# Contents

# List of Abbreviations

| | |
|---|---|
| Feature Oriented Domain Analysis | FODA |
| Czarnecki-Eisenecker notation | C.E notation |
| Cardinality-Based Feature notation | C.B notation |
| Graphical User Interface | GUI |

# 1 Introduction

The reuse of software has been a driving force of software engineering methods for a long time. However no one can be sure of future requirements, therefore the risk of developing reusable software is high. Domain Analysis provides one of the solutions to reduce this risk. Feature models, which are part of Domain Analysis methods, are used for describing common and different requirements for software systems as instances of a product line.

Industrial Partner is a company that designs and produces safety components for cars and that particularly develops software for that purpose. Pure:variants, one of Eclipse's plug-in, is used as one of the development platforms in the company.

The task was divided into two phases for thesis work. One phase was to analysis existing feature models and to develop a more precise definition of feature models. Another phase was to offer a new notion system to pure:variants.

This report describes the thesis which is a part of the master education Information Engineering at Jönköping University.

## 1.1 Background

The thesis work is joint project between industrial partner and School of Engineering. The main intension of work is to improve the knowledge management in model-driven development processes based on

(a) Semantic structures for components.

(b) Enhancements for established software engineering processes.

This proposed thesis work is composed of research part and development part. In research phase the work is to focus on improving the feature models by using existing notation systems.

The second phase of the work is to implement the proposed feature model notation system, which is based on the research phase. The purpose of the development is to pursue one of two possible prospects:

(1) Create a plug-in for Eclipse, replacing or complementing the graphical presentation of pure:variants.

(2) Create a stand-alone software, that uses XML-format output of pure:variants, parses it and creates the graphs "offline".

## 1.2 Purpose/Objectives

The purpose of this master as mentioned in the above section has two main phases. The purpose of the research phase is to refine the existing feature model notation systems. The purpose of development phase is to give users a new optional graphical representation platform for rendering feature models, which is suggested in the research phase.

## 1.3 Limitations

In the thesis, the research part is considered on feature diagrams and analysis of existing feature notation systems. The research part is considering only four types of notation systems. The development part is considering the conclusion from research part. The result is standalone software called NotationManager, which has one-way communication from Pure-Variant software.

## 1.4 Thesis outline

In Introduction Section the overview of the thesis work is described. It also describes limitation, scope, purpose/objective of the thesis work. After introduction section there are two main parts;

- Research

It section 2 contains the concepts about "Domain Engineering" and "Feature Model" are described. In section 3 is focused on literature reviews, finding out commonalities between different notation systems. In section 4 conclusions and suggestions of existing notation systems are given.

- Implementation

In section 5 the tools which has supported in implementing the results from Research part, as well as the development processes are described.

In section 6 the conclusions are given and future work is suggested.

# 2 "Theoretical Background"

## 2.1 Domain Engineering

For software is made up by different functional parts, each part can be considered as a domain, such as database systems, synchronization packages, workflow systems, GUI libraries, numerical code libraries, etc. Obviously systems and components within a same domain share lots of characteristics as well as requirements. That means many commonalities can be found among systems within the same domain. Therefore, a company has already developed some software systems in a particular domain. And they are going to develop a new software system in this domain. By reusing some parts of existing systems, the company can produce a new one in shorter time and at lower cost. Domain Engineering is a systematic approach to achieve this goal. [3]

"*Domain Engineering is the activity of collecting, organizing, and storing past experience in building systems or parts of systems in a particular domain in the form of reusable assets (i.e. reusable workproducts), as well as providing an adequate means for reusing these assets (i.e. retrieval, qualification, dissemination, adaptation, assembly, etc.) when building new systems.*"[3]

Domain Engineering is composed by three process components which are independent of the time dimension, *Domain Analysis*, *Domain design* and *Domain implementation*. [3]

### 2.1.1 Domain Analysis

Domain analysis is the process to identifying, collecting, organizing and representing the relevant information gathered by domain experts as *domain models*. Domain Analysis is not only recording information but extending it in a creative way. [2, 3]

Domain Analysis generally involves the following activities: [3]

- *Domain planning, identification, and scoping*: planning of the resources for performing domain analysis, identifying the domain of interest, and defining the scope of the domain;
- *Domain modeling*: developing the domain model.

Table 2-1 gives a more detailed list of Domain Analysis activities.

| **Domain Analysis major process components** | **Domain Analysis activities** |
|---|---|
| *Domain characterization*<br>*(domain planning and scoping)* | *Select domain*<br>Perform business analysis and risk analysis in order to determine which domain meets the business objectives of the organization. |

| | |
|---|---|
| | *Domain description*<br>Define the boundary and the contents of the domain. |
| | *Data source identification*<br>Identify the sources of domain knowledge. |
| | *Inventory preparation*<br>Create inventory of data sources. |
| *Data collection*<br>*(domain modeling)* | *Abstract recovery*<br>Recover abstractions |
| | *Knowledge elicitation*<br>Elicit knowledge from experts |
| | *Literature review* |
| | *Analysis of context and scenarios* |
| *Data analysis*<br>*(domain modeling)* | *Identification of entities, operations, and relationships* |
| | *Modularization*<br>Use some appropriate modeling technique, e.g. object-oriented analysis or function and data decomposition. Identify design decisions. |
| | *Analysis of similarity*<br>Analyze similarities between entities, activities, events, relationships, structures, etc. |
| | *Analysis of variations*<br>Analyze variations between entities, activities, events, relationships, structures, etc. |
| | *Analysis of combinations*<br>Analyze combinations suggesting typical structural or behavioral patterns. |

| | |
|---|---|
| | *Trade-off analysis*<br>Analyze trade-offs that suggest possible decompositions of modules and architectures to satisfy incompatible sets of requirements found in the domain. |
| *Taxonomic classification*<br>*(domain modeling)* | *Clustering*<br>Cluster descriptions. |
| | *Abstraction*<br>Abstract descriptions. |
| | *Classification*<br>Classify descriptions. |
| | *Generalization*<br>Generalize descriptions. |
| | *Vocabulary construction* |
| *Evaluation* | Evaluate the domain model. |

Table 2-1Common Domain Analysis process [3]

### 2.1.2 Domain Design

The second activity within Domain Engineering is Domain Design. Domain Design is an activity of developing an adoptable architecture (Design) for the systems in domain of interest. [1, 3]

*"Abstractly, software architecture involves the description of elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns. In general, a particular system is defined in terms of a collection of components and interactions among these components. Such a system may in turn be used as a (composite) element in a larger system design."* [3]

The architecture patterns are listed as follows:

- Layers pattern
- Pipers and filters pattern
- Blackboard pattern
- Broker pattern
- Middle-view-control pattern
- Microkernel pattern

While describing the real world, usually more than one pattern is used at the same time. And different patterns are used in different view, parts and levels of architecture. The architecture design should not only achieve all important requirements but also leave a large degree of freedom for implementation. [3]

### 2.1.3 Domain implementation

The third activity in Domain Engineering is Domain Implementation. Domain Implementation is an activity of translating the results from previous two and implements them.

## 2.2 Feature models

"*Feature model describes properties distinguishing between common and variable requirements. They structure requirements by generalizing them by concepts. They provide a very flexible means of description. Meanwhile, they are applied in some industrial projects for describing software for multiple uses, like component-based systems, reusable libraries, and e.g.*" [4]

Feature model is the product of Feature Modelling. It is expressed mainly by a feature diagram. And there are some additional information called short semantic description of each feature, rationale for each feature, stakeholders and client programs interested in each feature, examples of systems with a given feature, constraints, default dependency rules, availability sites, binding sites, binding modes, open/closed attributes, and priorities. These definitions are given in Table 2-2.

| | |
|---|---|
| **Feature diagram** | It consists of a set of nodes and edges, which form a tree. The root of a tree represents a concept, and other nodes stand for features. There are also descriptions of both single feature and feature group. In Original FODA notation, which is described in section 3.1.1, feature group type is represented by arcs. |
| **Semantic description** | A short description about feature’ s semantic. It is helpful when the feature is implemented by other models. |
| **Rationale** | Explanation of why a feature is included in the model, and constrains of the feature if it is using in an application. |
| **Stakeholders and client programs:** | ➢ Stakeholders: users, customers, developers, managers, etc.<br>➢ Client programs: the program which needs the feature |

| | |
|---|---|
| **Exemplar systems:** | Existing system which implement the feature. |
| **Constraints and default dependency rules:** | ➢ Constraints record required dependencies between variable features, possibly over multiple feature diagrams.<br>➢ Default dependency rules suggest default values for unspecified parameters based on other parameters. |
| **Availability sites, binding sites, and binding mode** | ➢ Availability site describes when, where, and to whom a variable feature is available.<br>➢ Binding site describes when, where, and by whom a feature may be bound.<br>➢ Binding mode determines whether a feature is statically, changeably, or dynamically bound. |
| **Open/closed attribute** | ➢ Open attribute: new direct variable subfeatures (or features) are expected<br>➢ Closed attribute: no other direct variable subfeatures (or features) are expected. |
| **Priorities** | They are assigned to features in order to record their relevance to the project |

Table 2-2 Feature model concepts [3]

For features represent functionalities of a system, which are needed by customers. Both customers and developers can use the feature model as a communication medium. A customer has to understand the meaning of each feature before using the system. For example in Figure 2-1, it shows a feature model of car.

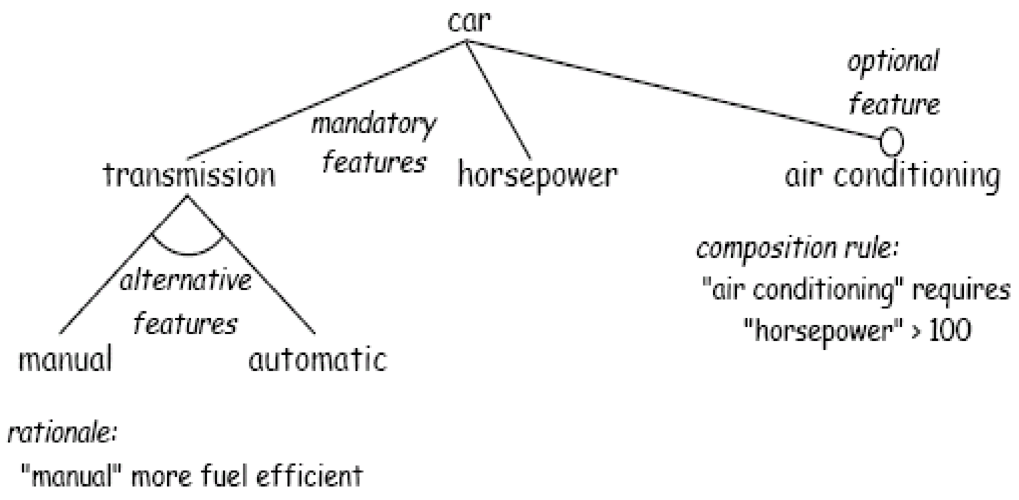


Figure 2-1 Example of a car represented by Original FODA [3]

If a person wants to buy a car, he has to make a choice between manual transmission and automatic transmission, because it is impossible to have both. For the rationale is “manual” more fuel efficient, if the person concerns fuel efficiency he may chose manual transmission. [2] More definitions of feature diagrams are discussed in section 3.

# 3 Research analysis

## 3.1 Literature review

The purpose of this research part of report is to analysis the existing feature model notations. For that, research is to focus on to find out the commonality between different notation systems, to compare the efficiency of the notation systems under consideration and to choose the most efficient one.

### 3.1.1 Notations concepts

There are four accepted sets of notations. First is *Original FODA notation*.

In Figure 2.1, there are three concepts:

- **Mandatory feature**, this feature is chosen if its parent is chosen. It is represented by the text. It is represented by text directly.
- **Optional feature**, this feature may be chosen only if its parent is chosen. In the diagram there is a white circle above the text.
- **Alternative feature group**, there are at least two features which share the same parent. When their parent is chosen, one of the features will be chosen. There is an empty arc under the parent, which includes all the connection between the parent and its alternative features in the diagram.

Second is *Czarnecki-Eisenecker* (C.E) *notation*. It is extended from the Original FODA notations. And it has one more concept besides the three concepts mentioned above.

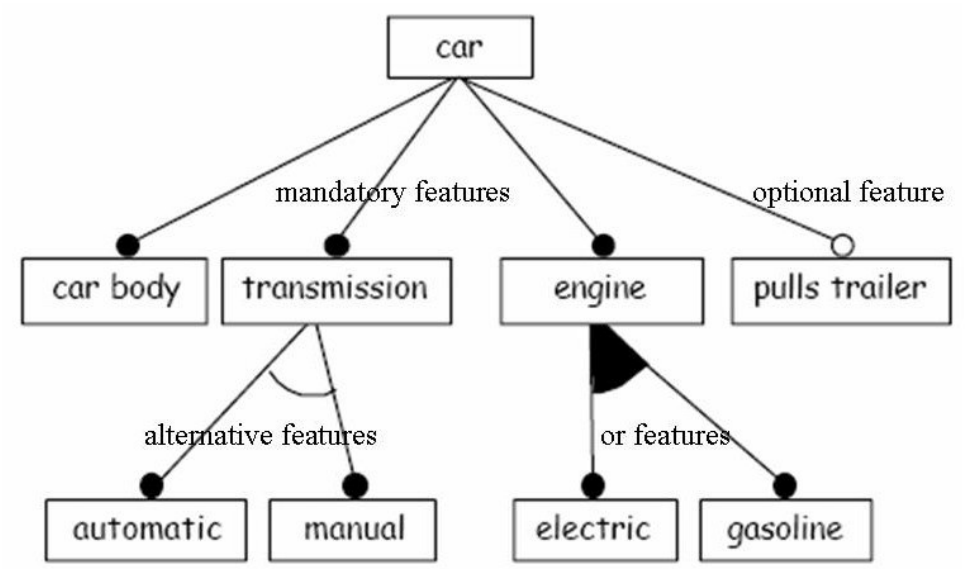


Figure 3-1 Example of a car represented by C.E [3]

In Figure 3-1, there are four concepts:

- **Mandatory feature**, this feature is chosen if its parent is chosen. It is represented by a label with a black circle above.

- **Optional feature**, this feature may be chosen only if its parent is chosen. In the diagram there is a circle above the label.
- **Alternative feature group**, there are at least two features which share the same parent. When their parent is chosen only one will be chosen. There is an empty arc under the parent, which includes all the connection between the parent and its alternative features in the diagram.
- **Or feature group**, there are at least two features which share the same parent. When their parent is chosen more than one can be chosen. There is a filled arc under the parent, which includes all the connection between the parent and its alternative features in the diagram.

Third is *Cardinality-Based Feature* (C.B) *notation*. Or it is called extended C.E notation. This set of notations borrows some concept from UML class diagrams. It complements the expression, which C.E notation is lacked, about feature number. There are no special features called mandatory or optional features. The new concepts in C.B notation are Feature cardinalities and Group cardinalities. In addition, there is one more concept for exploring the diagram, diagram modularization.

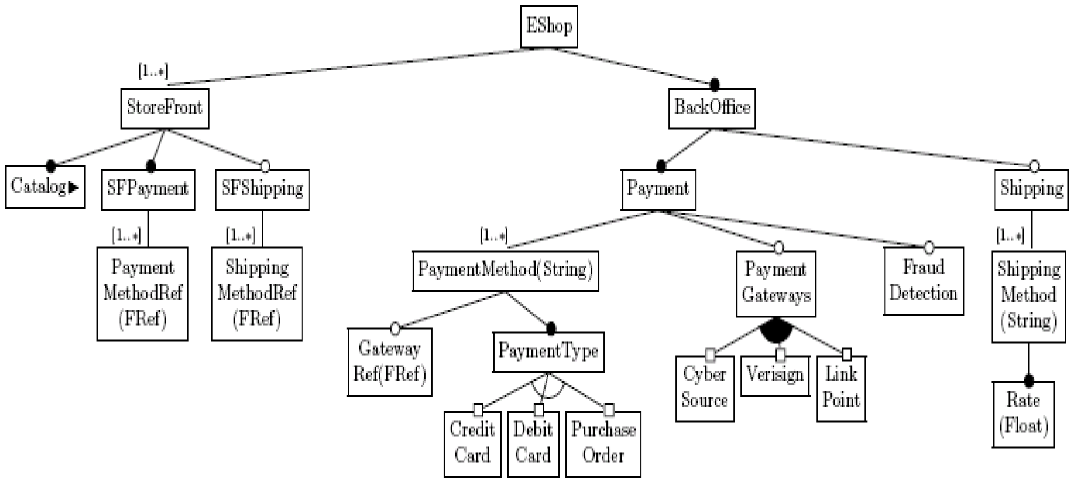


Figure 3-2 Example of an Eshop represented by C.B notation [7]

In Figure 3-2, there are two concepts:

- **Feature cardinalities**, in order to express how many features are there, cardinalities are used, such as [2…*]. Mandatory and optional features are special cases of features with cardinalities [0…1] or [1…1].
- **Group cardinalities**, to express both alternative features and or feature which are defined above. In another word, how many features will be chosen in a feature group is expressed by cardinalities as <m-n>.

Although there are not particular definitions of mandatory and optional features, even for or feature group and alternative feature group. In the latest version of diagram the expressions from C.E are still remained. When the cardinalities is [0…1] or [1…1], features are represented by labels with circles above. It's the same expression as old concept (C.E notation), or feature group and alternative group, using small white squares instead of circles.

Fourth is *FeatuRSEB*, which is FODA used in Reused-Driven Software Engineering Business. This set of notations is based on UML diagrams' elements.[5]

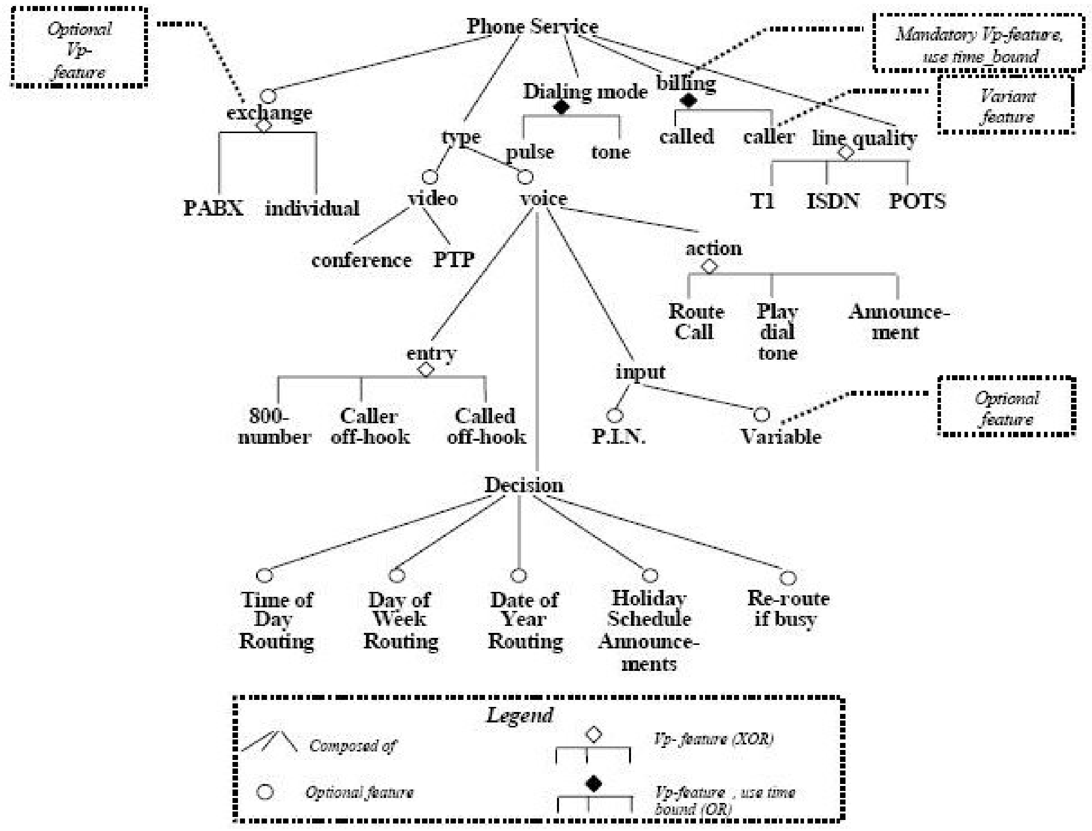


Figure 3-3 Example of phone service [5]

In Figure 3-3, there are four concepts:

- **Mandatory feature**, this feature is chosen if its parent is chosen. It is represented by text directly.
- **Optional feature**, this feature may be chosen only if its parent is chosen. In the diagram there is a white circle above the text.
- **Alternative feature group**, there are at least two features which share the same parent. When their parent is chosen only one will be chosen. A white diamond is used to express the meaning of alternative feature group.
- **Or feature group**, there are at least two features which share the same parent. When their parent is chosen more than one can be chosen. A black diamond is used to express the meaning of alternative feature group.

## 3.2 Comparison among notation systems

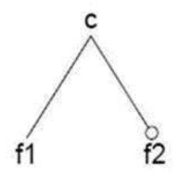
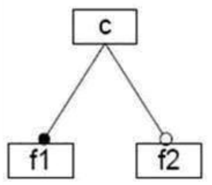
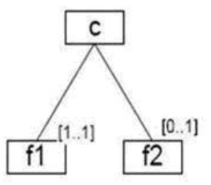
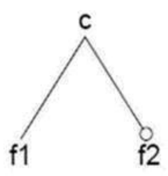
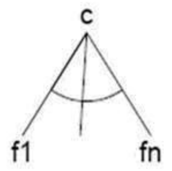
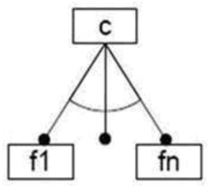
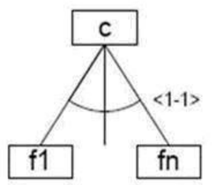
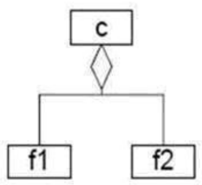
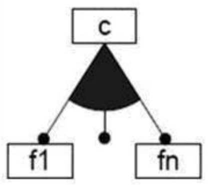
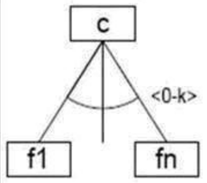
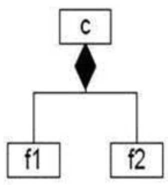
| | Original FODA notation | Czarnecki-Eisenecker notation | Cardinality-Based Feature notation | FeatuRSEB |
|---|---|---|---|---|
| Mandatory and optional subfeature |  |  |  |  |
| Alternative subfeature group |  |  |  |  |
| Or subfeature group | |  |  |  |

Table 3-1 Comparison among notation systems [8]

In Table 3-1, a brief view of the four notation systems is given. It shows how they represent the same information. However, the table does not include all situations. For the definition of features are completely different between cardinality-based notation and others, the table just pick the special cases. It is just for comparison requirement.

The C.E notation is most accepted, because it extended from *Original FODA notation*, at the same time it brings some new possibilities. Both of them use circles and arcs to stand for feature types. As the extension of *Original FODA notation*, C.E notation add black circle to express mandatory feature. And it brings a new concept called "Or subfeature group". As a result, it can represent more information from the true world.

The C.B notation is also called C.E extended notation. It imports cardinality idea in order to increase its flexibility performance. In C.B notation, if the cardinality of feature is [0...1] or [1...1], it is represented still in C.E way. It is the same if the group cardinality is [1...1] or [0...k]. Only if the information can not be represented by C.E way, it will use cardinality. So it has more flexibility than C.E notation while expressing the numbers.

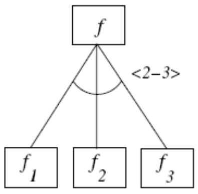


Figure 3-4 Example of group cardinality

For example in Figure 3-4, the feature group {f1, f2, f3}, when f is chosen at least two in the group should be chosen. However, the same way can not be done by C.E notation. As it mentions in section 3.1.1, mandatory and optional feature in C.E notation are special case in C.B notation. In C.B notation, the range of features is unlimited. And there is a *feature diagram references* concept. It helps to connect current diagram with other diagrams.

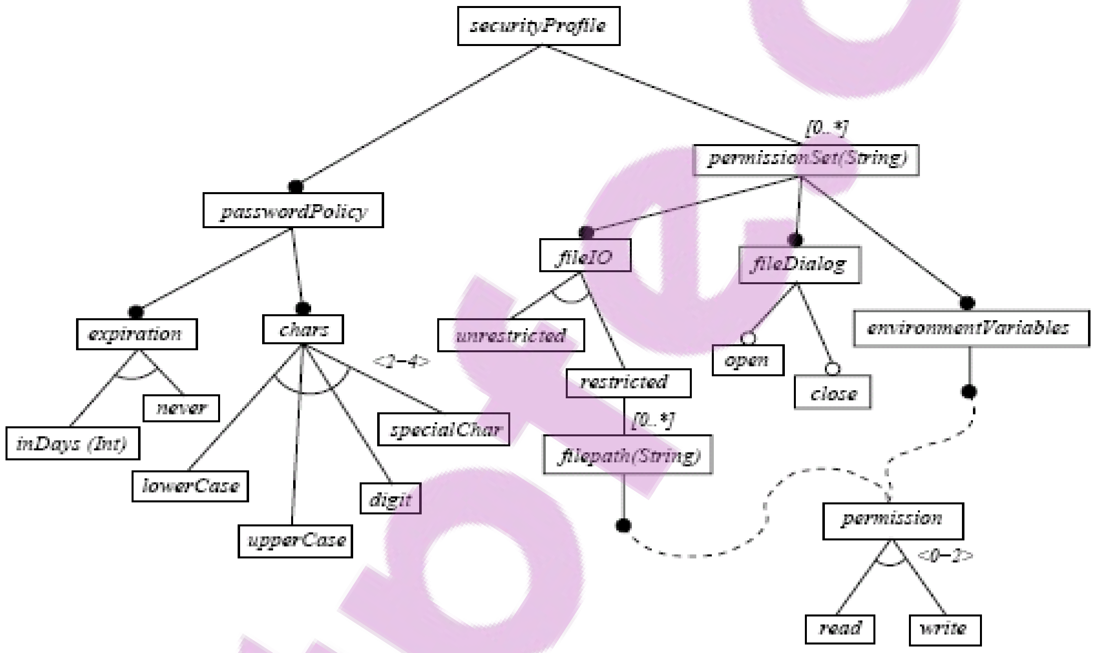


Figure 3-5 Security profile Example [6]

In Figure 3-5, “permission” is the root of another diagram. It is referred by the nodes under both “filepath (String)” and “environmentVariables”. A large diagram can be separated into lots of small independent diagrams. Lots of small diagrams can make up a large diagram too. The relations among different diagrams are clear. As a result the way of expression information becomes flexible.

However, too much figures in a diagram increases the difficulty for readers’ reading. Simple notations such as circles and arcs are easier to recognize.

*FeatuRSEB* gives another graphical performance, it use diamond instead of arc. However, it does not offer more concepts in the feature diagram. So it can be considered the same as C.E notation.

# 4 Results

According to the analysis in section 3, the concepts expressed by the four notation systems are shown in Figure 4-1:

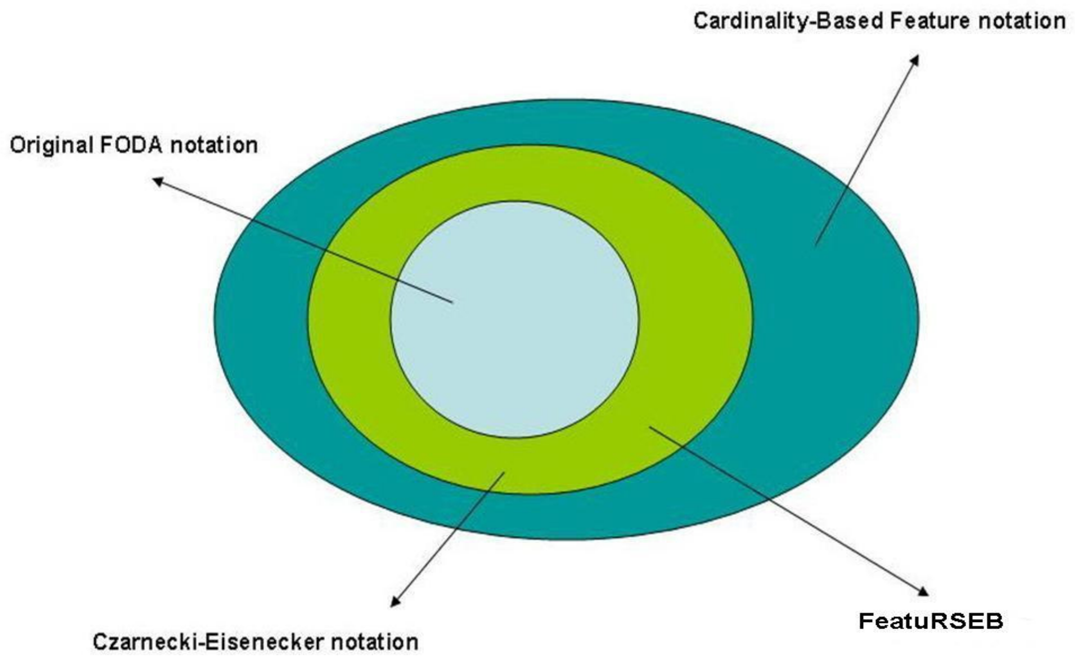

Cardinality-Based Feature notation
Original FODA notation
Czarnecki-Eisenecker notation
FeatuRSEB

Figure 4-1 Relationships of the four notation systems

C.E notation is extended from Original FODA notation, and there are not completely different between them. FeatureRSEB has the same concepts with C.E notation but expresses in another way. C.B notation use cardinality concepts which are totally different from C.E notation. So C.B notation and C.E notation should be analyzed deeper.

Obviously C.B notations can express the most information, however too much information increases the difficulty of reading. As in other areas of software engineering, different models are used in different situations, notation systems also can be used in different situations.

If C.E notation is used in an abstract level, that means mandatory does not simply equal to cardinality [1...1]. Mandatory stands for an abstract concept that when the feature’ s parent is chosen it has to be chosen. For example in a car concept, feature wheel is mandatory. That means a car have to have wheels. It does not mean that a car just have one wheel. C.B notation is used in a detail level. In this case, mandatory concept in C.E notation is limited to cardinality [1...1]. And black circles are still kept to represent the mandatory concept. So the new relationships of the four notation systems are shown in Figure 4-2.

In addition, the *feature diagram references* concept in C.B notation can be used in C.E notation. As a result, C.E notation diagrams can be connected as well.

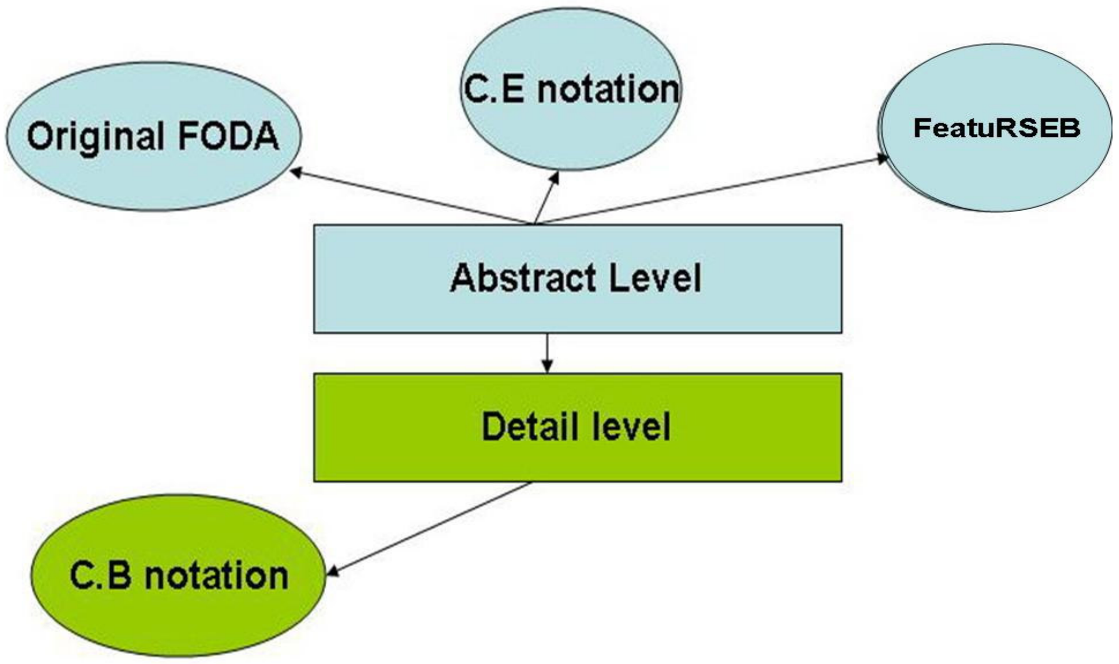


Figure 4-2 New Relationships of the four notation systems

All notation systems are interested in the relation between features and subfeatures. However in some areas constrains between features are also interesting. In Figure 2-1, “air conditioning” requires “horsepower” > 100. It is a example of constrains. The authors have suggested two methods for expressions.

➢ To write the constrain under the feature, shown in Figure 4-3:

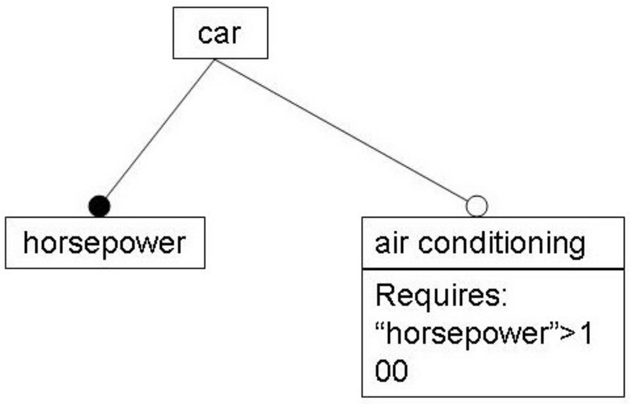


Figure 4-3 Constrain expression method 1

➢ To draw a arrow to the related feature, and to write the constrain up the arrow, shown in Figure 4-4

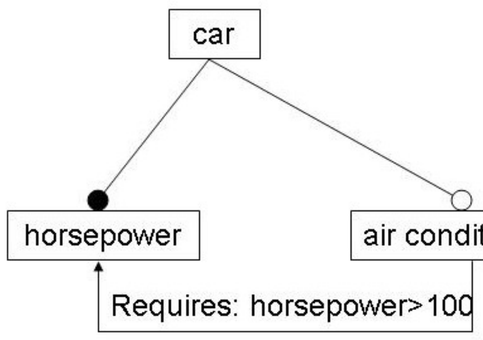


Figure 4-4 Constrain expression method 2

Users can not have a direct view of constrain relations in method 1, because readers have to match the name of feature by themselves. By method 2, arrows help readers to connect features. However in a complex example, too many arrows may overlap with each other. Then it is tough to be recognized.

In conclusion, feature diagrams are not responsible for representing all information from feature models. Users can make choices according to requirements. In abstract level, C.E notation plus *feature diagram references* is enough. In detail level, C.B notation is fine. And users can chose one of the methods based on situations to express constrains between features.

# 5 Resulted implementation tool

## 5.1 Implementation supported tools

### 5.1.1 Eclipse

Eclipse.org is an open source community. Eclipse community provides the software developers an open-neutral development platform and application frameworks for software development. The community is not only facilitating creation, evolution, promotion and supporting Open Source Eclipse Platform but also cultivating both an open source community and an ecosystem of complementary products, capabilities, and services [9].

Eclipse provides plug-in based framework environment to the developers. This plug-in based framework environment makes easy for developer to create, integrate and utilize software tool. This plug-in based environment helps saving time and money for the software developers [9].

Platform for Eclipse is written in Java language and it comes with wide range of plug-ins construction toolkits and examples. Eclipse platform is already deployed on a wide range of workstations including Linux, Solaris, HP-U, OS X and Windows based systems [9].

Currently Eclipse.org community has launched Eclipse SDK 3.2.1 for windows platform development which requires JRE (Java Runtime Environment) version 1.5.0 for support.Eclipse open source platform is providing different versions of SDKs (Standard Development Kits) for product feature variant management to the developers and software application vendors. For feature model rendering Eclipse has many plug-ins. Pure-Variant is one these plug-ins.

Eclipse open source platform for development can be integrated with different plug-ins to develop different software applications. The proposed thesis work is focused on one of the plug-ins named “Pure-Variant”. Pure-variant user interface (UI) is completely based on Eclipse. The following section will provide some information about Pure-Variant.

### 5.1.2 Pure-Variant

Products which are closely related to each other have mostly identical code, with only few differences which specify the unique functionality. When talking about product line approach all product parts are divided into commonalities and variability. In re-engineering approach, for efficient use it is of important concern to store, manage common and variable group of products [10].

Variant management is used when joined software development for a software product line is required. Pure-Variant is a tool used for variant management of product line based (group of similar products). Pure-Variant is used to outline and manage all software products along with the components, constraints and term of usage. The information provided by Pure-Variant and supported tool in entire software configuration process efficient and valid solutions are automatically created from the chosen features [10].

Pure-Variant has given new dimensions for development of custom-made software solutions. Pure-Variant integrates seamlessly into existing development processes and it is independent of the programming language. Due to these two characteristics Pure-Variant is really easy to start with [10].

The following Figure 5-1 gives the overview of the models supported in Pure-Variant.

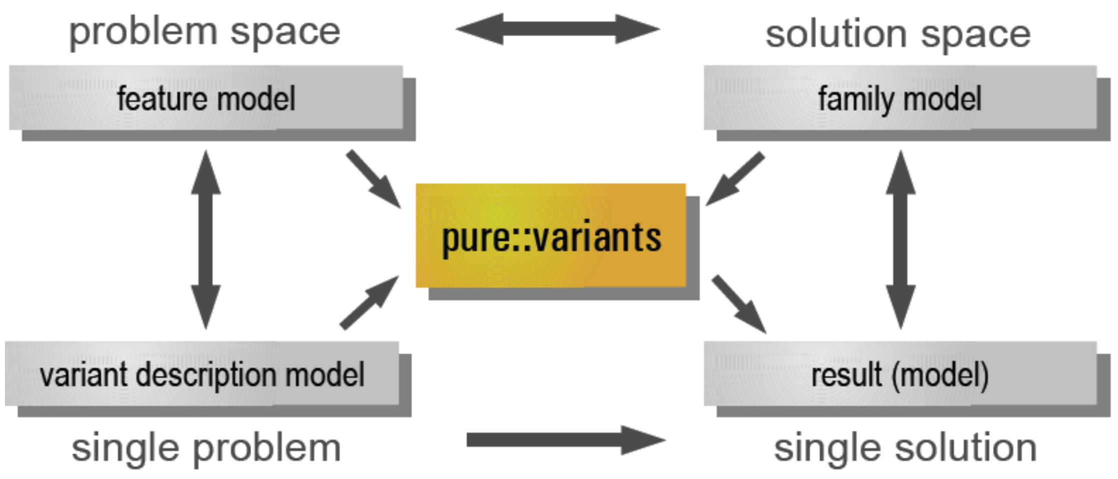


Figure 5-1 “Models in Pure-Variant and variant management” [11]

There are few basic concepts regarding Pure-Variant. This section will discuss about these basic concepts in Pure-Variant.

- Pure-Variant Element

Pure-Variant has different types of elements. These all types of elements belong to same element class. Every element has a type. Pure-Variant has following types of elements;

Ps: feature, ps: component, ps: part, ps: source.

Every element has standard information like;

ID, Unique Name, Visible Name, Description [11].

- Pure-Variant Element Relations

Pure-Variant elements have relations “1: N”. Source element has the information about the relations. In Pure-Variant the user has the option to add his/her relations and add description. He/she can also define the relation restrictions [11].

- Pure-Variant Element attributes

The Pure-Variant element attribute has main characteristics, it has name, has type, may also has some restrictions on it. Pure-Variant element may have any number of attributes. Attribute values may be fixed or variable/calculations [11].

All elements that are selected must have valid attribute value. Attributes which don't have any value have default value [11].

The following Table 5-1 will discuss about some properties of Pure-Variant. These include Modeling schemes, basic structures, limitations/restrictions of Pure-Variant.

| | |
|---|---|
| **Pure-Variant Modelling Schemes** | In Pure-Variant following types of modelling can be done based on the development requirements.<br>➢ Feature Model<br>➢ Family Model<br>➢ Configuration Space<br>Variant description model |
| **Pure-Variant Models Basic Structure** | There are few restrictions which limits the availability/validity of associated item.<br>➢ Logical expression closely related to OCL notation system is used in Pure-Variant.<br>➢ pvProlog is used for evaluation.<br>➢ Depending on restricted item type precise semantic of restriction varies.<br>➢ Example of restriction is as follows<br>Hasfeature ('A') or not (hasFeature ('B')) [11] |
| **Pure-Variant Limitations/restrictions** | There are few restrictions which limits the availability/validity of associated item.<br>➢ Logical expression closely related to OCL notation system is used in Pure-Variant.<br>➢ pvProlog is used for evaluation.<br>➢ Depending on restricted item type precise semantic of restriction varies.<br>➢ Example of restriction is as follows<br>Hasfeature ('A') or not (hasFeature ('B')) [11] |

Table 5-1 Basic Properties of Pure-Variant

## 5.2 Development focus

The thesis development work is focused on feature model rendering. The following sections will explain some further details about Feature models.

### 5.2.1 Pure-Variant Feature Models

Feature Models is made up of elements of class "ps : feature". Pure-Variant is supporting four types of feature groups. These four groups are as follows

- Ps : mandatory [n]
- Ps : optional [0-n]
- Ps : alternative [1]
- Ps : or [1-n]*

Feature in Pure-Variant feature model at most have only one children group of each group type [11].

Feature models gives easy understanding of product features to the users and also represents relationship dependencies between them. The following UI (user interface) will give the information about feature variant management on Eclipse platform. The following UI will also gives the information about necessary views and operations for editing which are used for efficient feature model handling [11].

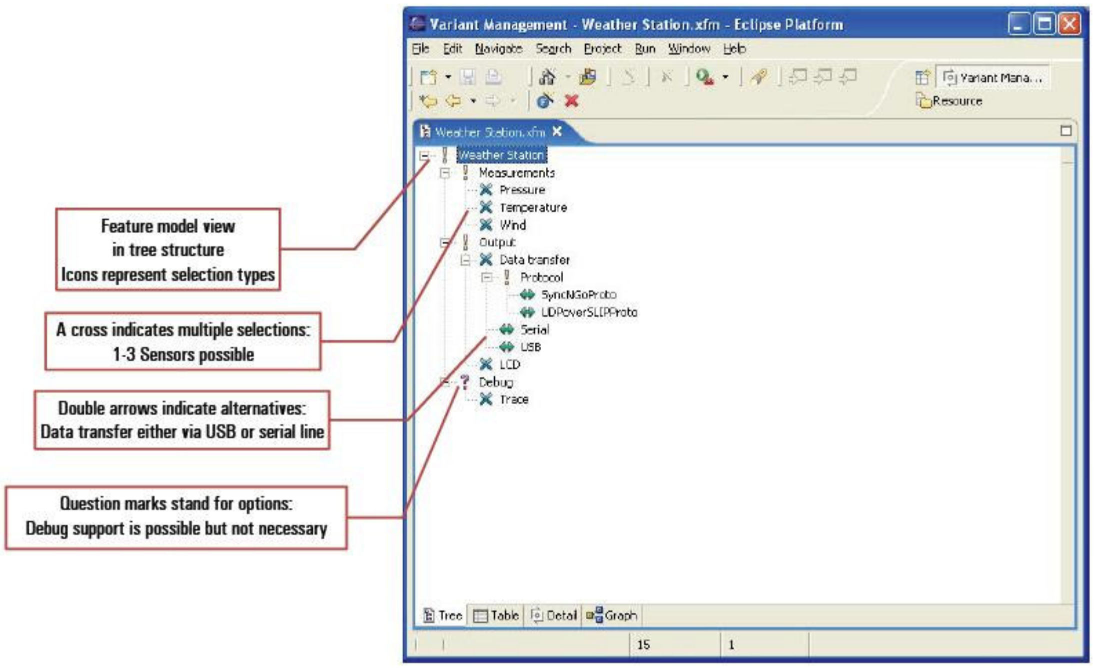

Variant Management - Weather Station.xfm - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Variant Mana...
Resource
Weather Station.xfm
Weather Station
Measurements
Pressure
Temperature
Wind
Output
Data transfer
Protocol
SyncNGoProto
LDPoverSLIPProto
Serial
USB
LCD
Debug
Trace
Tree
Table
Detail
Graph
15
1
Feature model view in tree structure Icons represent selection types
A cross indicates multiple selections: 1-3 Sensors possible
Double arrows indicate alternatives: Data transfer either via USB or serial line
Question marks stand for options: Debug support is possible but not necessary

Figure 5-2 "Feature Model tree structure view in Eclipse Platform" [11]

Figure 5-2 shows the graphical view of the feature model tree structure in Eclipse.

The following UI (user interface) shows graphical representation of the feature model tree structure. Figure 5-3 shows Pure-Variant Feature Model View.

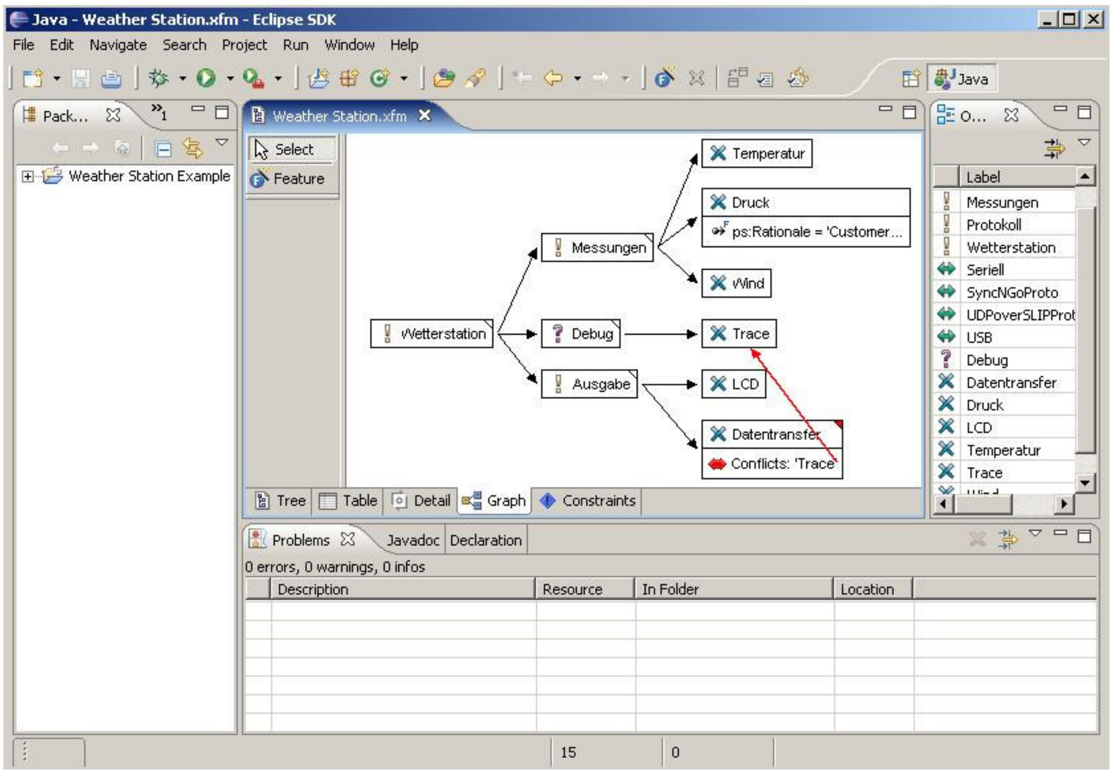


Figure 5-3 “Feature Model Graphical View on Eclipse Platform”

## 5.3 Implementation Proceedures

This section will present the development results.Scope of the implementated results. The proceedures followed for developing and implementing the results from the research section of the thesis report.

### 5.3.1 Scope of Implementated Results

This section of report will give information regarding the scope. The implementation results are divided in to two groups. While the development is done in modules so there are two major modules. The following sections will give the scope of each of these two implemented modules.

#### 5.3.1.1 Front End(GUI) Development result Scope/Limitations

Front End(GUI) implementation results has some limitations, it has certian scope. Front End development is done in Java NetBeans 5.5.The following are limitations of the front end development results.

- It can demonstrate two types of notation systems to user.
- User can drag the features using mouse pointer.
- User can show/hide feature. User can also show/hide feature children.
- It can save the resulted feature model graphical view as JPEG file format.
- User cannot insert, delete, update, modify, feature values.

#### 5.3.1.2 Back End Development Scope result Scope/Limitations

Back End developed results also have scope/limitations. Back End development is done using Java NetBeans 5.5, and understanding of XML file structures and implemention. The followings are limitations of back end developed results.

- Pure-Variant feature model XML files which are exported to the disk can only be read.
- Result can only read Pure-Variant feature model XML files. But vice versa is not possible.
- Result can only save XML file as developer defined XML structure.
- Only feature model related information can be extracted from the Pure-Variant feature model XML file. Which is then used by Front End(GUI) to demonstrate feature Model on layout.

### 5.3.2 Development Process

The following section will discuss the development process of both Front End(GUI) and Back End Results. The problems developers faced and options they selected for getting the results.

#### 5.3.2.1 Front End(GUI) Development Process

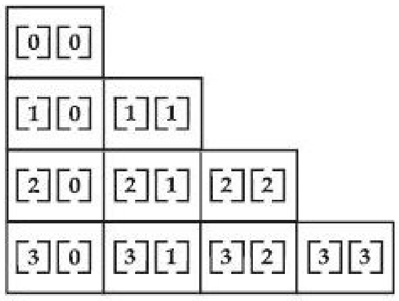
During development of Front End there are two very important points to consider. The following section will discuss these two problems. The choices that developer made and comparisons between them.

**Problem 1**

| | |
|---|---|
| Description | How to represent features? |
| Options | 1. use JLabel<br>2. use rectangle string |
| Comparision | JLabel’s attributes are perfectly suitble for representing features, because it’s bound can change automaticly with it’s text length. The position of text can be center,left or right. And there are lots of types of JLabel boundary. However to rerange JLabel’s position by mouse is not evry each. And the way of attaching a circle to a JLabel is unknown.<br>Rectangle string means to draw a string to the pane and then to draw a rectangel around it. In this way, the rerangement is simple, however how to make the rectangle suit the string is a problem. And the type of boundary can’t have options. |
| Selected option | For rerangement is the main requriement, the second method is chosed. The solution of making the rectangle to suit the string was found. The solution of combing JLabels and circles |

| | |
|---|---|
| | has not been found yet. |

**Problem 2**

| | |
|---|---|
| Description | How to do the layout of features? |
| Requriments | 1. parent should be in the middle of it’ s children.<br>2. each feautre should not be overlaped with others. |
| Comparision | It’ s very difficult to achieve both requriments. The input feature model is unkonwn, as a result how many features in some level is uncertain. That means if a feature is drawn, the feature next to it can not simplily be drawn next to it. Because if both of the features have children, some of their children will be overlap.<br>If we just consider the second requriment, to use two-dimension group is a good sulotion. The features in the same level just need to be drawn one by one. It looks like follow,<br> [0][0]<br>[1][0] [1][1]<br>[2][0] [2][1] [2][2]<br>[3][0] [3][1] [3][2] [3][3] |
| Solution | Finally the process is devided into two step. First go from top to bottom. The goal is to achieve requriment 2. Before a feature is drawn calculate its previous feature, previous feature’ s children and itself’ s children to make sure its children and its privous feature’ s children won’ t be overlaped. Then go from bottom to top. The goal is to achieve requriment 1. If one feature is not in its children’ s center, shift it to the center and remember the shifting distance. Then shift the features behind it with the same distance. |

### 5.3.2.2 Back End Development Process

During developing Back End results. The developer faced three main modules to develop. Development of these three modules will be discuesed in the following sections.

- **Reading of PureVariant XML file module**

This section of the report will discuss the first module for developing Back End results.

**Problem 1**

| | |
|---|---|
| **Description**: | Understanding differences between ordinary XML file structure and PureVariant XML file structure. |
| **Options:** | Reading/understanding Pure-Variant XML file structure. |
| **Comparison:** | none. |
| **Developer Choice:** | The developers selected the material and literature to understand the basic structure elements from Pure-Variant official website. |

**Problem 2**

| | |
|---|---|
| **Description**: | How to Extract feature model development related information from Pure-Variant exported XML file? |
| **Options:** | ➢ Extract all information from PureVariant exported XML file. Then select feature model related information.<br>➢ Understand feature model related information from Pure-Variant exported XML file and indetify XML tags to get information from them. |
| **Comparison:** | Extracting all information from Pure-Variant XML file is not good idea. It will decrease the effeciency of the reading module, and also will take more time. An other option is to select the tags which gives feature modeling rendering information. It will increase the effecincy of the reading module, and also save time. |
| **Developer Choice:** | Developer choosed second option as it is effecienct and less time consuming. |

**Problem 3**

| | |
|---|---|
| **Description**: | How to Save feature model information extracted from XML file. And use it in different classes? |
| **Options:** | ➢ Save extracted information in class objects. And pass it between different classes.<br>➢ Save extracted information in class objects. And used hash maps to store the objects of the classes and then pass the information between classes. |

| | |
|---|---|
| **Comparison:** | Saving information in class objects is a good options. But when it comes to passing them it gives some difficulties to pass them. Second option is to use the same first option but save the objects in to hash maps (which stores the class objects and unique identifier to identify each object in hash map). Dispite of sending many objects between classes its better to store them in hash map and send it between classes for information exchange. It will increase the effeciency, decrease the resouce utilization and also decrease the information exchange traffic between classes. |
| **Developer Choice:** | Developer choosed second option due to more efficiency and less resource utilization. |

**Problem 4**

| | |
|---|---|
| **Description**: | How to save the hashmap objects identifiers? |
| **Options:** | ➢ Use static arrays.<br>➢ Use vectors. |
| **Comparison:** | For working on hashmaps which contains class objects and unique identifiers for every class object. Static arrays for saving unique identifiers, which has fixed size is not a good option. As the developer don’ t know about how many objects will be stored in the hashmap. So vectors (dynamic arrays) are used to store object unique identifiers and then used to manipulate the objects. |
| **Developer Choice:** | Using vectors is effecient as compared to static arrays. So second option is appropriate to use. |

**Problem 5**

| | |
|---|---|
| **Description**: | which structure is use to save feature children information? |
| **Options:** | ➢ Static arrays<br>➢ Vectors (Dynamic arrays). |
| **Comparison:** | Each feature in feature model has children. It may be 1 or more. As developer doesn’ t know how many children can a feature has. So one option is to use static arrays which is not effecient. Other option is to use vectors which is effecient in this situation. As it expand dynamically. |
| **Developer** | Using vectors is effecient as compared to static arrays in this |

| | |
|---|---|
| **Choice:** | situation. So second option is appropriate to use. |

- **Writing/Saving feature model related information in XML file**

**Problem 1**

| | |
|---|---|
| **Description**: | which XML structure should be to save the extracted XML file information? |
| **Options:** | ➢ Use Pure-Variant XML file structure for saving/writing extracted information<br>➢ Use self defined XML file structure for saving/writing extracted information |
| **Comparison:** | First option is appropiate when both communication; communication(reading) of Pure-Variant XML file from developed result, vice versa. As the requirnment was to read XML file and to render it on layout. So for that purpose self structured is used which is easily interpratable by developed source code. Second choice is also appropriate as it gives easy understanding to the developer about what information is in the XML structure. And in future if some changes are to be made then developer can easily check and understand it. |
| **Developer Choice:** | Using self structured XML file is appropriate according to given requirnment. |

- **Reading of Self Structured saved XML file**

**Problem 5**

| | |
|---|---|
| **Description**: | which logic use for reading self structured XML file? |
| **Options:** | ➢ Use same logic for reading Pure-Variant XML file<br>➢ Define own logic for reading self structured XML file |
| **Comparison:** | Using same logic which is used for reading Pure-Variant XML file is not appropriate in this situation. As developer has defined his own structure for saving the extracted XML file information. So defining own logic is better and effecient in this situation. |
| **Developer Choice:** | Develpor has selected second choice due to more effeciency and less resouce utilization. |

## 5.4 Interface Description and working

The Graphical User Interface has following interfaces working of each interface will be discuessed in the following section.

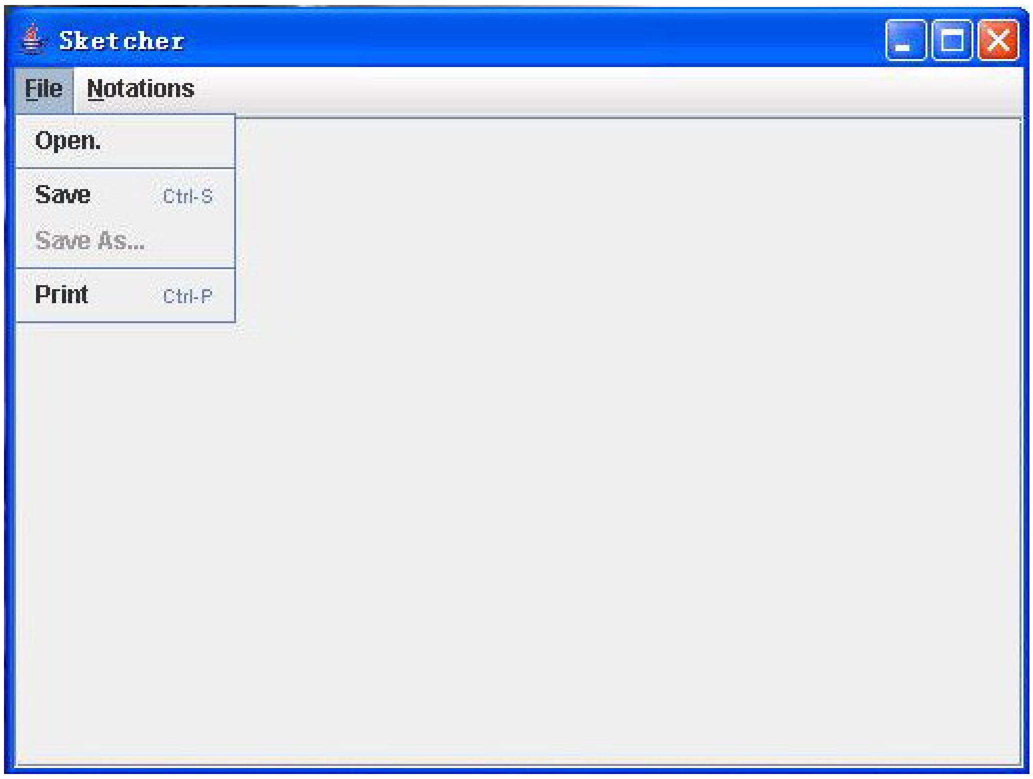


Figure 5-4 Starting Interface

**Description:**

This is the first interface shown in Figure 5-4 the user will see after execution. There is a menu bar on left hand side which is show. The user can open XML file, user can select Pure-Variant exported XML file. User can also select developer defined XML file. Menu dropdown bar has another option of save which user can select when he/she wants to save the selected XML file as developer defined structured XML file. In the begining the "Save As.."option is disable because no rendering is done on the frame. But once the rendering is done the user can save the rendered feature model as "Jpeg"image file format.

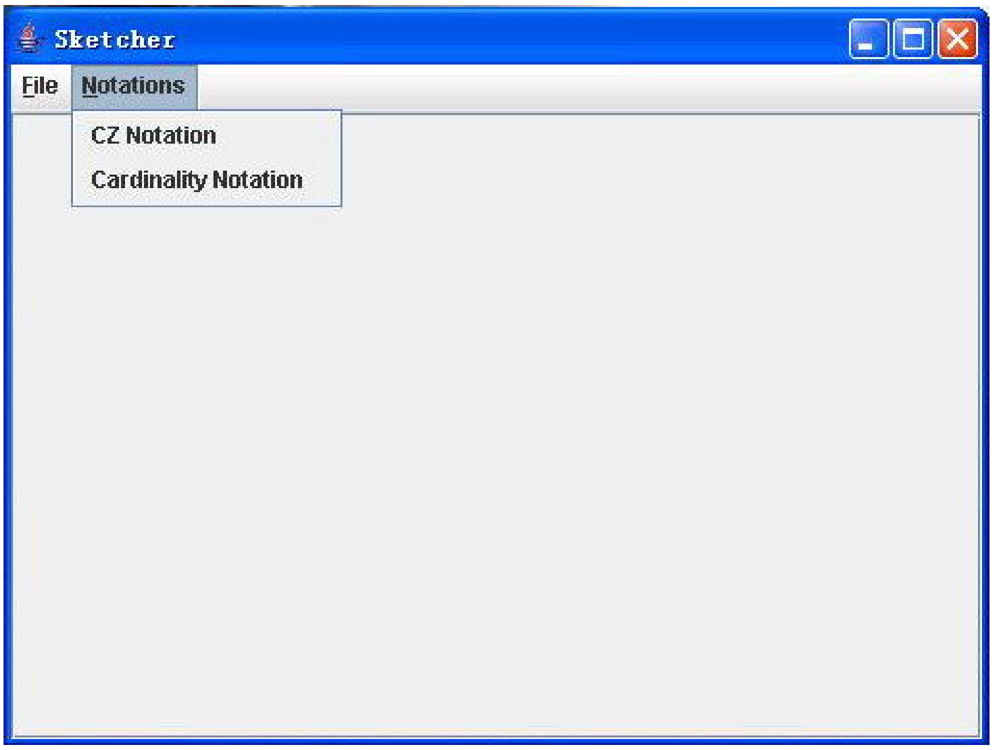


Figure 5-5 Notation System Selection

**Description:**

This interface shown in Figure 5-5 demonstrates that the user can also switch between notation systems. The developed result has two types of notation systems”C-Z ”notation system, and ”Cardinality based” notation system.

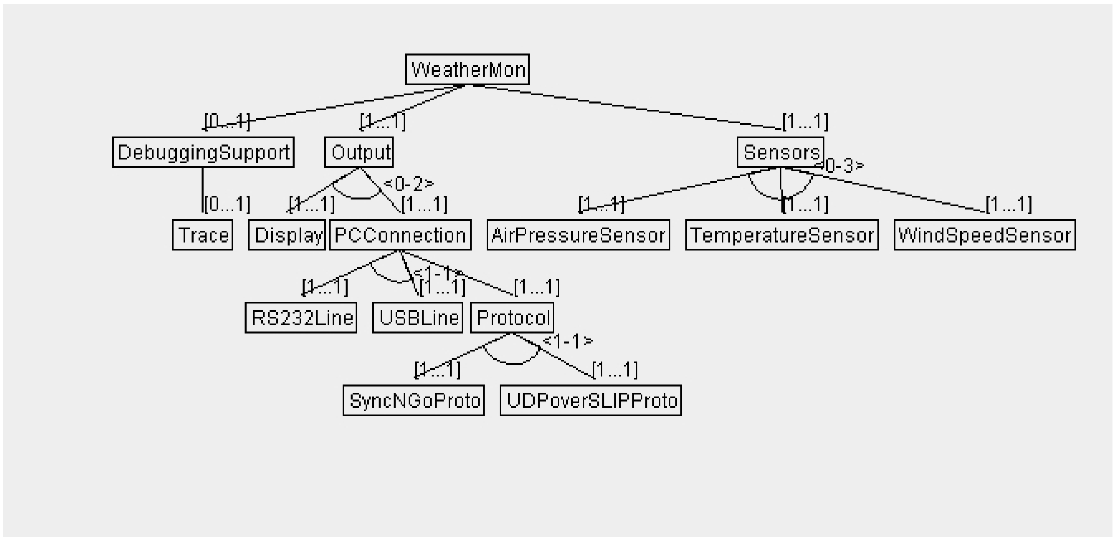


Figure 5-6 Rendered Cardinality based feature model

**Description:**

This interface shown in Figure 5-6 demonstrates rendering of one Pure-Variant exported XML file example in ”Cardinality based” notation system. The user can drag the features on layout canvas. This rendering is done when user selects the ”Cardinality Notation”from the menu bar ”Notations”.

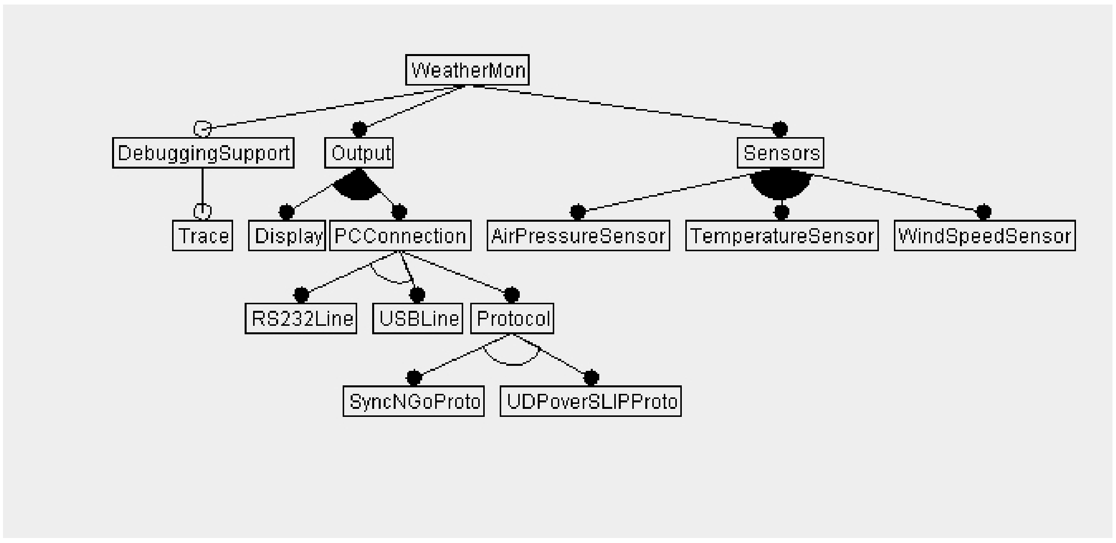


Figure 5-7 Rendered Feature Model in C-Z notation System

**Description:**

This interface shown in Figure 5-7 demonstrates rendering of one Pure-Variant exported XML file example in "C-Z" notation system. The user can drag the features on layout canvas. This rendering is done when user selects the C-Z notation from the menu bar "Notations".

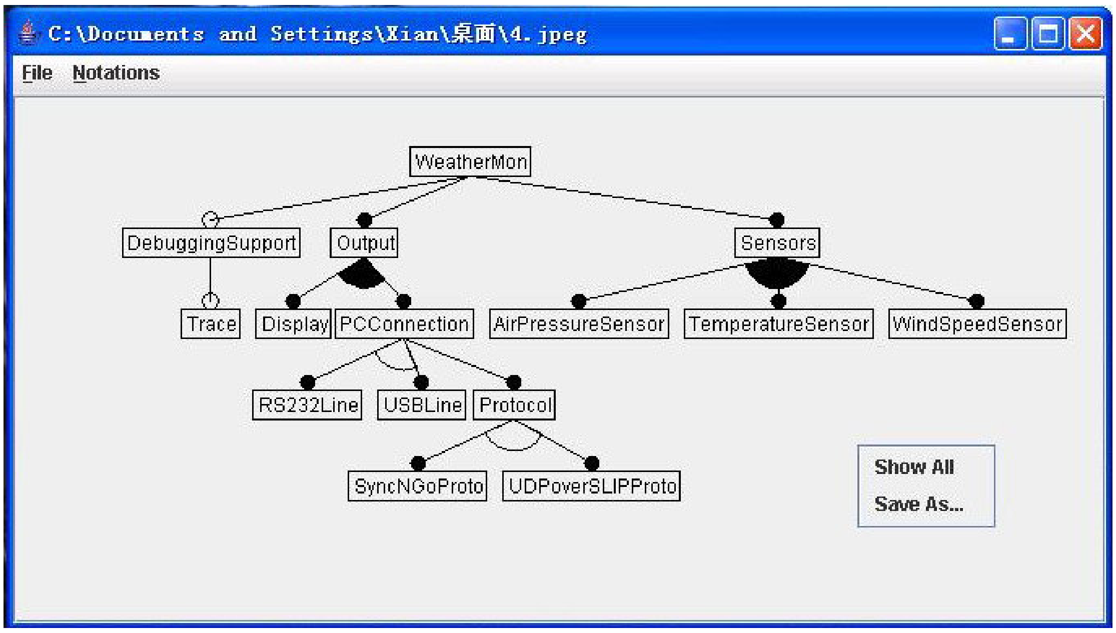


Figure 5-8 Canvas Popup menu

**Description:**

This interface shown in Figure 5-8 demonstrates that when on the rendered canvas user does the mouse right click there appears a "popup" menu which options like "Show All" and "Save As". "Show All"is active when user has already hide some features from the canvas. This popup menu item will show all features and childrens on rendering area. "Save As.."popup menu item has the same funtionality as the menu bar item has. It will save the rendered feature model in "Jpeg"image file format.

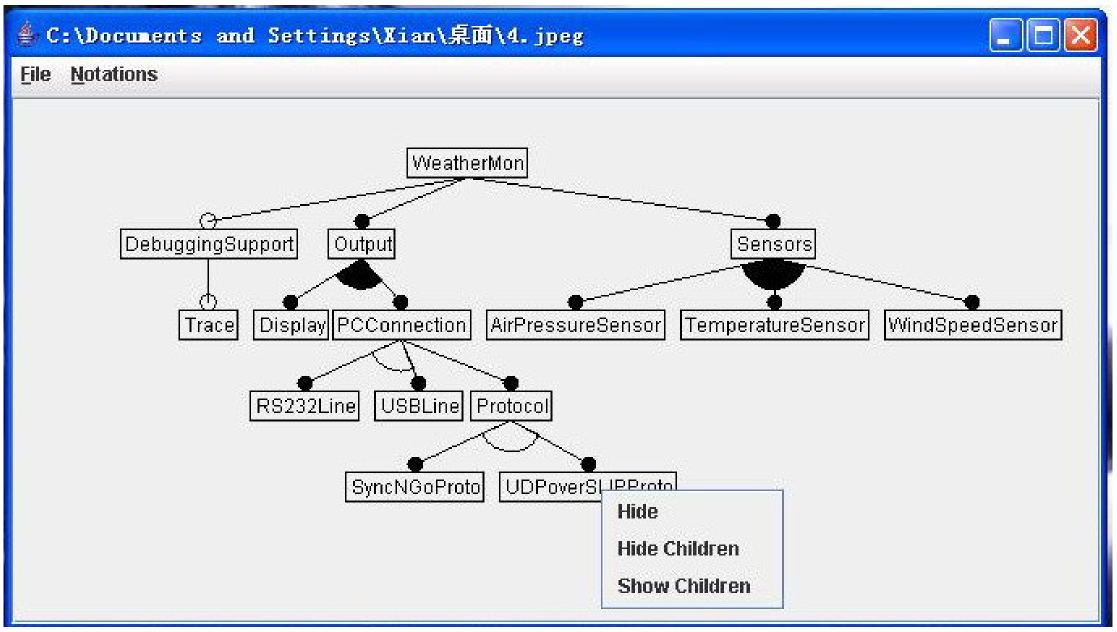


Figure 5-9 Feature PopUp Menu

**Description:**

This interface demonstrates shown in Figure 5-9 that when the user will mouse right click on some rendered feature there will appear a popup menu with following popup menu items ”Hide”,”Hide Children”,”Show Children”.”Hide” popup menu item will hide the particular feature from the rendering area. By selecting the ”Hide Children” it will hide the children of the particular rendered feature from rendering area. By selecting ”Show Children”it will show the hide children (if already hide) on rendering area.

## 5.5 Comparison of developed result with Pure-Variant

The developed result has some advantages over Pure-Variant rendering scheme.

- Developed result gives two notation systems to user for rendering feature models.
- Developed result render the optional and alternative features in groups which clear user mind about which are groups from he/she can select the features from feature model.
- User has clear vision about rendered features.
- Notation systems used are easy to interprate and understand.
- User has given the popup menu’s for more funtionality.

The developed solution may not be the best as compared to the rendering of feature models using ”Pure-Variant”. The developed solutions is just an attempt to give user more knowledge about new notation systems. It is an attempt to render feature models in a better way. Yet its not the best solution.

# 6 Conclusion and discussions

In this final thesis, standalone rendering software for feature modelling is developed. It gives user two more notation systems expressions to choose besides Pure-Variant graphical rendering tool. The developed result helps user to understand feature modelling concepts and notation system expressions. The results that have been concluded in research segment are developed in implementation section.

In variant management process, feature models helps to manage different product features, common feature groups as well as variable feature groups. There are different notation systems representing feature models. All notation systems are situation dependent. The research results give suggestions when and where it is suitable to use which kind of notation system. The created tool (NotationManager) was designed in order to be used by the company as a helpful tool for managing their artifacts. The company is recently using Pure-Variant as feature model rendering tool. NotationManager is rendering feature model into two other notation systems, C.E notation and C.B notation. C.E notation focuses on abstract level of feature models. C.B notation focuses on more detailed level of feature models.

The implementation of research results is done in Java with understanding of XML file structures. The developed result may not be the best solution in the given situation. It is an attempt to render feature models in efficient and in different notation system. The suggested two expressions for constraints in section 4 are developed in NotationManager. The developed solution works offline as standalone software. The communication is one sided between Pure-Variant and NotationManager .In future the standalone software can be developed into plug-in software for Eclipse, so the communication can be both sided.

# 7 References

[1] Carnegie Mellon University (2006) http://www.sei.cmu.edu/domain-engineering/domain_engineering.html (Acc. 2006-12-01)

[2] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.

[3] Dipl. –Inf. Krzysztof Czarnecki; (Oct. 1998) Generative Programming : Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models, [Ph. D. thesis ]. Technische Universitat Ilmenau ,Germany.

[4] Matthias Riebisch (2003), Towards a More Precise Definition of Feature Models, pp. 64-76.

[5] Giss, M. L. Favaro, J. d'Alessandro, M.: Integrating Feature Modeling with the RSEB. In: *Proc. of 5th International Conference on Software Reuse*, Vicoria, B.C., Canada. IEEE Computer Society Press (1998).

[6] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged Configuration Through Specialization and Multi-Level Configuration of Feature Models. Software Process Improvement and Practice, special issue on "Software Variability: Process and Management, 10(2), 2005, pp. 143 – 169

[7] K. Czarnecki and C. H. P. Kim. Cardinality-Based Feature Modeling and Constraints: A Progress Report. In OOPSLA'05 International Workshop on Software Factories (online proceedings), 2005

[8] Miloslav Šípka. Exploring the Commonality in Feature Modeling Notations. In Mária Bieliková, editor, Proceedings of IIT.SRC 2005: Student Research Conference in Informatics and Information Technologies, Bratislava, pages 139-144. Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, April 2005.

[9] The Eclipse Foundation (2006), http://www.eclipse.org/org/ (Acc. 2006-12-02)

[10] Pure Systems, http://www.pure-systems.com/Variant_Management.49.0.html (Acc.2006-12-02)

[11] Pure-Variants Concepts and UI interface, http://www.pure-systems.com/fileadmin/downloads/pv-intro-tool-concepts-en.pdf (Acc. 2006-12-02)