Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 14 Issue 3 Version 1.0 Year 2014
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Mobilemedia SPL Creation by FeatureIDE using FODA

By Manjinder Kaur & Parveen Kumar

Lovely Professional University, India

Abstract- Software Product Lines are used in many areas, combining to form new technologies and products. A product line is a group of products that share a common development platform and vary by the composition and implementation method for the functionalities. This paper describes the implementation or creation of MobileMedia feature model using FODA (Feature Oriented Domain Analysis) methodology using FeatureIDE eclipse plug-in. The feature model created in this depicts various outlines as feature model as visual model, collaboration diagram view of model, its configuration, FeatureIDE Statistics. Basically the paper shows the concept how SPLs can be viewed as feature diagrams using various tools in order to deal with them. This modelling has been widely used by software product line communities and a number of extensions have been proposed.

Indexterms: featureide, feature model, mobilemedia, SPLS.

GJCST-C Classification : D.2

Mobilemedia SPL Creation by FeatureIDE using FODA

Manjinder Kaur [α] & Parveen Kumar [σ]

Abstract- Software Product Lines are used in many areas, combining to form new technologies and products. A product line is a group of products that share a common development platform and vary by the composition and implementation method for the functionalities. This paper describes the implementation or creation of MobileMedia feature model using FODA (Feature Oriented Domain Analysis) methodology using FeatureIDE eclipse plug-in. The feature model created in this depicts various outlines as feature model as visual model, collaboration diagram view of model, its configuration, FeatureIDE Statistics. Basically the paper shows the concept how SPLs can be viewed as feature diagrams using various tools in order to deal with them. This modelling has been widely used by software product line communities and a number of extensions have been proposed.

Indexterms: featureide, feature model, mobilemedia, SPLS.

I. Introduction

With the advancement in the software engineering, many new concepts have been introduced and changing drastically. Software Product Lines (SPLs), an innovative approach in the Software Engineering has changed many things in the industrial area. A product derived from a software product line consists of various components selected from existing component libraries; these components communicate with a common platform to perform specific functionalities.

Skilled software engineers use technologies and practices from a variety of fields to improve their productivity in creating software and to improve the quality of the delivered product called as Product Line Engineering (PLE) [6]. SPLs can be viewed as models by using the concept of feature modelling involving various methodologies. In this paper, FODA (Feature Oriented Domain Analysis) methodologies in designing the SPL using eclipse plug-in FeatureIDE.

II. Software Product Lines (SPLS)

A Software Product Lines are defined as a family of different products which shares same set of core assets or it can be said, a product line consists of multiple systems, which have same architecture and share common core assets with variability among systems. A core asset includes shared components, framework or infrastructure, tools, process, documentation, test cases as these are reused.

Basically, SPL is a family of products designed to take advantage of their common aspects and predicted variability in order to improve quality, delivery time and reduction in cost. Product line engineering (PLE) helps to design, develop, deliver, and evolve a portfolio of common products, with feature variations and functions, through which each stage of the systems and the software development lifecycle from requirements to design, development and testing. Many methods and practices are introduced that is Software Reuse, Component-Based Development and Product Line Engineering (or Product Family Engineering). Enhance the efficiency of SW development when multiple products are to be developed simultaneously:

- Higher productivity
- Higher quality
- Faster time to market
- Lower labour needs

Software Product Line Methods (SPLMs) are the software development approaches in which a set of software systems share a common set of feature produced from a set of reused core assets. Core assets are software artifacts that are re-used in the production of customized products in a software product line (SPL). The assets include the requirements, architecture, components, modelling and analysis, plans, etc. A SPL product can be quickly assembled from core assets, and hence it achieves manufacturing efficiency. SPLM supports "producing goods and services to meet individual customer's needs with near mass production efficiency". The following figure depicts the overview of Software Product Line Methods:

Author α: Student, Department of Computer Science & Engineering, Lovely Professional University, Phagwara, Punjab, India.
e-mail: maenj_chauhan@hotmail.com
Author σ: Assistant Professor, Department of Computer Science & Engineering, Lovely Professional University, Phagwara, Punjab, India.
e-mail: parveen.it@gmail.com

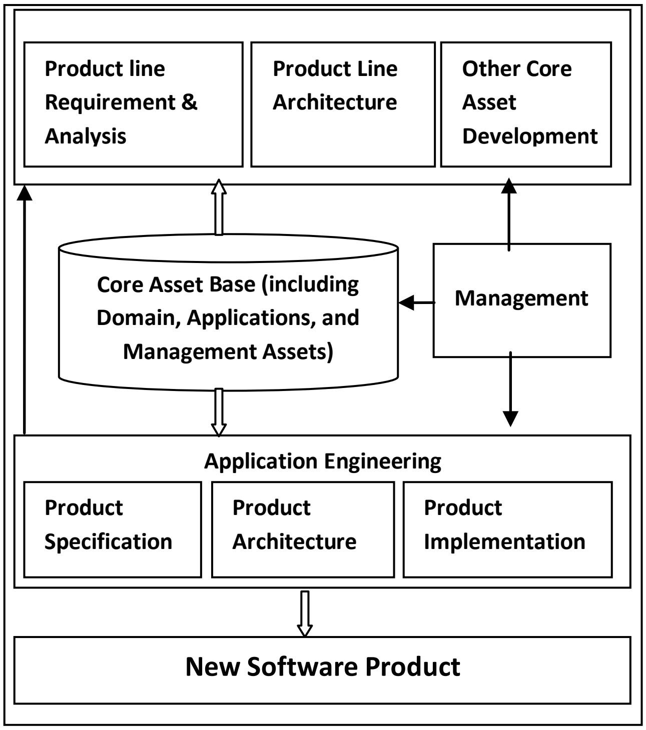

Product line Requirement & Analysis
Product Line Architecture
Other Core Asset Development
Core Asset Base (including Domain, Applications, and Management Assets)
Management
Application Engineering
Product Specification
Product Architecture
Product Implementation
New Software Product

Figure 1: Overview of SPL Methods

III. Feature Model by foda Methodology

Feature model, a representation of products of SPL in a way to express features. Feature models were first introduced in FODA (Feature-Oriented Domain Analysis) method by Kang in 1990 and since then this modeling has been widely used by software product line communities and a number of extensions have been proposed. A feature model is represented by means of feature diagrams. A feature diagram is a graphical or visual notation of a feature model in the form of AND-OR tree, and also various other extensions as feature cloning, feature attributes, collaboration diagrams, configurations etc. This model is basically used as a input to produce other different assets like documents, architecture definition or code. A feature can be defines as a quality or characteristic of software system or system. Therefore, a feature model is a model that defines features and their dependencies depicting in the feature model as cross-tree constraints. A feature configuration, a set of feature describing a member of an SPL and the member will contain a feature if and only if features are in its configuration.

IV. Basic Feature Model Notations

Basic Feature Model is a relationship between a parent feature and its child features categorized as [1]:

- Mandatory – child feature is required.
- Optional – child feature is optional.
- Or – at least one of the sub-features must be selected.
- 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.
- A excludes B – A and B cannot be part of the same product.

V. FODA Methodology

As said earlier, Feature models were first introduced in FODA (Feature-Oriented Domain Analysis) method developed by Software Engineering Institute. It is a domain analysis or product analysis method (analyzing related software systems in a domain to find commonality and variations), which involves conversion as feature model to domain engineering being used in the advance concepts in software engineering and software reuse. Domain Analysis was coined by James Neighbors in 1980s and is the first phase of domain engineering. Therefore it can be said that FODA is a domain analysis technique. The main objective of feature-oriented domain analysis if to create a domain model which represents the family of systems which is then refined into a particular desired system within a domain supporting the functional and architectural reuses [4]. FODA methodology not only identifies the systems in the domain but also the external system interacting with the domain which is known as FODA context analysis. Then further FODA feature analysis from the feature model, configures requirements and the candidate systems by analyzing the end-user's view. Configurable requirements can then be selected from the developed feature model in order to specify the final system and from this view the customer's demands can be satisfied by following this process and achieving the efficiency through technology reuse.

VI. SPL Tools Supporting Feature Models [2]

- FeatureIDE
- Clafer
- Pure::Variants
- Hydra
- S. P. L. O. T. (Software Product Line Online Tools)
- Ahead Tool suite
- Eclipse Modelling Framework Feature Model Project
- FaMa Tool suite
- ToolDay (Tool for Domain Analysis)
- BeTTy Framework
- FAMILIAR

- Feature Model Plug-in
- Gears
- Dopler
- Varmod
- PULSE-BEAT
- FeatureMapper
- MetaEdit+
- Holmes
- FAMA-FW
- MTP (Meta Programming Text processor)
- XVCL
- SPL conqueror
- Velvet
- The guidsl Tool

VII. FeatureIDE Tool for Mobilemedia SPL Creation

FeatureIDE is an Eclipse plug-in or Eclipse based IDE supporting Feature-Oriented Software Development, development of SPLs which involves Domain Analysis, Software Generation and Requirement Analysis. It is the integration of various SPL techniques like Feature-Oriented programming (FOP) (integration of AHEAD, FeatureC++, FeatureHouse), Delta programmming (DOP) including DeltaJ, Aspect-oriented Programming (AOP) including AspectJ and preprocessors (includes annotations of Colligens, Munge, Antenna and Type Chef) [1]. This tool is still under development that is involving new implementations in it. Feature IDE provides a way to implement or create following feature:

- A feature model editor, graphical and text-based.
- Constraint editor with syntax and semantic checking like dead feature detection.
- Configuration Editor for creation and editing of features.
- SPL source code abstraction.
- Supports refactoring, generalizations etc.
- Statistics display of FeatureIDE project.
- Outline view of feature model.
- Collaboration Diagram view.
- Debugging and Lay outing the feature models or manyother new implementations are under development.

VIII. Mobilemedia Feature Model

MobileMedia manipulates media on mobile phones that is music, photo and video. It is basically a family of multimedia management application for mobile devices and is used in research community of SPLs. In this paper, MobileMedia uses 31 features that are Mobile Selection, Mobile Management, Gallery, Internet and Games, which further involve more features as shown in the figure 2. This representation in the figure is called a feature model diagram. A node represents a feature and edges represent dependency between the features. Feature-model diagram also represents constraints on the selected feature when product is build. A white circle (hollow) indicates the optional feature and the black circle (solid) indicates the mandatory feature.

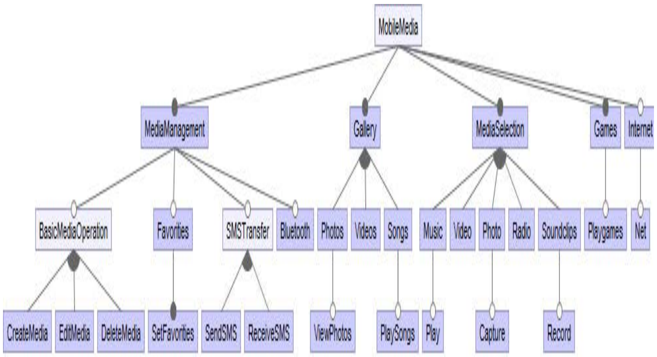


Figure 2 : A Feature Model of Mobile media

In the figure, or-relation has also been used as an arc among the children's from which at least one features from them is selected if their parent is selected. In Feature-Oriented Programming, a feature is implemented as an independent feature module. When a feature is selected, then the corresponding module is compiled together with the other feature modules which have been selected. A group of optional features may interact with another group also mandatory one does can also interact. This feature model involves Abstract and concrete features involving various mandatory and optional feature with AND, OR features. Mobile Media, Basic Media Operation and SMS Transfer are the abstract features and remaining the concrete features and terminal features.

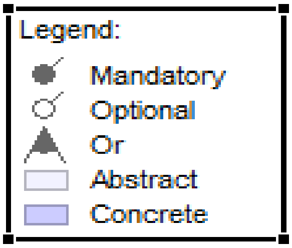
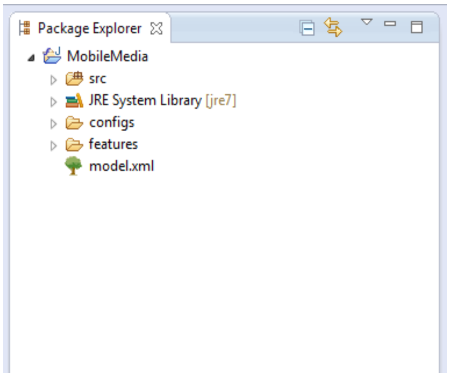


Figure 3 : Legend

Figure 3 displays the Legend shown in the Feature IDE when a feature model diagram is implemented and it shows side by side the symbols with the feature relation being used in the diagram.

a) *Creation of Feature Model*

First of all a feature project is created by selecting Feature IDE Project and choosing the composer as AHEAD [7]. Then project name is mentioned. When Feature IDE project will be created it will be displayed on the left side pane of window in the Package Explorer like shown in the figure and model with root and base will be displayed in the Feature Diagram IDE.

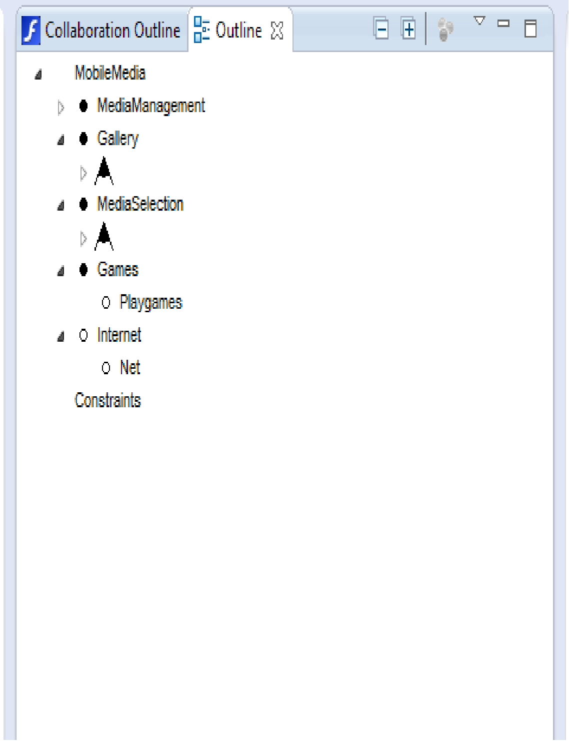


Package Explorer
- MobileMedia
  - src
  - JRE System Library [jre7]
  - configs
  - features
  - model.xml

Figure 4 : MobileMedia FeatureIDE Project

b) *Outline View of the Feature Model*

Outline view displays the overall outline of the feature model that in a brief what it involves like what are the mandatory features, optional features and constraints.



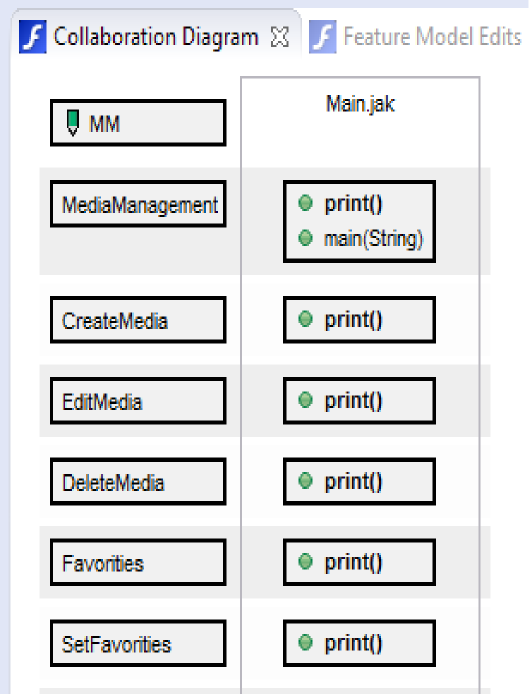
```
MobileMedia
  ● MediaManagement
  ● Gallery
  ● MediaSelection
  ● Games
      ○ Playgames
  ○ Internet
      ○ Net
Constraints
```

Figure 5 : Outline View

Solid circles represent the mandatory features and the hollow circles represent the optional features and another symbol represents the or-relation between the features involving it.

c) *Collaboration Diagram of Featureide Project*

Collaboration Diagram shows the diagram of the feature model as what is the main class being involved and in what feature it is present and shows what the other class features that refines the main feature.



	Main.jak
MM	

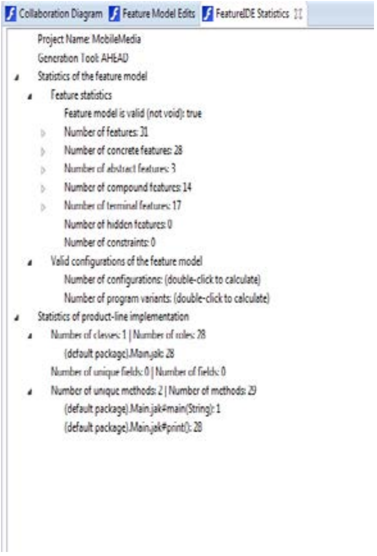
MediaManagement	print()
	main(String)

CreateMedia	print()
EditMedia	print()
DeleteMedia	print()
Favorities	print()
SetFavorities	print()

Figure 6 : Collaboration Diagram

Basically it shows the Main Jak file and the functions involved in the other class features as print ( ) method.

d) *Featureide Statistics of the Project*



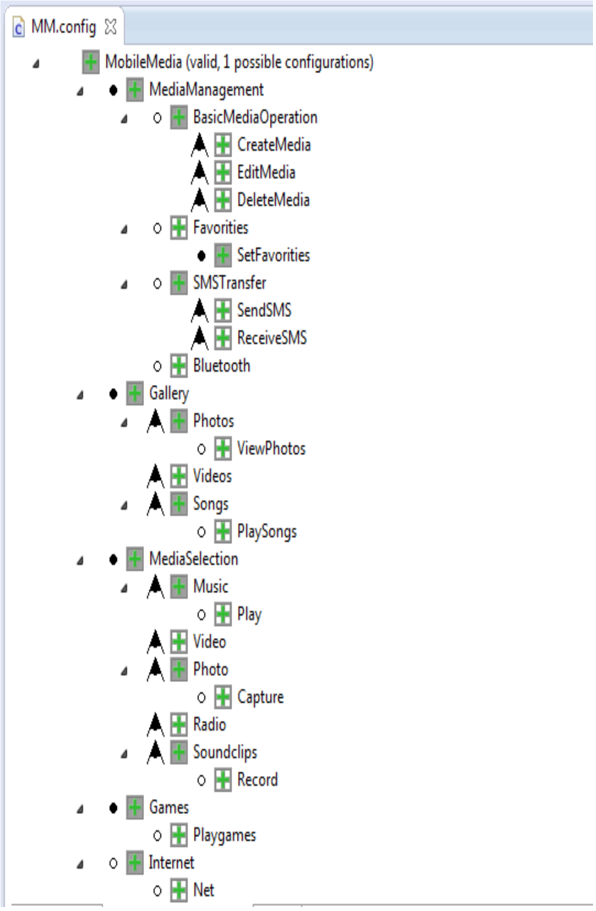
```
Project Name: MobileMedia
Generation Tool: AHEAD
Statistics of the feature model
    Feature statistics
        Feature model is valid (not void): true
        Number of features: 31
        Number of concrete features: 28
        Number of abstract features: 3
        Number of compound features: 14
        Number of terminal features: 17
        Number of hidden features: 0
        Number of constraints: 0
    Valid configurations of the feature model
        Number of configurations: (double-click to calculate)
        Number of program variants: (double-click to calculate)
Statistics of product-line implementation
    Number of classes: 1 | Number of roles: 28
        (default package).Main.jak: 28
    Number of unique fields: 0 | Number of fields: 0
    Number of unique methods: 2 | Number of methods: 29
        (default package).Main.jak#main(String): 1
        (default package).Main.jak#print(): 28
```

Figure 7 : Feature IDE Statistics

This statistics depicts the composer or generation tool AHEAD (Algebraic Hierarchical Equations for Application Design) being used and the statistics involved under Feature IDE. AHEAD composer supports the composition of Jak files where Jak extends the Java with Keywords for Feature-Oriented Programming. In the Mobile Media SPL, feature statistics explain the number of Abstract feature, Concrete features, Compound features, Terminal features, Hidden features and the Constraints involved in it. Other statistics involves the product-line implementations which include the unique methods, classes and unique fields.

e) Configuration File Creation

Configuration view displays the number of configurations the user wants to involve in its SPL feature model. It may the main single configuration involving many configurations.

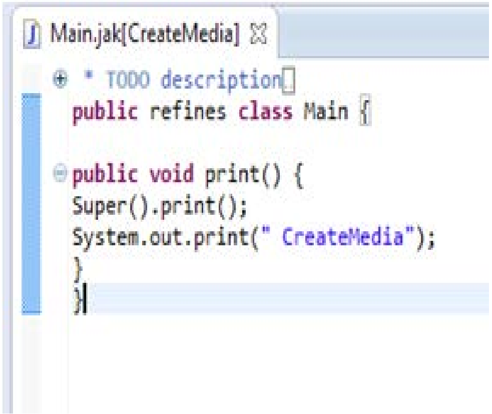


```
MobileMedia (valid, 1 possible configurations)
    MediaManagement
        BasicMediaOperation
            CreateMedia
            EditMedia
            DeleteMedia
        Favorities
            SetFavorities
        SMSTransfer
            SendSMS
            ReceiveSMS
        Bluetooth
    Gallery
        Photos
            ViewPhotos
        Videos
        Songs
            PlaySongs
    MediaSelection
        Music
            Play
        Video
        Photo
            Capture
        Radio
        Soundclips
            Record
    Games
        Playgames
    Internet
        Net
```

Figure 8 : Configuration File

Here Mobile Media involves the main configuration and further its configuration that is feature under it which are refined by the main class involved in the Mobile Media. Refine is the keyword used in AHEAD composer of Feature IDE is used to specify the refinements of an existing class. In the following it has been shown how a class refines the main class.

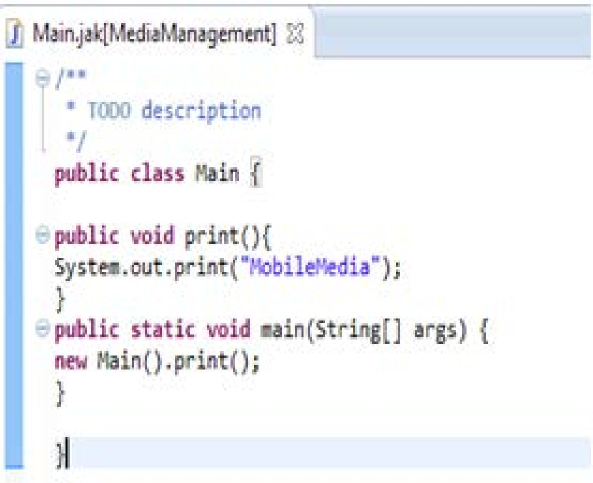
The following figure displays the Jak file creation of a single feature which uses the refine to use the main class.



```
 * TODO description
public refines class Main {

public void print() {
Super().print();
System.out.print(" CreateMedia");
}
}
```

Figure 9 : Source Code Showing a how a Class Main Jak is Refined by another Lower Level Class.

The main class shows the source like this file when created for the first time.

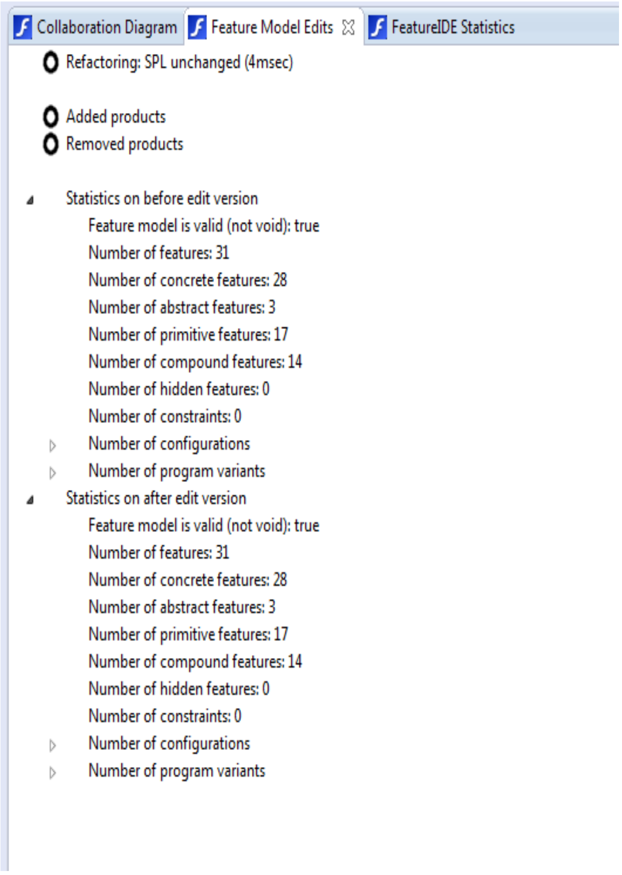


```
/**
 * TODO description
 */
public class Main {

public void print(){
System.out.print("MobileMedia");
}
public static void main(String[] args) {
new Main().print();
}

}
```

Figure 10 : Code Showing how a Main Jak File is Implemented.

f) Feature Model Edits

Feature Model Edit pane shows how much editing has been done to the features and the products involved in the feature model. The number of added and removed products is represented in this edit, the overall how much refactoring has been done to the SPL. In this figure, refactoring has been done to the Mobile Media SPL that before editing how many features were involved in the model and after refactoring how much features has been altered in the model.



```
Refactoring: SPL unchanged (4msec)

Added products
Removed products

Statistics on before edit version
    Feature model is valid (not void): true
    Number of features: 31
    Number of concrete features: 28
    Number of abstract features: 3
    Number of primitive features: 17
    Number of compound features: 14
    Number of hidden features: 0
    Number of constraints: 0
    Number of configurations
    Number of program variants
Statistics on after edit version
    Feature model is valid (not void): true
    Number of features: 31
    Number of concrete features: 28
    Number of abstract features: 3
    Number of primitive features: 17
    Number of compound features: 14
    Number of hidden features: 0
    Number of constraints: 0
    Number of configurations
    Number of program variants
```

Figure 11 : Feature Model Edit

IX. Conclusion

In this paper, SPLs has been discussed as how they can be presented as graphically. Number of tools has been introduced in the market to work upon it as PLE is the new trend in the software engineering. “Engineering” in product lines means the activities are taken into account involved in planning, producing, delivering, and deploying, and retiring products etc. SPLs are under research in various areas. A feature model of Mobile Media that is SPL has been proposed in this paper which is providing a way to represent software product lines in graphical manner in the Feature IDE tool as eclipse plug-in using the FODA methodology. Also defining how it works and shows various outputs of Mobile Media SPL. The feature model created in this depicts various outlines as feature model as visual model, collaboration diagram view of model, its configuration, Feature IDE Statistics and also how a main jak file is created and other files refine the main jak file.

References Références Referencias

1. Takeyama1 and Shigeru Chiba23 “Implementing Feature Interactions with Generic Feature Modules”.
2. Qaiser, Shahid Muhammad et. al., (2010) “Software Product Line: Survey of Tools”, LiU Electronic Press.
3. Bayer Joachin et. al. (1999) “PuLSE: A Methodology to Develop Software Product Lines”, ACM Press.
4. Klaus Pohl, Günter Böckle, Frank vander Linden, (2005) Software Product Line Engineering, Foundations, Principles, and Techniques, Springer.
5. Mathieu Acher, Roberto E., Rick Rabiser, “A Survey on Teaching of Software Product Lines”, Author manuscript, published in "Eight International Workshop on Variability Modelling of Software-Intensive`Systems (VaMoS'14) (2014)".
6. http://www.productlineengineering.com.
7. http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/.