

East West University
Software product Line (SPL) Feature Tree Analysis Tool

By

Md. Iearul Islam

&

Hanufa Zaman

A project submitted in partial fulfillment for the degree of B.Sc. in
Computer Science and Engineering

In the

Faculty of Science and Engineering



Department of Computer Science and Engineering

January 2016

Declaration

We hereby declare that this submission is our own work and that to the best of our knowledge and belief it contains neither material nor facts previously published or written by another person. Further, it does not contain material or facts which to a substantial extent has been accepted for the award of any degree of a university or any other institution of tertiary education except where an acknowledgement.

(Hanufa Zaman)

(MD. Iearul Islam)

Letter of Acceptance

The project entitled “Software product Line Feature Tree Analysis Tool” submitted by MD. Iearul Islam (ID: 2011-3-60-023), Hanufa Zaman (ID: 2011-3-60-002), to Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh is accepted by the department in partial fulfillment of requirements for the Award of the Degree of Bachelor of Science in Computer Science and Engineering on January, 2016

Board of Examiners

Dr. Shamim H. Ripon

Assistant Professor and Chairperson,

Department of Computer Science and Engineering,

East West University, Dhaka, Bangladesh

Acknowledgement

First of all thanks to our Supervisor **Dr. Shamim H Ripon** for providing us this opportunity to test our skill in the best possible manner. He enlightened, encouraged and provided us with ingenuity to transform our vision into reality.

This project would not be possible without the help of our project supervisor, **Dr. Shamim H Ripon**, Associate Professor and Chairperson, Department of Computer Science and Engineering, East West University. He helps us to understand all the matters easily which make us to create this project. We would like to express our sincere and deep regards to him.

Lastly, we both are really thankful to Almighty **ALLAH**. So at the end deliberately we want to pay tribute to our parents. We call them “Our Heroes, Our Mentors”. These are the people that **ALLAH** has used to discover nature and deepen our academic career.

TABLE OF CONTENTS

Contents	Page No
Abstract	6
1 Introduction	7
1.1 Introduction and Motivation	7
1.2 Objectives	8
1.3 Contribution	8
1.4 Outline	9
2 Background	
2.1 Software Product Line	10
2.2 Feature Model	11
2.3 DOT	12
3 Logical Representation of Feature Model	
3.1 Feature Tree Representation	13
3.2 Feature Modeling Notation	13
3.2.1 Basic Feature Models	14
3.2.1.1 Mandatory	14
3.2.1.2 Optional	15
3.2.1.3 Alternative	15
3.2.1.4 Optional Alternative	16
3.2.1.5 Or	16
3.2.1.6 Optional Or	17
3.2.2 Example of Feature Tree	18
3.2.3 Cardinality Based Feature Models	18
3.2.4 Extended Feature Models	18
3.3 Feature Analysis	19
3.3.1 Feature Tree Format	19
3.4 Functional Decomposition	20
4 Tool Implementation	
4.1 Technical Feature	21
4.2 Overall Steps of the implementation	22
4.3 Implementation of each Component	24
5 User Manual	
5.1 Process Overview	26
5.2 Work Flow	27
5.2.1 Configure your workspace	27
5.2.2 Create Project	27
5.2.3 Add Features and Manage Logical Condition	28
5.2.4 View Project and Take Necessary Features	28
5.2.5 Get Graphical Representation	29
6 Conclusion	
6.1 Strength of this System	30
6.2 Drawback of the System	30
6.3 Future work	30
References	31
Appendix	32

List of Figures

Figure Name	Page No
Figure 3.1: Mandatory	14
Figure 3.2: Alternative	15
Figure 3.3: Optional Alternative	16
Figure 3.4: Or	16
Figure 3.5: Optional Or	17
Figure 3.6: Feature Tree Representation with the help of Logical notations	18
Figure 3.2.2: Feature Tree Representation with the help of Logical notations	16
Figure 4.1 Flow Chart Diagram For Create a New Project	22
Figure 4.2 Flow Chart Diagram to Make Customized Featured Tree	23
Figure 4.3 Database for Variants	24
Figure 4.4 Condition to Show Mandatory and Optional	24
Figure 4.5 Conditions for Or Statement	25
Figure 4.6 Function to view a complete project	25
Figure 5.1 Create Project	27
Figure 5.2 Add Features and Manage Logical Condition	28
Figure 5.3 View Project and Take Necessary Features	28
Figure 5.4 Full Graph of a Family Product	29
Figure 5.5 Customize Graph	29

Abstract

The concept of a software product line (SPL) is to promise about approaching for increasing planned reusability in industry. Feature models are enable for planning and strategic decisions both in architectural and in component development. Feature models contribute to efficiency and structure of various software development activities. Feature models have a tree structure, with features forming nodes of the tree. The arcs and groupings of features represent feature variability. There are six different types of feature groups: mandatory, alternative, or, optional, optional alternative and optional or. The logical representation provides a precise and rigorous formal interpretation of the feature diagrams. Software product lines (PLs) present a solid approach in large scale reuse. Due to the PLs' inherit complexity, many PL methods use the notion of "features" to support requirements analysis and domain modeling. The approach makes use of extensions in the feature modeling techniques and adopts plug-in architectures as a means of mapping feature structures and at the same time satisfying the demanded PL variability and flexibility.

Chapter 1

Introduction

1.1 Introduction and Motivation

The report introduces the overall description of “SPL Feature Tree Analysis Tool”. Software Product Line (SPL) is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. It will explain the Feature Tree by the rules of Logical representation. Software product lines centralize upon the idea of designing and implementing a family of systems to produce qualitative applications in a domain, promote large scale reuse and reduce development costs. [2] Features are abstract concepts effectively supporting communication among diverse stakeholders of a product line, and therefore, it is natural and intuitive for people to express commonality and variability of product lines in terms of features. Also, it has been recognized that the information codified by a feature model is most critical for developing reusable software assets. [3] Software product lines (PLs) replace the various separately-developed systems of a domain. Thus PLs embed at least the additive complexity present in each of these systems, posing a challenge in their development and demanding extensive variability. Additionally, PLs must follow all principles of modern software, being flexible, extendible and maintainable. In order to keep a balance between these requirements and virtues, PLs must adhere to a high level of abstraction: alone the variability and size of PLs impose the use of explicit domain modeling techniques and the development of a solid architecture. [2]

1.2 Objectives

Based on the logical rules, the concept of a feature is introduced. A formalized definition of a feature in the software field is given in a logical unit of behavior that is specified by a set of functional and quality requirements. A feature represents an aspect valuable to the customer. The main objectives of Software Product Lines (SPL) and the Feature Model Analysis Tools are given bellow:

- Develop a tool to represent feature Tree
 - Represent the logical relation of the features
 - Create customized tree based on user selection
- Graphically represent both full and customized tree.
- This tool can represent a logical tree by using internal logical conditions for a large family product.
- Using 6 logical rules from this family product it can select as the user require for and it can provide a customized graphical tree.
- It can manage the variability of Product Line. It can guide to follow the Product Line. It can improve the communication between the system and users.

1.3 Contribution

The aim of this chapter is to provide a comprehensive description of the notion of variability modeling in the context of software product line engineering and to give an overview of the techniques proposed for variability modeling. Product line can be modeled in many different ways based on different viewpoints. For the problem space, user goals and objectives, required quality attributes, and product usage contexts are typically modeled in product line engineering.

Most of the properties of SPL feature tree modeling have been worked properly. Though there are some drawbacks in our system. In this system we implement the sides that are give bellow

- A full tree of a family product can be given input and it also can provide graphical representation of this product.
- By following originating logical rules it can select feature and give a customized graphical representation.

1.4 Outline

Chapter 2: In this chapter we will glimpse the background that is about Software Product Line, Feature Model and Dot.

Chapter 3: Here the Logical Representation of Feature Model will be chronicled by the Logical representation of Features through the six logical rules and Feature analysis.

Chapter 4: Our Tool Implementation will be elucidated by the overall steps of the implementation through the flowchart and the implementation of each component.

Chapter 5: It is all about the user Manual of the whole system.

Chapter 6: The rest of the things of the report will be represented in this chapter. That is all about the conclusion part of this paper and we will also discuss the achievement from this project and the future work with this project to develop it more.

Chapter 7: Last but not list that is the guidance of reference and the codes full of appendix.

Chapter 2

Background

2.1 Software Product Line

Software product lines, or software product line development, refer to software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production. In, a definition of a software product line is given as “a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way”. [4]

Actually, Software product line (SPL) is a software intensive system sharing a common and managed set of features that satisfy the needs of particular market segment or mission and that are developed from a set of core assets in a particular market segment or mission and that are developed from a set of core assets in a prescribed way [5]. Product line technology is a way of improving the software development lifecycle and reuse by providing facilities to reuse the model of the system families, it is possible to increase the productivity and decrease the possible errors significantly. [6] The main idea of software product line is to explicitly identify all the activities which are common to all members of the family as well as which are different and then arrange them in a model. This implies a huge model which will help the stakeholders to be able to trace any design choices and variability decisions as well. Finally, the derivation of the product is done by selecting the required variants from the model and configuring them according to product requirements. Manufacturers have long employed analogous engineering techniques to create a product line of similar products using a common factory that assembles and configures parts designed to be reused across the product line. For example, automotive manufacturers can create unique variations of one car model using a single pool of carefully designed parts and factory specifically designed to configure and assemble those parts. The

characteristic that distinguishes software product lines from previous efforts is predictive versus opportunistic software reuse. Rather than put general software components into a library in the hope that opportunities for reuse will arise, software product lines only call for software artifacts to be created when reuse is predicted in one or more products in a well defined product line. Recent advances in the software product line field have demonstrated that narrow and strategic application of these concepts can yield order of magnitude improvements in software engineering capability. The result is often a discontinuous jump in competitive business advantage, similar to that seen when manufacturers adopt mass production and mass customization paradigms.

While early software product line methods at the genesis of the field provided the best software engineering improvement metrics seen in four decades, the last generation of software product line methods and tools are exhibiting even greater improvements. New generation methods are extending benefits beyond product creation into maintenance and evolution, lowering the overall complexity of product line development, increasing the scalability of product line practice with orders of magnitude less time, cost and effort. Domain and application engineering are the two main phases of SPL development. [7]

2.2 Feature Model

In software development, a feature model is a compact representation of all the products of the Software Product Line (SPL) in terms of feature. [4] 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. [8]

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 by Kang in 1990. Since then, feature modeling has been widely adopted by the software product line community and a number of extensions have been proposed.

A key technical innovation of software product-line is the use of feature to distinguish product-line members. A feature is an increment in program functionality. A particular product-line member is defined by a unique combination of features. The set of all legal feature combinations defines the set of product-line members. Feature models define features and their usage constraints in product-lines. Current methodologies organize features into a tree, called a feature diagram (FD), which is used to declaratively specify product –line members. Relationships among FDs and grammars, and FDs and formal models/logic programming have been noted in the past, but the potential of their integration is not yet fully realized.

2.3 DOT Graph (graph description language)

DOT is a plain text graph description language. It is a simple way of describing graphs that both humans and computer programs can use. DOT graphs are typically files that end with the .gv (or .dot) extension. The .gv extension is preferred in cases where there could be confusion with the .dot file extension used by early (pre-2007) versions of Microsoft word. [9]

Various programs can process DOT files. Some like OmniGraffle, dot, neato, twopi, circo, fdp, and tred will read a DOT file and render it in graphical form. Others, like gvpr, gc, acyclic, ccomps, sccmap, and tred, will a DOT file and perform calculations on the represented graph. Finally, others, like GVedit, KGraphEditor, lefty, dotty, and grappa provide an interactive interface. Most programs are part of the Grapviz package or use it internally.

Chapter 3

Logic Representation of Feature Model

3.1 Feature Tree Representation

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.

3.2 Feature modeling Notation

Current feature modeling notations may be divided into three main groups, namely:

- Basic feature models
- Cardinality-based feature models
- Extended feature models

3.2.1 Basic feature models

A Feature Model (FM) is a hierarchical arranged set of features. It represents all possible products of an SPL (Software Product Line) in a single model. Every Feature is an increment in product functionality. The complete feature tree of CAD domain is illustrated. It can be used in different stages of development. Though A FM is a tree like structure so it consists of relations between a parent feature and its child features, also cross-tree constrains that are typically inclusion or exclusion statements of the form “if a feature F is included, then feature X must also be included”. [10] The relation between a parent (variation point) features and its child feature (variants) are categorized as follows:

3.2.1.1 Mandatory

A child feature is said to be mandatory when it is required to appear when the parent feature appears. For instance, it is mandatory to have a special platform for Android mobile phone. A mandatory feature is included if its parent feature is included.

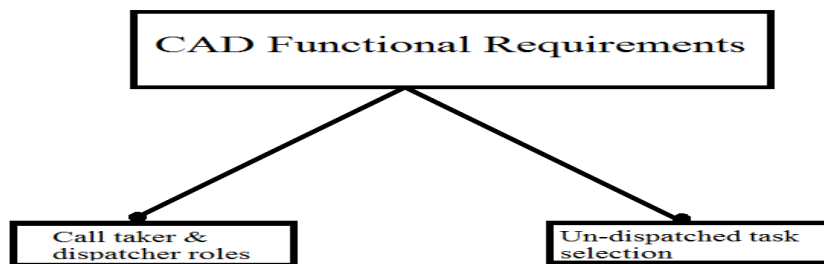


Figure 3.1: Mandatory

3.2.1.2 Optional

A child feature is said to be optional when it can or not appear when the parent features appears. For instance, it is optional to have pdf reader software in the mobile phone. An optional feature may or may not be included if its parent is included.

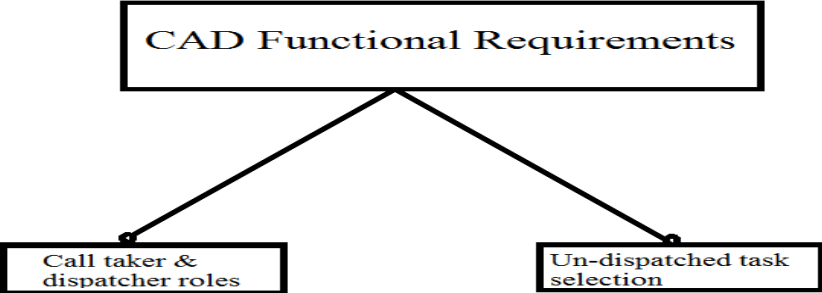


Figure 1.2: Optional

3.2.1.3 Alternative

A set of child features are said to be alternative when only one child feature can be selected when the parent feature appears. For instance a Gamer cannot select both Automatic and Manual for car control during car selection. One and only one feature from a set of alternative features are included when parent feature is included.

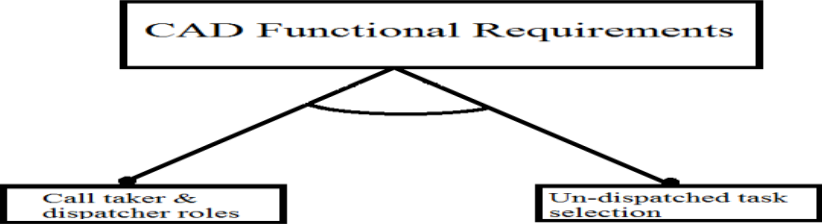


Figure 3.2: Alternative

3.2.1.4 Optional Alternative

One feature from a set of alternative features may or may not be included if parent included. One feature from a set of alternative features may or may not be included if parent included.

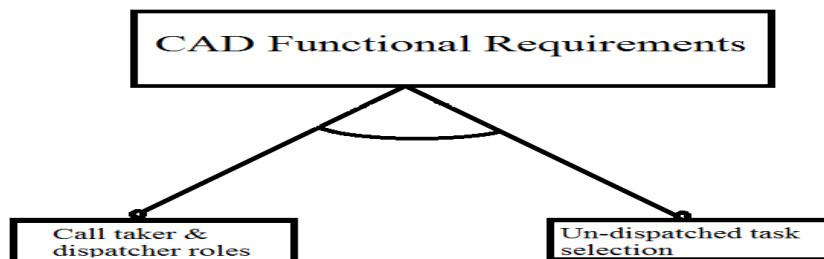


Figure 3.3: Optional Alternative

3.1.3.5 Or

A set of child features are said to have an or-relation with their parent when one or more sub features can be selected when the parent feature appears. For instance, the engine of a car can be electric, gasoline or both at the same time. At least one from a set of feature is included when parent is included.

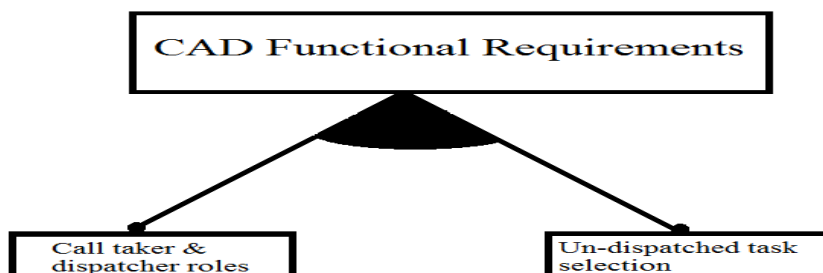


Figure 3.4: Or

3.2.1.6 Optional or

One or more optional feature may be included if the parent is included. One or more optional feature may or may not be included if the parent is included.

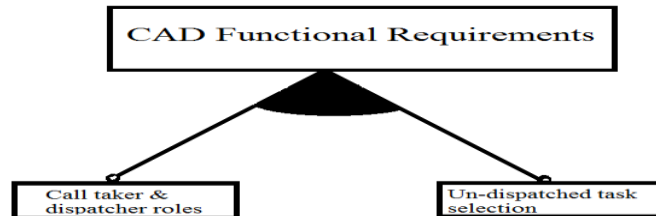


Figure 3.5: Optional Or

A feature model can be considered as a graph consists of a set of sub graphs. Each sub graph is created separately by defining a relationship between the variation point (denoted as v_i) and the variants ($v_{i,j}$) by using the expressions. The complexity of a graph construction lies in the definition of dependencies among variants. When there is a relationship between cross-tree (or cross hierarchy) variants (or variation points) we denote it as a dependency. Typically dependencies are either inclusion or exclusion: if there is a dependency between p and q, then p is included then q must be included (or excluded). Dependencies are drawn by dotted lines. [10]

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.

3.2.2 Example of a Feature Tree

As an example, the figure below illustrates how feature models can be used to specify and build configurable on-line shopping systems. The software of each application is determined by the features that it provides. The root feature (i.e. E-Shop) identifies the SPL. Every shopping system implements a catalogue, payment modules, security policies and optionally a search tool. E-shops must implement a high or standard security policy (choose one), and can provide different payment modules: bank transfer, credit card or both of them. Additionally, a cross-tree constraint forces shopping systems including the credit card payment module to implement a high security policy. [10]

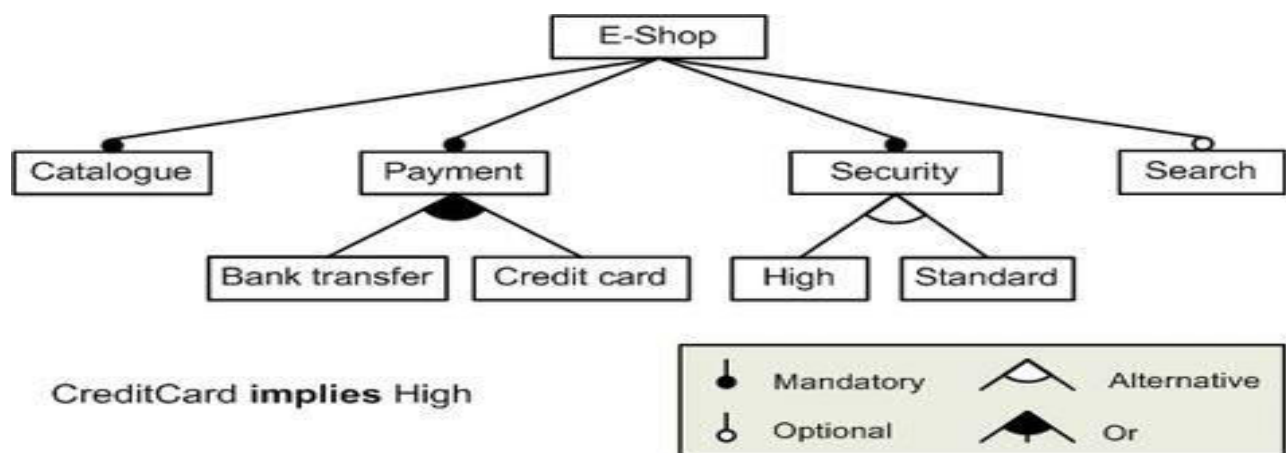


Figure 3.6: Feature Tree Representation with the help of Logical notations

3.2.3 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 as we want (as long as the other constraints are respected). This notation is useful for products extensible with an arbitrary number of components.

3.2.4 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.

3.3 Feature Analysis

A feature tree is a way to look at planned product features hierarchically, so that it can be quickly understood the relationships among product features that have specified. Feature Trees are high-level models organizing features into feature groups, capturing the entire scope of a project into a single model. A *feature* consists of one or more logically related system capabilities that provide value to a user and are described by a set of functional requirements. Many business analysts use features as a way to describe the scope of a project. However, a simple list does not readily show the size and complexity of various features. Nor does quickly skimming a feature list easily reveal the full scope of a project. A feature tree is a visual analysis model that organizes a set of features in a format that makes them easy to understand.

3.3.1 Feature Tree Format

The structure of feature trees is based on fishbone diagrams or Ishikawa diagrams, which are commonly used to organize information into logical groupings based on relationships. Fishbone diagrams are typically used to model cause-and-effect relationships, but feature trees use the same format to organize the planned features of a software solution.

A feature tree can show up to three levels of features, commonly called level 1 (L1), level 2 (L2), and level 3 (L3). L2 features are sub features of L1 features, and L3 features are sub features of L2 features. Below L3 lie individual requirements. A feature tree does not necessarily need to have three levels of features; if the solution and features are simple, there might be just L1 features and their more detailed functional requirements.

Features have an ideal abstraction level for communication and discussion with stakeholders about how the product really needs to be valued. Features sometimes come in lists like a backlog or in tables or are sometimes visually structured, these are the feature trees. In general there are two types of feature trees:

- Trees to decompose the product into different levels of features. They are visualized as a fishbone diagram or as a mind map or as story maps. They show a hierarchy and help to structure features into groups and visualize dependencies.
- Trees focusing on the development axis (feature vector): these trees grow from the stem to the leaves.

3.4 Functional Decomposition

Feature trees show all of the planned product features at once, giving a quick view of the solution's breadth of functionality. Organizing the features in this fashion makes it easy to identify missing and unnecessary features. The feature tree provides a functional decomposition of the solution for use throughout all phases of the project, including organization of the requirements, planning the work around the requirements, and bounding of the scope of work. Feature trees provide a much richer view of features than a simple list can show.

Chapter 4

Tool Representation

4.1 Technical Feature

- I. Programming Language: C and PHP
- II. Programming Method: Recursive Algorithm of Graph Traverse
- III. Design: HTML5, CSS3, JavaScript
- IV. Using Frameworks: Codeigniter 2.2.6
- V. Graph Visualization Software: graphviz (dot tools).

4.2 Overall steps of the implementation

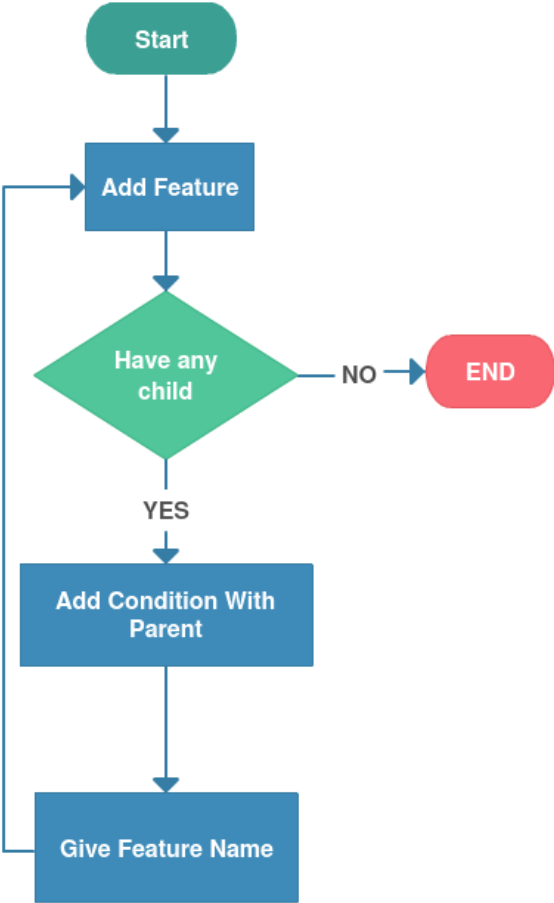


Figure 4.1 Flow Chart Diagram For Create a New Project

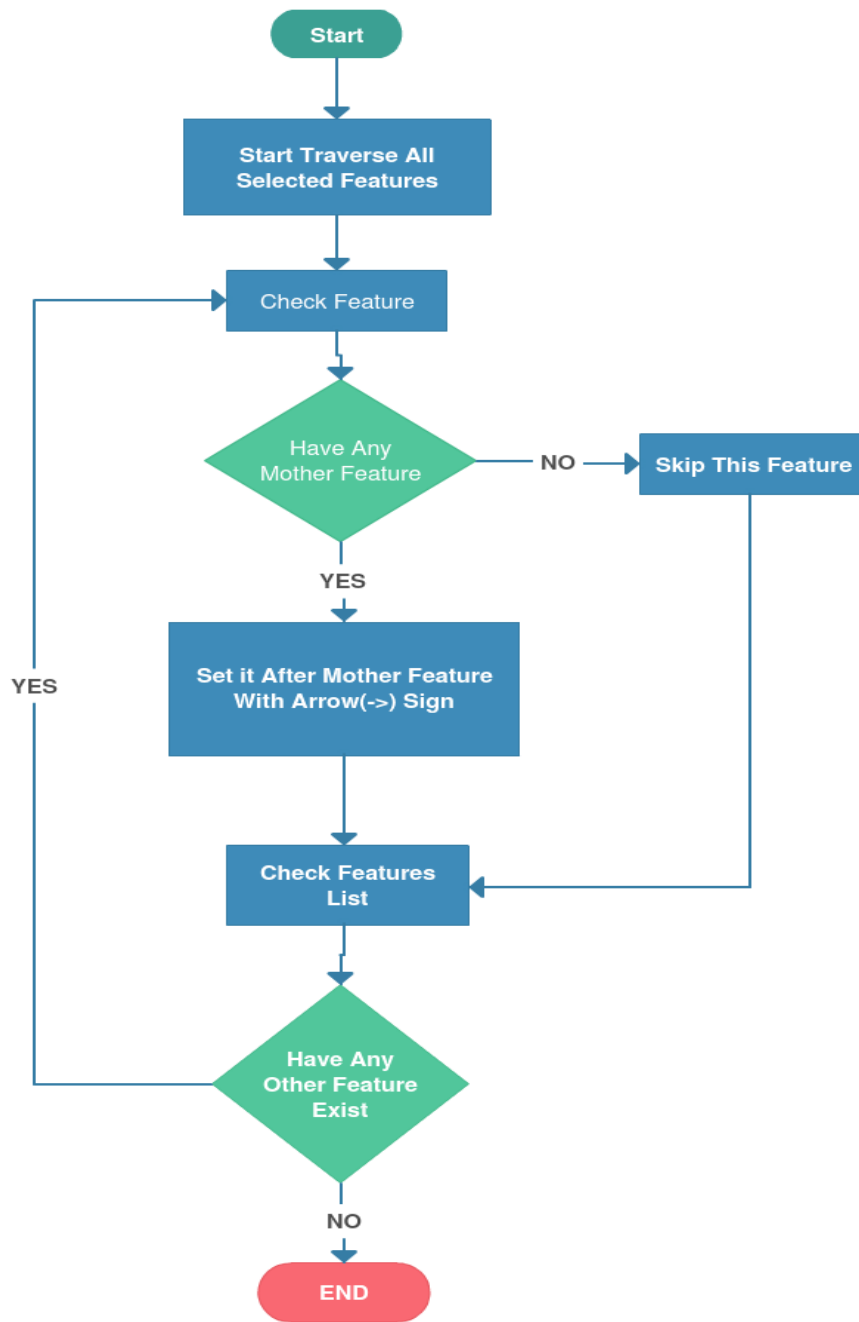


Figure 4.2 Flow Chart Diagram to Make Customized Featured Tree

4.3 Implementation of each component

+ Options										
		id	url	project_url	title	mother	condition	mother_condition	mother_type	
<input type="checkbox"/>				1	cad-functional-requirements	cad-functional-requirements	CAD Functional Requirements		Mandatory & Optional	0
<input type="checkbox"/>				8	call-taker-and-dispatcher-roles	cad-functional-requirements	Call Taker and dispatcher roles	cad-functional-requirements	Alternative	Mandatory & Optional
<input type="checkbox"/>				9	merged	cad-functional-requirements	Merged	call-taker-and-dispatcher-roles	Mandatory & Optional	Alternative
<input type="checkbox"/>				10	separated	cad-functional-requirements	Separated	call-taker-and-dispatcher-roles	Mandatory & Optional	Alternative
<input type="checkbox"/>				11	un-dispatched-task-selection	cad-functional-requirements	Un Dispatched Task Selection	cad-functional-requirements	Or	Mandatory & Optional
<input type="checkbox"/>				12	task-priority	cad-functional-requirements	Task Priority	cad-functional-requirements		Mandatory & Optional
<input type="checkbox"/>				13	urgency	cad-functional-requirements	Urgency	un-dispatched-task-selection		Or
<input type="checkbox"/>				14	fcfs	cad-functional-requirements	FCFS	un-dispatched-task-selection		Or
<input type="checkbox"/>				15	operator	cad-functional-requirements	Operator	un-dispatched-task-selection	Alternative	Or
<input type="checkbox"/>				16	priority	cad-functional-requirements	Priority	un-dispatched-task-selection		Or
<input type="checkbox"/>				17	call-taker	cad-functional-requirements	Call Taker	operator		Alternative
<input type="checkbox"/>				18	dispatcher	cad-functional-requirements	Dispatcher	operator		Alternative

Figure 4.3 Databases for Variants

```

jQuery(document).ready(function() {
    if($('#<?=$mother?>').is(":checked")) {
        $('.<?=$mother?>_show').show();
    }else{
        $('.<?=$mother?>_show').hide();
    }
    $('#<?=$mother?>').click(function() {
        if($('#<?=$mother?>').is(":checked")) {
            $('.<?=$mother?>_show').show();
        }else{
            $('.<?=$mother?>_show').hide();
        }
    });
    <?php if($mother_condition == 'Optional Alternative'){ ?>
    $('.<?=$mother?>_check').click(function() {
        if($(this).is(":checked")) {
            $('.<?=$mother?>_check').removeAttr('checked');
            $(this).attr('checked', true);
            $.uniform.update();
        }
    });
    <?php } ?>

```

Figure 4.4 Condition to Show Mandatory and Optional

```

<?php if($mother_condition == 'Or'){ ?>
$( ".<?=$mother?>_check" ).click(function(){
    if( $(this).is(":checked")){

    }else{
        if( $('.<?=$mother?>_check').is(":checked")){
        }else{
            $(this).attr('checked', true);
            $.uniform.update();
            var v_id = $(this).attr('id');
            if( $('#'+v_id).is(":checked")){
                $('.'+v_id+'_show').show();
            }else{
                $('.'+v_id+'_show').hide();
            }
        }
    }
});
<?php } ?>

```

Figure 4.5 Conditions for Or Statement

```

function view($url = NULL){
    if(empty($url)){
        $this->session->set_flashdata('message_error','Project Not Exists. ');
        redirect('projects','refresh');
    }
    $where = array(
        'url' => $url
    );
    $data['project'] = $this->CoreZ_IT->get_where('projects', $where)->row();
    if(empty($data['project']->url)){
        $this->session->set_flashdata('message_error','Project Not Exists. ');
        redirect('projects','refresh');
    }
    $where = array(
        'project_url' => $url
    );
    $variants = $this->CoreZ_IT->get_where('variants', $where)->result();
    $data['variants'] = array();
    $data['child_variants'] = array();
    foreach($variants as $variant){
        if(!empty($variant->mother)){
            $data['child_variants'][$variant->mother][] = $variant;
        }else{
            $data['variants'][] = $variant;
        }
    }
    $data['message'] = $this->session->flashdata('message');
    $data['message_error'] = (validation_errors() ? validation_errors() : $this->session->flashdata('message_error'));
    $data['page_title'] = 'projects';
    $this->CoreZ_IT->render_backend('view', $data);
}

```

Figure 4.6 Function to view a complete project

Chapter 5

User Manual

5.1 Process Overview

To use this system we need a PHP5 supported hosting services. XAMPP is one of the best localhost services. For online use we can maintain any online hosting service. To make graphical view we need dot program which is provided by Graphviz.

Sequence for using the software to manage functions:

1. Configure your workspace
2. Create a Project
3. Add Features
4. Managing Logical Conditions
5. View Project
6. Take Necessary Features
7. Print Graphical Representation

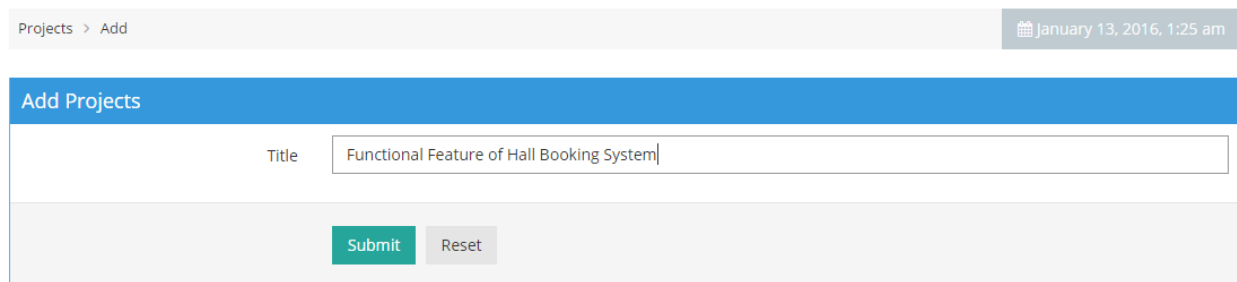
5.2 WorkFlow

Here we describe step by step the procedure to use the tools. How to create the database. How to add a project and how to get the customize graphical view etc.

5.2.1 Configure your workspace

1. Install PHP5 in your web host server.
2. Install Graphviz software on that server and add dot program path on default environment variable path.
3. Put all source files on root folder.
4. Add the database on phpmyadmin mysql database.
5. Configure database username and password with system.

5.2.2 Create Project



The screenshot shows a web interface for adding a project. At the top, there is a breadcrumb 'Projects > Add' and a timestamp 'January 13, 2016, 1:25 am'. Below this is a blue header bar labeled 'Add Projects'. The main form area has a 'Title' label and a text input field containing 'Functional Feature of Hall Booking System'. At the bottom of the form are two buttons: a green 'Submit' button and a grey 'Reset' button.

Figure 5.1 Create Project

Click Add project give a project title then press the submit button. Now you get a new project.

5.2.3 Add Features and Manage Logical Condition

The screenshot shows a web form titled "Add Projects". The form is structured as follows:

- Title:** Functional Feature of Hall Booking System
- Logical Representation for It's Child:** Mandatory & Optional
- Child:** Add New
- Title:** Reservation Charge
- Title:** Reservation Mode
- Title:** Notification
- Title:** Reservation Management
- Title:** Handle Conflicts

At the bottom of the form, there are two buttons: "Submit" and "Reset".

Figure 5.2 Add Features and Manage Logical Condition

Add a feature and add its child's and select the logical condition with mother feature.

5.2.4 View Project and Take Necessary Features

The screenshot shows a web form titled "Logic Representation". The form is structured as follows:

- Checkboxes:**
 - Functional Feature of Hall Booking System
 - Reservation Charge
 - Deposite
 - Tax
 - Basic Charge
 - Discount
 - Reservation Mode
 - Block
 - Multiple Rooms
 - Multiple Times
 - Single
 - Notification
 - Reservation Management
 - Add, Modify
 - Delete
 - Handle Conflicts

At the bottom of the form, there are two buttons: "Submit" and "Cancel".

Figure 5.3 View Project and Take Necessary Features

Click on view button beside on project that you need. Checkmark on your require features what you want to get with your project.

5.2.5 Get Graphical Representation

Click on Full Graph but for the full family product graphical view and submit button for your customized graph.

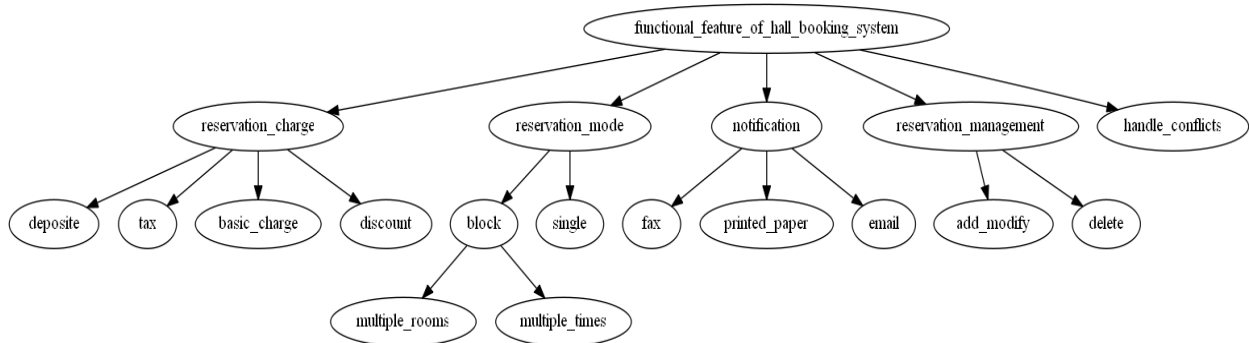


Figure 5.4 Full Graph of a Family Product

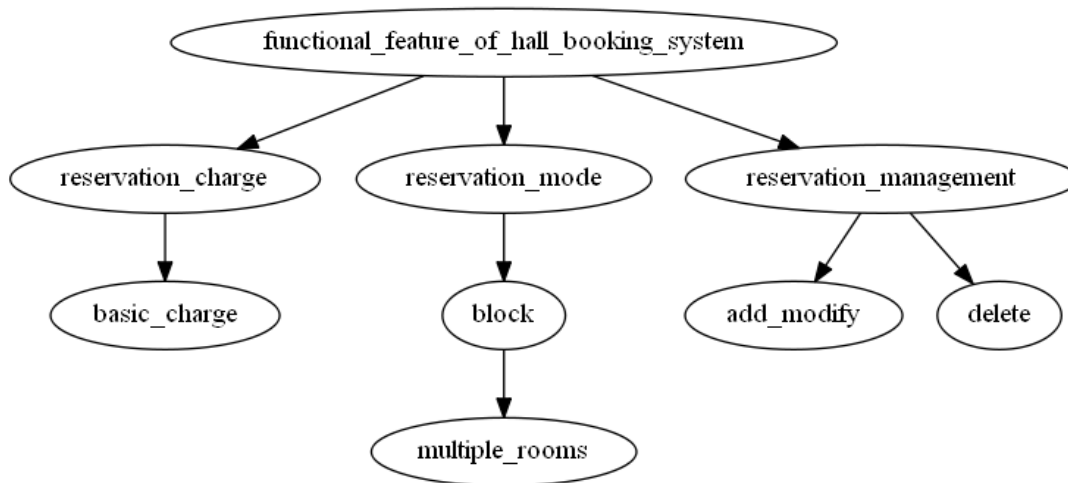


Figure 5.5 Customize Graph

Chapter 6

Conclusion and Future Work

As large as the family product remains we can represent it in a graphical view and also it can be represented by the logical rules which can create a customized tree and its graphical image.

6.1 Strength of this System

- It can work with any number of features.
- As the term of level increase as much as it can so that the system can work properly.
- It can provide both the Logical and Graphical image.
- As much as the family product remains the system can represent it properly.
- The system is fully dynamic so anyone can add a new project without any programming knowledge.

6.2 Drawback of the System

- It cannot express internal dependency
- Multiple types of relation without mandatory and optional cannot be implemented on child from same mother variant.

6.3 Future Work

In future we want to work with this project. We want to improve this system. There are some features we want to add though the system, they are given below

- We will append internal dependency between features of different mother variants.
- In future we will contrivance all type of logical relation between features of same mother tree.

References

1. http://ceur-ws.org/Vol-509/paper_9.pdf
2. https://swk-www.informatik.uni-hamburg.de/~riebisch/publ/FArM_NODE.pdf
3. http://www.springer.com/cda/content/document/cda_downloaddocument/9783642365829-c1.pdf
4. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. The description logic handbook: theory, implementation, and application. Cambridge University Press, New York, NY, USA, 2003.
5. Paul Clements and Linda Northrop. Software Product Lines: Practices and Patterns. Addison-Wesley Professional, 3rd edition, August 2001.
6. David Benavides, Pablo Trinidad, and Antonio Ruiz-Cortes. Automated reasoning on feature models. In Proceedings of the 17th international conference on Advanced Information Systems Engineering, CAiSE'05, pages491-503, Berlin, Heidelberg, 2005. Springer-verlag.
7. Daniel Beradi. Using dls to reason on uml class diagrams. In In Proc. Workshop on Applications of Description Logics, pages1-11, 2002.
8. Krzysztof Czarnecki and Ulrich W. Eisenecker. Generative programming: methods. Tools and applications. ACM Press/Addision-Wesley Publishing Co., New York, NY, USA, 2000.
9. DOT Graph (graph description language)
[http://en.wikipedia.org/wiki/DOT_\(graph_description_language\)](http://en.wikipedia.org/wiki/DOT_(graph_description_language))
10. https://en.wikipedia.org/wiki/Feature_model

Appendix

```
1. <?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2.
3. class Projects extends CZ_Controller{
4.
5.     /**
6.      *
7.      * @copyright      Copyright (c) 2016.
8.      * @author         Md Iearul Islam and Hanufa Zaman
9.      * @version        1.0.2
10.     * @Last Update    11/29/2015
11.     *
12.    */
13.    function __construct()
14.    {
15.        parent::__construct();
16.    }
17.
18.    function index(){
19.        $data['projects'] = $this->CoreZ_IT->get('projects')->result();
20.        $data['message'] = $this->session->flashdata('message');
21.        $data['message_error'] = (validation_errors()) ? validation_errors() : $this-
22.        >session->flashdata('message_error');
23.        $data['page_title'] = 'All Projects';
24.        $this->CoreZ_IT->_render_backend('index', $data);
25.    }
26. }
```

To view the variant list with parent from database.

```
25.    function variants($url = NULL){
26.        if(empty($url)){
27.            $this->session->set_flashdata('message_error','Project Not Exists. ');
28.            redirect('projects','refresh');
29.        }
30.        $where = array(
31.            'url' => $url
32.        );
33.        $data['project'] = $this->CoreZ_IT->get_where('projects', $where)->row();
34.        if(empty($data['project']->url)){
35.            $this->session->set_flashdata('message_error','Project Not Exists. ');
36.        }
37.    }
```

```

36.         redirect('projects','refresh');
37.     }
38.     $where = array('project_url' => $url);
39.     $variants = $this->CoreZ_IT->get_where('variants', $where)->result();
40.     $data['variants'] = array();
41.     foreach($variants as $variant){
42.         $data['variants'][$variant->url] = $variant;
43.     }
44.     $data['url'] = $url;
45.     $data['message'] = $this->session->flashdata('message');
46.     $data['message_error'] = (validation_errors()) ? validation_errors() : $this-
>session->flashdata('message_error');
47.     $data['page_title'] = 'All Varients';
48.     $this->CoreZ_IT->_render_backend('variants', $data);
49. }
50. function add_variants($url = NULL){
51.     if(empty($url)){
52.         $this->session->set_flashdata('message_error','Project Not Exists.');
```

```

53.         redirect('projects','refresh');
54.     }
55.     $where = array(
56.         'url'     => $url
57.     );
58.     $data['project'] = $this->CoreZ_IT->get_where('projects', $where)->row();
59.     if(empty($data['project']->url)){
60.         $this->session->set_flashdata('message_error','Project Not Exists.');
```

```

61.         redirect('projects','refresh');
62.     }
63.     $where = array('project_url' => $url);
64.     $this->form_validation->set_rules('title', 'Title', 'required');
```

```

65.     if($this->form_validation->run() == true)
66.     {
67.         $url_var = $this->CoreZ_IT->url_check($this->input->post('title'), 'variants');
```

```

68.         $condition = $this->input->post('condition');
```

```

69.         $values = array(
70.             'title'     => $this->input->post('title'),
71.             'url'       => $url_var,
72.             'condition' => $condition,
73.             'project_url' => $url

```

```

74.         );
75.         $this->CoreZ_IT->insert('variants',$values);
76.
77.         $mother = $url_var;
78.         $childs = $this->input->post('child');
79.         foreach($childs as $child){
80.             $url_var = $this->CoreZ_IT->url_check($child, 'variants');
81.
82.             $values = array(
83.                 'title'           => $child,
84.                 'url'             => $url_var,
85.                 'mother'         => $mother,
86.                 'project_url'    => $url,
87.                 'mother_condition' => $condition
88.             );
89.             $this->CoreZ_IT->insert('variants',$values);
90.
91.         }
92.         $this->session->set_flashdata('message','Variants Successfully Added.');
```

```

93.         redirect('projects/variants/'.$url, 'refresh');
```

```

94.     }
95.     $data['title'] = array(
96.         'class' => 'form-control',
97.         'name'  => 'title',
98.         'type'  => 'text',
99.         'value' => $this->form_validation->set_value('title'),
100.     );
101.     $data['child'] = array(
102.         'class' => 'form-control',
103.         'name'  => 'child[]',
104.         'type'  => 'text'
105.     );
106.     $data['condition_value'] = array(
107.         'Mandatory & Optional' => 'Mandatory & Optional',
108.         'Alternative'          => 'Alternative',
109.         'Optional Alternative' => 'Optional Alternative',
110.         'Or'                   => 'Or',
111.         'Optional Or'          => 'Optional Or'
112.     );
113.     $data['condition_name'] = 'condition';
114.     $data['condition_selected'] = $this->form_validation->set_value('condition', 'Mandatory & Optional');
```

```

112.         $data['dropdown_class'] = 'class="form-control"';
113.         $data['url'] = $url;
114.         $data['message'] = $this->session->flashdata('message');
115.         $data['message_error'] = (validation_errors()) ? validation_errors() : $this-
>session->flashdata('message_error');
116.         $data['page_title'] = 'projects';
117.         $this->CoreZ_IT->_render_backend('add_variants', $data);
118.     }
119.     function edit_variants($url = NULL){
120.         if(empty($url)){
121.             $this->session->set_flashdata('message_error','Project Not Exists.');
```

```

122.             redirect('projects','refresh');
```

```

123.         }
124.         $where = array(
125.             'url' => $url
126.         );
127.         $data['variant'] = $this->CoreZ_IT->get_where('variants', $where)->row();
128.         if(empty($data['variant']->url)){
129.             $this->session->set_flashdata('message_error','Variant Not Exists.');
```

```

130.             redirect('projects','refresh');
```

```

131.         }
132.         $this->form_validation->set_rules('title', 'Title', 'required');
```

```

133.         if($this->form_validation->run() == true)
134.         {
135.             $condition = $this->input->post('condition');
```

```

136.             $values = array(
137.                 'title' => $this->input->post('title'),
138.                 'condition' => $condition
139.             );
140.             if($data['variant']->mother_condition == 'Mandatory & Optional'){
141.                 $values['mother_type'] = ($this->input-
>post('mother_type') == 'Mandatory' ? 1 : 0 );
142.             }
143.             $this->CoreZ_IT->update('variants', $values, $where);
144.             $values = array(
145.                 'mother_condition' => $condition
146.             );
147.             $this->CoreZ_IT->update('variants', $values, array('mother' => $url));
148.             $mother = $url;

```

```

149.         $childs = $this->input->post('child');
150.     foreach($childs as $child){
151.         if(!empty($child)){
152.             $url_var = $this->CoreZ_IT->url_check($child, 'variants');
153.             $values = array(
154.                 'title'      => $child,
155.                 'url'        => $url_var,
156.                 'mother'     => $mother,
157.                 'project_url' => $data['variant']->project_url,
158.                 'mother_condition' => $condition
159.             );
160.             $this->CoreZ_IT->insert('variants',$values);
161.         }
162.     }
163.     $this->session->set_flashdata('message','Variants Successfully Added.');
```

```

164.     redirect('projects/variants/'.$data['variant']->project_url, 'refresh');
```

```

165. }
166. $data['title'] = array(
167.     'class' => 'form-control',
168.     'name'  => 'title',
169.     'type'  => 'text',
170.     'value' => $this->form_validation->set_value('title', $data['variant']->title),
171. );
172. $data['condition_value'] = array(
173.     'Mandatory & Optional' => 'Mandatory & Optional',
174.     'Alternative'          => 'Alternative',
175.     'Optional Alternative' => 'Optional Alternative',
176.     'Or'                   => 'Or',
177.     'Optional Or'         => 'Optional Or'
178. );
179. $data['condition_name'] = 'condition';
180. $data['condition_selected'] = $this->form_validation->set_value('condition', $data['variant']->condition);
181.
182. $data['status_value'] = array(
183.     'Mandatory' => 'Mandatory',
184.     'Optional'  => 'Optional'
185. );
```

```

186.         $data['status_name'] = 'mother_type';
187.         $data['status_selected'] = $this->form_validation-
>set_value('mother_type', ($data['variant']->mother_type == 1 ? 'Mandatory': 'Optional'));
188.         $data['dropdown_class'] = 'class="form-control"';
189.         $data['child'] = array(
190.             'class' => 'form-control',
191.             'name' => 'child[]',
192.             'type' => 'text'
193.         );
194.         $data['url'] = $url;
195.         $where = array(
196.             'mother' => $url
197.         );
198.         $data['child_variants'] = $this->CoreZ_IT->get_where('variants', $where)-
>result();
199.         $data['message'] = $this->session->flashdata('message');
200.         $data['message_error'] = (validation_errors()) ? validation_errors() : $this-
>session->flashdata('message_error');
201.         $data['page_title'] = 'projects';
202.         $this->CoreZ_IT->_render_backend('edit_variants', $data);
203.     }

```

To delete existing variant

```

204.     function deleteVariant($url = NULL){
205.         $where = array('url' => $url);
206.         $data['variant'] = $this->CoreZ_IT->get_where('variants', $where)->row();
207.         if(empty($data['variant'])){
208.             $this->session->set_flashdata('message_error', 'Variant not Exists.');
```

```

209.             redirect('projects', 'refresh');
```

```

210.         }
```

```

211.         $this->CoreZ_IT->delete('variants', $where);
212.         $this->session->set_flashdata('message', 'Variant has been deleted.');
```

```

213.         redirect('projects/variants/'.$data['variant']->project_url, 'refresh');
```

```

214.     }

```

To add new project

```

215.     function add(){
216.         $this->form_validation->set_rules('title', 'Title', 'required');
```

```

217.         if($this->form_validation->run() == true)

```

```

218.     {
219.         $url = $this->CoreZ_IT->url_check($this->input->post('title'), 'projects');
220.         $values = array(
221.             'title'           => $this->input->post('title'),
222.             'url'             => $url
223.         );
224.         $this->CoreZ_IT->insert('projects',$values);
225.         $this->session->set_flashdata('message','Project Successfully Added.');
```

226. redirect('projects', 'refresh');

```

227.     }
228.     $data['title'] = array(
229.         'class' => 'form-control',
230.         'name'  => 'title',
231.         'type'  => 'text',
232.         'value' => $this->form_validation->set_value('title'),
233.     );
234.     $data['message'] = $this->session->flashdata('message');
```

235. \$data['message_error'] = (validation_errors()) ? validation_errors() : \$this->

```

    >session->flashdata('message_error');
```

236. \$data['page_title'] = 'projects';

237. \$this->CoreZ_IT->_render_backend('add', \$data);

238. }

To edit variant child and their conditions

```

239. function edit($url = NULL){
240.     if(empty($url)){
241.         $this->session->set_flashdata('message_error','Project Not Exists.');
```

242. redirect('projects','refresh');

```

243.     }
244.     $where = array(
245.         'url'  => $url
246.     );
247.     $data['project'] = $this->CoreZ_IT->get_where('projects', $where)->row();
248.     if(empty($data['project']->url)){
249.         $this->session->set_flashdata('message_error','Project Not Exists.');
```

250. redirect('projects','refresh');

```

251.     }
252.     $this->form_validation->set_rules('title', 'Title', 'required');
```

253. if(\$this->form_validation->run() == true)

```

254.         {
255.             $values = array(
256.                 'title'          => $this->input->post('title')
257.             );
258.
259.             $this->CoreZ_IT->update('projects', $values, $where);
260.             $this->session->set_flashdata('message', 'Project Successfully
Updated. ');
261.             redirect('projects', 'refresh');
262.         }
263.         $data['title'] = array(
264.             'class' => 'form-control',
265.             'name'  => 'title',
266.             'type'  => 'text',
267.             'value' => $this->form_validation->set_value('title', $data['project']-
>title),
268.         );
269.         $data['message'] = $this->session->flashdata('message');
270.         $data['message_error'] = (validation_errors()) ? validation_errors() : $this-
>session->flashdata('message_error');
271.         $data['page_title'] = 'projects';
272.         $this->CoreZ_IT->render_backend('edit', $data);
273.     }

```

To Show Full Project with all features

```

274.     function view($url = NULL){
275.         if(empty($url)){
276.             $this->session->set_flashdata('message_error', 'Project Not Exists. ');
277.             redirect('projects', 'refresh');
278.         }
279.         $where = array(
280.             'url'  => $url
281.         );
282.         $data['project'] = $this->CoreZ_IT->get_where('projects', $where)->row();
283.         if(empty($data['project']->url)){
284.             $this->session->set_flashdata('message_error', 'Project Not Exists. ');
285.             redirect('projects', 'refresh');
286.         }
287.         $where = array(

```



```

288.         'project_url' => $url
289.     );
290.     $variants = $this->CoreZ_IT->get_where('variants', $where)->result();
291.     $data['variants'] = array();
292.     $data['child_variants'] = array();
293.     foreach($variants as $variant){
294.         if(!empty($variant->mother)){
295.             $data['child_variants'][$variant->mother][] = $variant;
296.         }else{
297.             $data['variants'][] = $variant;
298.         }
299.     }
300.     $data['message'] = $this->session->flashdata('message');
301.     $data['message_error'] = (validation_errors()) ? validation_errors() : $this->session-
>flashdata('message_error');
302.     $data['page_title'] = 'projects';
303.     $this->CoreZ_IT->_render_backend('view', $data);
304. }

```

To make Customize Graphical Image

```

305.     private function child_variant($mother, $mother_condition, $child_variants){
306.         $text = '';
307.         if(!empty($child_variants[$mother])){
308.             $childs = $this->input->post($mother);
309.             if(!empty($childs)){
310.                 if($mother_condition == 'Alternative' || $mother_condition == 'Optional
Alternative'){
311.                     $text .= str_replace('_', ' ', $mother).' _> '.str_replace('_',
', ' ', $childs).';';
312.                     $text .= $this->child_variant($childs,$child_variants[$mother][$childs]-
>condition, $child_variants);
313.                 }else{
314.                     foreach($childs as $child){
315.                         $text .= str_replace('_', ' ', $mother).' _> '.str_replace('_',
', ' ', $child).';';
316.                         $text .= $this->child_variant($child,$child_variants[$mother][$child]-
>condition, $child_variants);
317.                     }
318.                 }

```

```

319.
320.     }
321. }
322.     return $text;
323. }

```

To make Full Graphical Image

```

324.     private function child_variant2($mother, $child_variants){
325.         $text = '';
326.
327.         if(!empty($child_variants[$mother])){
328.             $childs = $child_variants[$mother];
329.             foreach($childs as $key => $child){
330.                 $text .= str_replace('_', '_', $mother).' _> '.str_replace('_',
331.                 ' ', '_ ', $key).'.';
332.
333.                 $text .= $this->child_variant2($key, $child_variants);
334.             }
335.         }
336.
337.         return $text;
338.     }
339. }

```

To Select necessary feature from family product and download graphical Image

```

336.     public function submit($url = NULL){
337.         $this->load->helper('download');
338.
339.         if(empty($url)){
340.             $this->session->set_flashdata('message_error', 'Project Not Exists. ');
341.             redirect('projects', 'refresh');
342.         }
343.
344.         $where = array(
345.             'url' => $url
346.         );
347.         $project = $this->CoreZ_IT->get_where('projects', $where)->row();
348.
349.         if(empty($project->url)){
350.             $this->session->set_flashdata('message_error', 'Project Not Exists. ');
351.             redirect('projects', 'refresh');
352.         }
353.
354.         $where = array(
355.             'project_url' => $url
356.         );

```

```

353.         $vars = $this->CoreZ_IT->get_where('variants', $where->result());
354.         $variants = array();
355.         $child_variants = array();
356.         foreach($vars as $variant){
357.             if(!empty($variant->mother)){
358.                 $child_variants[$variant->mother][$variant->url] = $variant;
359.             }else{
360.                 $variants[] = $variant;
361.             }
362.         }
363.         $text = ' digraph G {';
364.         foreach($variants as $variant){
365.             $text .= $this->child_variant($variant->url, $variant->condition, $child_variants);
366.         }
367.         $text .= '}';
368.         $file = 'abid.dot';
369.         file_put_contents($file, $text);
370.         exec("check.exe");
371.         $result = file_get_contents('new.png');
372.         force_download($project->title.'.png', $result);
373.     }

```

To download Customize Graphical Image

```

374.     public function full_graph($url = NULL){
375.         $this->load->helper('download');
376.         if(empty($url)){
377.             $this->session->set_flashdata('message_error','Project Not Exists. ');
378.             redirect('projects','refresh');
379.         }
380.         $where = array(
381.             'url' => $url
382.         );
383.         $project = $this->CoreZ_IT->get_where('projects', $where->row());
384.         if(empty($project->url)){
385.             $this->session->set_flashdata('message_error','Project Not Exists. ');
386.             redirect('projects','refresh');
387.         }
388.         $where = array(

```

```

389.         'project_url' => $url
390.     );
391.     $vars = $this->CoreZ_IT->get_where('variants', $where)->result();
392.     $variants = array();
393.     $child_variants = array();
394.     foreach($vars as $variant){
395.         if(!empty($variant->mother)){
396.             $child_variants[$variant->mother][$variant->url] = $variant;
397.         }else{
398.             $variants[] = $variant;
399.         }
400.     }
401.     $text = ' digraph G {';
402.     foreach($variants as $variant){
403.         $text .= $this->child_variant2($variant->url, $child_variants);
404.     }
405.     $text .= '}';
406.     $file = 'abid.dot';
407.     file_put_contents($file, $text);
408.     exec("check.exe");
409.     $result = file_get_contents('new.png');
410.     force_download($project->title.'_full.png', $result);
411. }
412.
413.}

```