
<td>Equivalence threshold</td>
<td>Eqth</td>
<td>A threshold determining high degree of similarity (equivalence) between features</td>
<td>0.97</td>
<td>Based on 100 pairs of features classified by six human judgers and conducting sensitivity and specificity analysis</td>
</tr>
<tr>
<td>Unique threshold</td>
<td>Unqth</td>
<td>A threshold determining wheatear a feature is considered unique or not</td>
<td>0.1</td>
<td>Based on the assumption of representative PLs; existence in less than 10% of the input PLs indicates on uniqueness</td>
</tr>
<tr>
<td>Merge threshold</td>
<td>Mrgth</td>
<td>A threshold determining wheatear two PLs are candidates for merging</td>
<td>0.8</td>
<td>Based on the size of input feature diagrams and their variability; sharing more than 80% similar features makes the PLs candidates for merging</td>
</tr>
<tr>
<td>Minimal no. of features</td>
<td>Minth</td>
<td>A threshold determining the minimal number of features justifying a separate PL</td>
<td>10</td>
<td>Based on the size of input feature diagrams and their variability; Sharing at least 10 (similar) features justify PL split</td>
</tr>
<tr>
<td>Split threshold</td>
<td>Splth</td>
<td>A threshold determining the number of PLs sharing similar portions that justify PLs split</td>
<td>2</td>
<td>Based on the number of input feature diagrams; Two PLs that share similar portions may justify PLs split to increase reusability</td>
</tr>
</tbody>
</table>
Appendix B. The recommendation types generated by the approach

<table>
<thead>
<tr>
<th>Recommendation type</th>
<th>Description</th>
<th>Situation</th>
<th>Generated Recommendation</th>
</tr>
</thead>
</table>
| Refinement          | Local additions | Aim to refine aspects that already exist to some extent in the given PL | (1) The parent is included in the PL  
(2) The child is included  
(3) The local strength $\geq$ local  
(4) There are "different enough" children that are not included in the PL | Consider adding in PL the feature $f_2$ under the feature $f_1$ using rel' |
|                     | Global additions | Aim to introduce new domain-related aspects to existing PLs | (1) The parent is included in the PL  
(2) No child is included  
(3) The global strength $\geq$ glbth | Consider adding in PL $f_2$ under $f_1$ using rel; Consider further adding $f_1$ under $f_2$ using rel' |
|                     | Restrictions | Aim to make restrictions that were accidentally missed | (1) There is a domain constraint involving certain features  
(2) Equivalent features appear in the PL without a corresponding constraint | Consider adding in PL that feature $f_1$ requires/excludes feature $f_2$ |
| Management recommendation | Similar features | Aim to recommend on co-development and co-maintenance of similar features | Features that are grouped in the same cluster | Consider co-developing and co-maintaining the features $f_1$. Feature $f$ and all the equivalent features are core features; consider prioritizing their development and maintenance |
|                     | Core features | Aim to recommend on prioritizing features that appear in different PLs | Features that the number of PLs in which they (or equivalent features) appear $\geq$ glbth | Consider adding feature $f_2$ into PL $f_1$ and PL2 using rel' |
|                     | Unique features | Aim to recommend on adding innovative features and removing old ones | Features that the number of PLs in which they (or equivalent features) appear $< uqth$ | Consider merging PL1 and PL2 using rel' |
| PLs merge | PL split | Aim to recommend on merging too similar PLs | PLs whose (cluster-wise) similarity $\geq$ mrgth Sub-trees (S) in the domain model that are "significant" enough (i.e., their number of features $\geq$ minth) and appear in at least split PLs | Consider splitting S and managing it as a separate product line |
|                   |              | Aim to recommend on splitting PLs that include aspects relevant to other PLs | | |

References


