**ISBN** 978-93-84468-20-0
**Proceedings of 2015 International Conference on Future Computational Technologies (ICFCT'2015)**
*Singapore, March 29-30, 2015, pp. 229-235*

# Re-engineering Based Feature Model Management for Software Product Line

Win Pa Pa Tun[1], and Khin Mar Myo[2]

[1,2]University of Computer Studies, Mandalay
The Union of the Republic of Myanmar

[1]papawin009@googlemail.com

***Abstract*****:** *Nowadays, Software Product Line Engineering (SPLE) is an emerging software engineering paradigm, which is based on the concept of reusing software artifacts gaining from the previous software development lifecycle. Researches concerning with domain analyzing, feature modeling (FM), common and variability analyzing processes have being developed for SPLE. So, this system proposes re-engineering based feature model management system for software product line about mobile phone environment. This system is intended not only to develop an FM integration method for meta-modeling but also to develop a method for extracting Common & Variability from the versioned products. Firstly the features of products will be reversely extracted from the technical description file with matrix format. After that, these features will be used by FM Integration and Common & Variability Extraction. This system is useful for mobile phone environment. The proposed system is implemented by using Java programming language.*

***Keywords:*** *Software Product Line Engineering (SPLE), Feature Extraction, Feature Model, Mobile Phone.*

## 1. Introduction

Software product line engineering is a process that delivers reusable components, which can be reused to develop a new application for the domain. Indeed, a software product line defines software product lines requirements, software architecture and a set of reusable components, shared by the products, which implements a considerable part of the products' functionalities. The purpose is to reduce the time and costs of production and to increase the software quality by reusing elements (core assets) which have been already tested and secured. These objectives can be realized by putting in common development artifacts such as requirement's documents, conception's diagrams, architectures, codes (reusable components), test's procedures, and maintenance's procedures. The general process of product lines is based on the reusability of requirements, architecture and components.

Software engineering benefits include reusability of requirements and their components, better analysis of requirements, another view on the requirements for the client, control of software quality, establishment of programming standards, a way to find and erase redundant implementations and complete reusable documentations. And last but not least, business benefits concern the reduction of production, maintenance and test costs (thanks to the reuse of commonalities between various products). Moreover, product lines generate a better efficiency in the processes and the possibility to improve the budget and time planning.

For the software product line, the proposed system is implemented as the re-engineering based feature model management system. This system proposes a FM extraction method by applying the high-level software artifacts instead of low-level ones. It will also provide a meta-model by integrating the feature matrix of product variants. This system also proposes a commonality and variability feature extraction method by means of meta-model.

This paper is organized into six sections. In the second section, related works are described. The background theory about the proposed system is expressed in the third section. And then, fourth section describes about the

proposed system. Experimental results are described in section five. Finally, conclusion is described in the sixth section.

## 2. Related Work

K. Yoshimura, F. Narisawa and K. Hashimoto [1] proposed an approach to suggest variability candidates across existing software products by analyzing the change history. A factor analysis technique is applied to analyze variability of the existing products and call the proposed approach "FAVE: Factor Analysis based Variability Extraction". The factor analysis is applied to changes between existing products and detects the co-change patterns that may indicate the variability of the product line. In the case study of the automotive engine-control system, four variability and their variation points are detected from the change history. The detected variability corresponds to the variability from the requirement viewpoint.

B. Klatt and K. Krogmann [2] proposed model-driven product consolidation into software product lines. Whether an existing SPL should be extended with an ad-hoc customized product (reactive approach) or a new SPL should be created by consolidating multiple products (extractive approach) –implementation level models of the existing products should be extracted in both cases. Such models contain the structural elements of the software, referable by other models and linking the original sources.

B. Zhang [3] presented the system about mining complex feature correlations from large software product line configurations. The input of our approach is product configurations separately documented in each product. The first process is configuration extraction which analyzes all existing product configurations and results into a configuration matrix consisting of selected features with their values (if given) across all products. Then the second process of data preparation adapts the information in the configuration matrix by unifying the data format and discretizing continuous feature values.

According to literature and concepts pointed out from the previous works, the proposed system intends to present the re-engineering based feature model management system.

## 3. Background Theory

In this section, the background theory about the feature extraction method and feature model creation method for the software product line.

### 3.1. Feature Extraction for Software Product Line

For feature extraction process, the proposed system uses the data mining methods. There are many data mining methods such as classification, association, clustering and so on. Among them, the proposed system uses the Frequent Pattern (FP)-Growth algorithm that is the association method.

#### 3.1.1. Frequent Pattern Growth (FP-Growth) Approach

Frequent pattern growth (FP-Growth) approach is an efficient approach for producing the frequent itemsets without generation of candidate itemsets. It is based upon the divide and conquers strategy [4]. Instead of generating a large number of candidates, the FP-growth approach preserves the essential groupings of the original data elements for mining. Then the analysis is focused on counting the frequency of the relevant data sets instead of candidate sets. Instead of scanning the entire database to match against the whole corresponding set of candidates in each pass, the method partitions the data set to be examined as well as the set of patterns to be examined by database projection. Such a divide-and-conquer methodology substantially reduces the search space and leads to high performance.

With the growing capacity of main memory and the substantial reduction of database size by database projection as well as the space needed for manipulating large sets of candidates, a substantial portion of data can be put into main memory for mining. New data structures and methods, such as FP-tree and pseudo-projection, have been developed for data compression and pointer-based traversal.

FP-growth may eliminate or substantially reduce the number of candidate sets to be generated and also reduce the size of the database to be iteratively examined, and therefore, lead to high performance [5]. The FP-growth approach consists of two steps:

- Constructing an FP-tree: The first step constructs a compact data structure called FP-tree that efficiently stores frequent patterns of a transaction database and enables efficient frequent pattern mining.

- Mining patterns using an FP-tree: The second step uses an FP-tree to recursively mine all frequent patterns [5].

### 3.1.2. FP-Growth Algorithm

FP-growth algorithm for discovering frequent pattern without candidate generation is as follows [6]:

**Algorithm:** FP-Growth. Mine frequent patterns using an FP-tree by pattern fragment growth.
**Input:** A transaction database, D; minimum support threshold, min-sup.
**Output:** The complete set of frequent patterns.
**Method:**

1. The FP-tree is constructed in the following steps.
   (a) Scan the transaction database D once. Collect the set of frequent items F and their supports. Sort F in support descending order as L, the list of frequent items.
   (b) Create the root of an FP-tree and label it as "null". For each transaction Trans in D do the following.

   Select and sort the frequent items in Trans according to the order of L. Let the sorted frequent items list in Trans be [p|P], where p is the first element and P is the remaining list. Call insert-tree ([p|P], T). If T has a child N such that N.item-name = p.item-name, then increment N's count by 1; else create a new node N, and let its count be 1, it parent link be linked to T, and its node-link to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert-tree (P, N) recursively.
2. Mining of an FP-tree is performed by calling FP-growth (FP-tree, Null), which is implemented as follows:

   **Procedure FP-growth** (Tree, $\alpha$)
   (1) **if** Tree contains a single path P then
   (2) **for each** combination (denoted as $\beta$) of the nodes in the path P
   (3) generate pattern $\beta \cup \alpha$ with support = minimum support of nodes in $\beta$;
   (4) **else for each** $a_i$ in the header of Tree {
   (5) generate pattern $\beta = a_i \cup \alpha$ with support = $a_i$. support;
   (6) construct $\beta$'s conditional pattern base and then $\beta$'s conditional FP-tree $Tree_\beta$;
   (7) **if** $Tree_\beta \neq \phi$ **then**
   (8) call **FP-growth** ($Tree_\beta$, $\beta$); }

## 3.2. Feature Model for Software Product Line

To create the feature model, this system uses the propositional formula. In software development, a feature model is a compact representation of all the products of the Software Product Line (SPL) in terms of "features". Feature models are visually represented by means of feature diagrams. Feature models are widely used during the whole product line development process and are commonly used as input to produce other assets such as documents, architecture definition, or pieces of code.

A SPL is a family of related programs. When the units of program construction are features—increments in program functionality or development—every program in an SPL is identified by a unique and legal combination of features, and vice versa. Feature models were first introduced in the Feature-Oriented Domain Analysis (FODA) method. Then, feature modeling has widely adopted by software product line community and a number of extensions have been proposed [8].

## 3.3. Traits of Feature Model

The traits of feature model syntax are the feature mandatory and optional features, and or- and xor-groups. These are as follows:

- **AND-groups:** An AND-group with parent f and children $f_1, \ldots, f_l$ entails implications from the parent f to children $f_i$ and vice-versa. Due to transitivity of implication we have that for any i, j = 1 . . l variable $f_i$ implies $f_j$. And-groups thus manifest themselves as cliques in the implication graph of $\phi$ (a clique is a

sub-graph in which any two vertices are connected by an edge). Candidates for maximal and-groups can be found by identifying maximal cliques. These can then be replaced by and-groups.

- **Mandatory features:** Mandatory features are logically indistinguishable from and-group members. All mandatory sub-features of a feature collapse into a single and-group.
- **OR-groups:** An OR-group with parent f and members $f_1, \ldots, f_k$ gives rise to the implication

$$f \rightarrow f_1 \vee \cdots \vee f_k \quad (1)$$

Recall that adding new terms to a disjunction weakens the formula (the set of solutions increases). Thus, if the above implication holds, so do many other implications with disjunctions comprising supersets of $f_1, \ldots, f_k$ enriched with arbitrary literals, e.g, $f \rightarrow f_1 \vee \cdots \vee f_k \vee f_{k+1}$. Consequently, it is needed to find all the minimal disjunctions of features implied by a parent. More precisely, given the parent feature f and all its children $f_1, \ldots, f_l$, it is needed to find all minimal disjunctions $f_1 \vee \cdots \vee f_k$ of a subset of $f_1, \ldots, f_l$ for which the following formula is a tautology:

$$\phi \rightarrow (f \rightarrow f_1 \vee \cdots \vee f_k) \quad (2)$$

Each of the disjunctions should be minimal in the sense that removing any literal from the disjunction would invalidate the above formula. Furthermore, $f_1, \ldots, f_l$ are children of f in G after cycle removal but before transitive reduction, so that all potential groups are detected.

It turns out that such minimal disjunctions can be computed by computing so called prime implicants of a formula. An implicant of a formula $\phi'$ is a conjunction of literals $l_1 \wedge \ldots \wedge l_n$ such that $l_1 \wedge \ldots \wedge l_n \rightarrow \phi'$ is a tautology. A prime implicant is an implicant that cannot be reduced by removing literals from it, in the sense that the resulting conjunction would not be an implicant of $\phi'$. As prime implicants are widely used in reliability analysis and in hardware synthesis, several efficient methods for computing them exist [48, 49] and can be used in tools to identify or-groups.

- **XOR-groups:** Every XOR-group is an OR-group, which can be characterized by prime implicants and that additionally requires that its members are mutually exclusive. Thus, if a prime implicant identifies an OR-group with members $f_1, \ldots, f_k$, and it also holds that

$$\forall_i = 1 \ldots k. \forall_j = 1 \ldots k, i \neq j. \phi \rightarrow f_i \barwedge f_j \quad (3)$$

then $f_i$'s form an XOR-group.

- **Optional features:** A feature g is an optional sub-feature of f if $g \rightarrow f$ holds and $f \rightarrow g$ does not [7].

## 3.4. Types of Feature Model

Current feature modeling notations may be divided into three main groups, namely that are basic feature models, cardinality-based feature models and extended feature models. These are as follows:

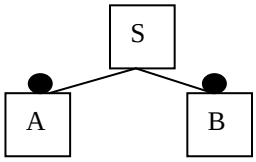
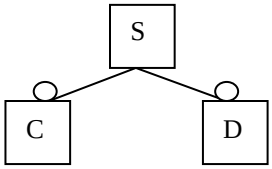
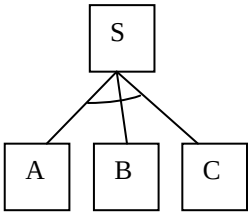
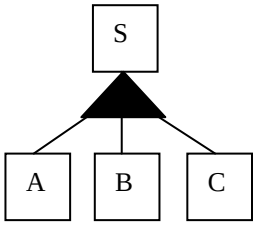
- **Basic Feature Model:** Relationships between a parent feature and its child features (or sub-features) are categorized as mandatory (child feature is required), optional (child feature is optional), Or (at least one of the sub-features must be selected) and Alternative (xor) (one of the sub-features must be selected). In addition to the parental relationships between features, cross-tree constraints are allowed. The most common are: A requires B (The selection of A in a product implies the selection of B) and A excludes B (A and B cannot be part of the same product).
- **Cardinality-based feature models:** Some authors propose extending basic feature models with UML-like multiplicities of the form [n,m] with n being the lower bound and m the upper bound. These are used to limit the number of sub-features that can be part of a product whenever the parent is selected. If the upper bound is m the feature can be cloned as many times. This notation is useful for products extensible with an arbitrary number of components [8].
- **Extended feature models:** Others suggest adding extra-functional information to the features using "attributes". These are mainly composed of a name, a domain, and a value [9].

## 3.5. Basic Feature Model using Propositional Formula

The semantics of a feature model is the set of feature configurations that the feature model permits. The most common approach is to use mathematical logic to capture the semantics of a feature diagram. Each feature corresponds to a boolean variable and the semantics is captured as a propositional formula. The satisfying valuations of this formula correspond to the feature configurations permitted by the feature diagram. For

instance, if $f_1$ is a mandatory sub-feature of $f_2$, the formula will contain the constraint $f_1 \Leftrightarrow f_2$. Propositional formula is shown in Table I.

TABLE I: Propositional Formula

| Concept | Diagram | Propositional Formula |
|---|---|---|
| Mandatory |   | $S \leftrightarrow B \ \wedge \ S \leftrightarrow A$ |
| Optional |   | $C \leftrightarrow S$ <br> $D \leftrightarrow S$ |
| OR |   | $(S \leftrightarrow A \vee B \vee C) \wedge$ almost1 (A, B, C) |
| X-OR |   | $S \leftrightarrow A \vee B \vee C$ |

## 4. Proposed System Design

For the software product line about the mobile phone domain, this system is implemented as the feature model management system by using Java programming language. In this system, there consists of three parts. These are as follows:

- Feature extraction,
- Feature diagram creation and
- Feature model creation.

In the feature extraction step, this system first loads each web page (html page) that consists of the technical description about the mobile phone. And then, mobile phone's features are extracted from the loaded web pages. After extracting each feature, this system uses the Frequent-Pattern (FP) growth algorithm. By using this algorithm, the proposed system creates the feature diagram about one brand of mobile phone. As a sample, HTC One Max is one brand of HTC mobile phone.

And then, this system creates the feature model by integrating each extracted feature diagram about each brand of mobile phone. In the feature model creation step, there are two sub-processes. In the first sub-process, this system performs the semantic analysis about feature from each feature diagram. And then, this system integrates each feature that has the same meaning. In the second sub-process, the propositional formula is used to define the relation between features. By using these propositional formulas, the proposed system defines each feature into mandatory, optional, or, alternative (xor) and so on. After defining each feature, this system produces the feature model about mobile phone to the user. Proposed system design is shown in Figure 1.

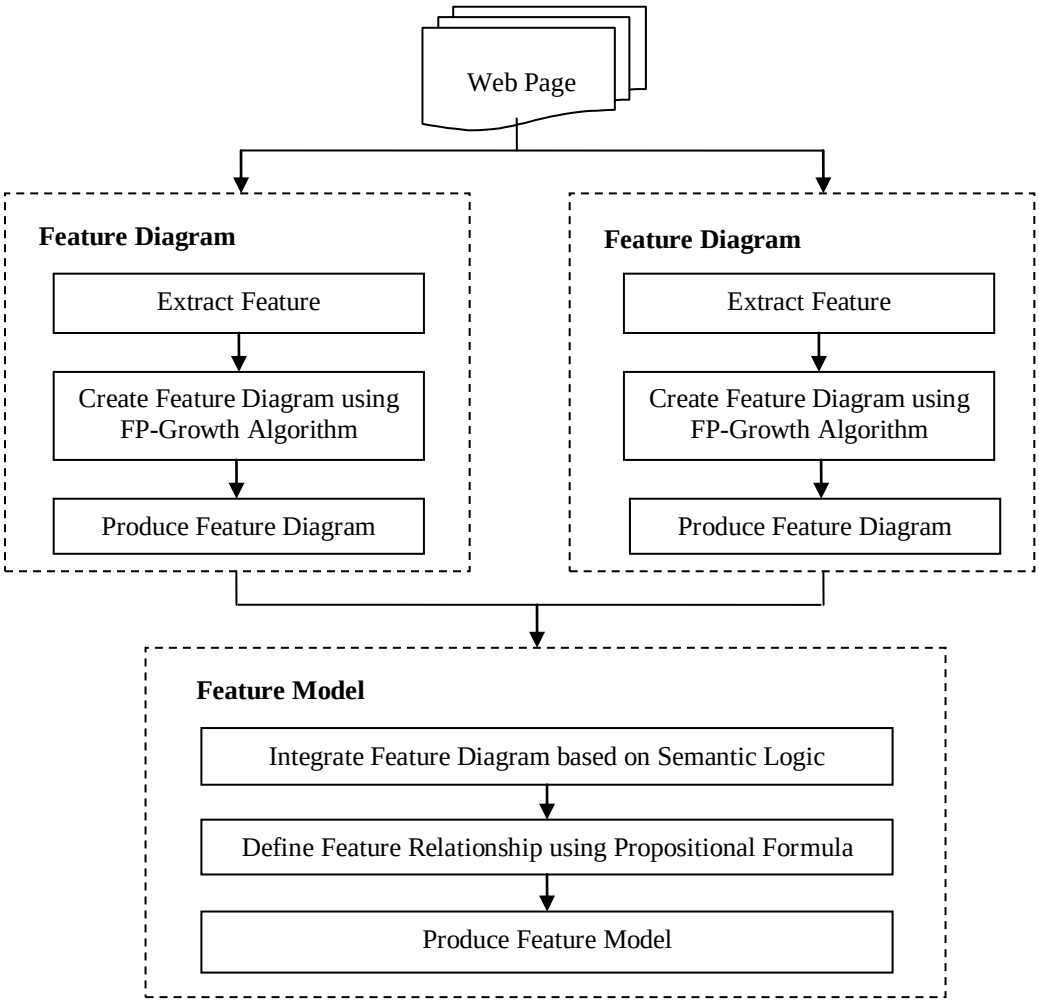



Fig 1: Proposed System Design

## 4.1. Sample Feature Model for HTC Mobile Phone

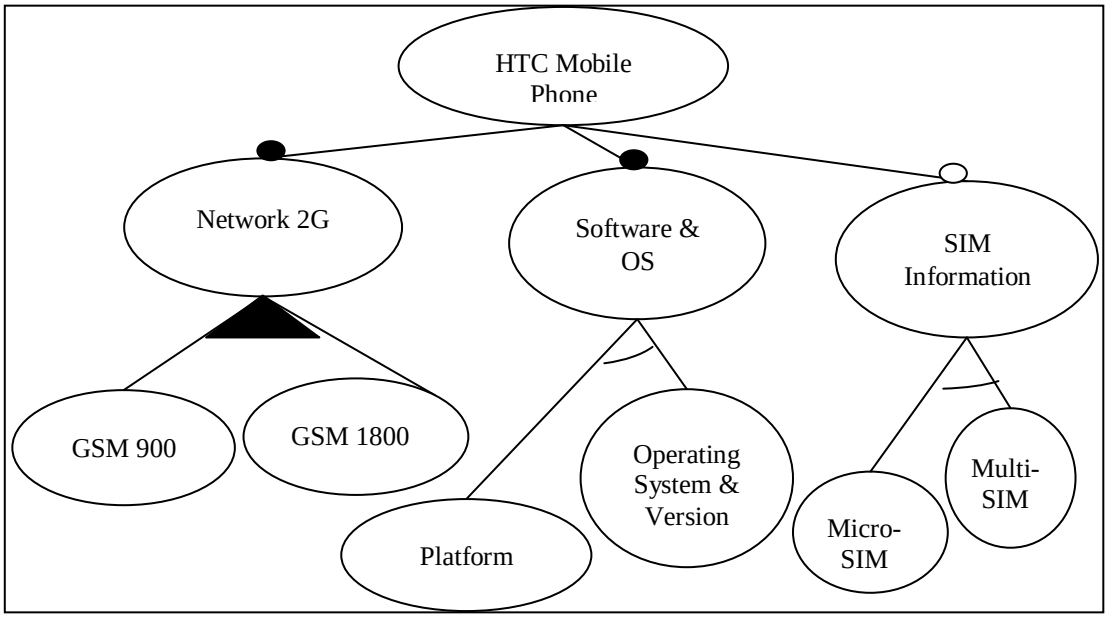



Fig 2: Feature Model for HTC Mobile Phone

## 5. Experimental Result

This system is tested by using different brands about mobile phone. These brands are HTC, LG, Nokia, Sony-Ericsson, and so on. This system first extracts mobile phone features and creates the mobile phone feature diagram. And then, each feature diagram is integrated into the feature model for mobile phone.

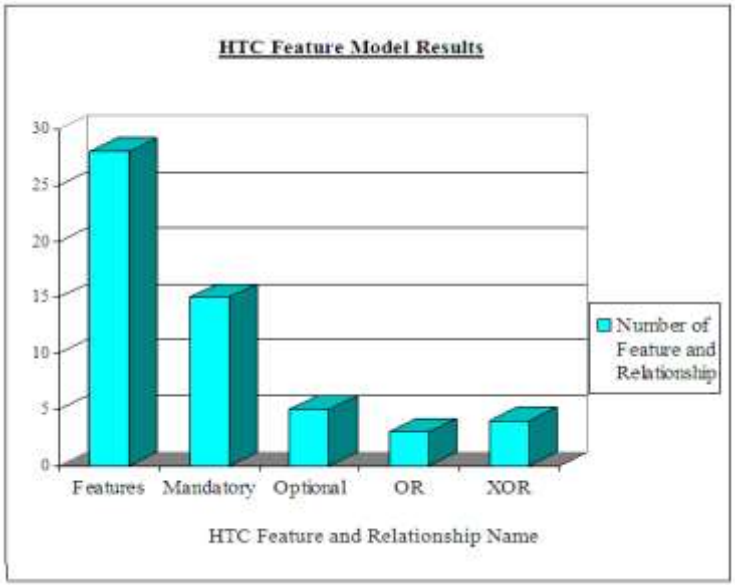



Fig 3: HTC Feature Model Results

## 6. Conclusion

The proposed system provides the software product line by producing the feature model about mobile phone. This system is tested by using many web pages that contain mobile phone information. This mobile phone information are about HTC, Nokia, LG and so on. By using the proposed system, the software developer can estimate the features about the coming mobile phone product. So, the proposed system provides many effective for the software product line about the mobile phone domain.

## 7. References


[1] K. Yoshimura, F. Narisawa and K. Hashimoto, "A Method to Analyze Variability Based on Product Release History: Case Study of Automotive System", Hitachi Research Laboratory, Hitachi, Ltd, Japan.

[2] B. Klatt, K. Krogmann, "Model-Driven Product Consolidation into Software Product Lines", FZI Forschungszentrum Informatik Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany, 2009.

[3] B. Zhang, "Mining Complex Feature Correlations from Large Software Product Line Configurations", Software Engineering Research Group, University of Kaiserslautern, Germany, Technical Report of AGSE, April 3, 2013.

[4] G. Kaur and S. Aggarwal, "Performance Analysis of Association Rule Mining Algorithms", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, August, 2013, pp. 856-858.

[5] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, USA. http://dx.doi.org/10.1145/342009.335372

[6] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", Simon Fraser University, United States of America, 2001.

[7] K. Czarnecki and A. Wasowski, "Feature Diagrams and Logics: There and Back Again", University of Waterloo, Canada, 2008.

[8] K. Czarnecki and S. Helsen and U. Eisenecker, "Staged configuration using feature models", Proceedings of the Third International Conference on Software Product Lines (SPLC '04), vol 3154, Springer, August 2004. http://dx.doi.org/10.1007/978-3-540-28630-1_17

[9] D. Benavides, P. Trinidad and A. Ruiz-Cortés. "Automated Reasoning on Feature Models". 17th Conference on Advanced Information Systems Engineering (CAiSE'05). Porto, Portugal. 2005. http://dx.doi.org/10.1007/11431855_34