The Pennsylvania State University

The Graduate School

Industrial Engineering Department

COMPUTER-AIDED GENERATION OF MODULARIZED CONCEPTUAL DESIGNS WITH ASSEMBLY AND VARIETY CONSIDERATIONS

A Thesis in

Industrial Engineering

by

Saraj Gupta

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2008

The thesis of Saraj Gupta was reviewed and approved* by the following:

Gül E. Okudan Kremer
Associate Professor of Engineering Design and Industrial Engineering
Thesis Adviser

Soundar R.T. Kumara
Allen E. Pearce/Allen M. Pearce Chaired Professor of Industrial Engineering

Richard J. Koubek
Professor of Industrial and Manufacturing Engineering
Head of the Department, Industrial and Manufacturing Engineering

*Signatures are on file in the Graduate School

ABSTRACT

Design is the foremost step in the development of any electro-mechanical product, and conceptual design stage is the most ambiguous and creative phase of design. Only few computational tools exist at this design stage, and mostly designers rely on personal experience or experience of co-workers to generate quality designs. Accommodating manufacturing and assembly considerations during the design phase is found to be imperative to prevent frequent problems from occurring during the production stage. In recent years, modularity has received a tremendous interest as a product design strategy, but very few modularization methodologies are found to focus on the conceptual design stage. Product variety management has gained primary importance in this era of mass customization as manufacturing firms see variety issues as the key to profitability.

The proposed research framework aims at generating computerized conceptual product designs by incorporating Modularity, Design for Assembly (DFA) and Design for Variety (DFV) principles at the conceptual design stage. Conceptual design alternatives from the proposed framework are ranked based on minimum assembly time and are composed of modules in a way that future changes in customer needs are satisfied only by replacing certain modules. The framework involves searching a design repository of components by using functional-basis and pre-defined graph grammar rules, to generate all possible conceptual design alternatives. These are ranked and filtered using a DFA index, and top two alternatives are selected. Selected designs are modularized and filtered using a DFV index to obtain the best design alternative. The proposed framework is demonstrated through the design of two electro-mechanical products.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my heartfelt gratitude to my adviser Dr. Gül Kremer for her unending support and guidance in the development of this research. I am greatly indebted to her for providing me the financial support throughout my graduate studies. Her insightful suggestions, coupled with personal advice and motivation, has enabled me to work harder in achieving my goals. In addition, I would like to thank Dr. Soundar R.T. Kumara for agreeing to be my reader for my MS thesis. He has been a very good mentor, offering me advice whenever I needed it. I would also like to thank my parents Rajnish Gupta and Savita Gupta for constantly supporting me and motivating me at every step throughout my MS program.

I extend my sincere thanks to all my professors who have given me the knowledge and wisdom that has helped me in ascending the echelons of success. Finally, I would like to thank all the staff members and friends, who have made my stay at Penn State so enjoyable and full of unforgettable experiences.

Chapter 1

Introduction

1.1 Motivation

Design is the foremost step in the development of any electro-mechanical product. However, the existing design tools in the literature primarily focus either on the initial design phase such as customer needs gathering, or on the later steps of design embodiment or detailed design (Bryant et al., 2005a). Existence of only few computational tools at the conceptual design stage has caused the designers to be limited to few options like drawing based on personal experience or experience of co-workers, utilizing patent searches and reverse-engineering of products to create quality designs (Bryant et al., 2005a). During the conceptual design stage, designers not only need to determine the physical nature of design, but verify the design functions as well (Bryant et al., 2005a; Bryant et al., 2005b). It is found that most of the formal approaches at the conceptual design phase of the design process are difficult to translate into computational algorithms. For example, Computer Aided Design (CAD) tools are extensively being used at the detailed design stages, but the shortcomings of traditional CAD tools for the conceptual design stage of electro-mechanical products has been well-documented in the literature (Tor et al., 2002). Moreover, conceptual design phase is found to be the most ambiguous, most creative and least understood stage of design (Benami and Jin, 2000). As a result, conceptual design phase was choosen for this research, since a tremendous research opportunity exists in this design phase.

In the past, once the drawings and CAD models were made, they were transferred to the manufacturing and assembly engineers (Boothroyd, 1994). These engineers would work on optimizing the processes used to produce the final product. However, manufacturing and assembly problems were found to occur frequently at this point, causing the fabricators to request changes in the product design. Therefore, accommodating manufacturing and assembly considerations within the design phase was found to be imperative, which led to the inception of the Design for Manufacturing and Assembly (DFMA) methodology. Over the years, a rising number of companies started to address DFMA in order to improve the manufacturing situation. This is because, manufacturing companies have found it imperative that changes and interventions with regards to manufacturing aspects should take place at the design stage in order to result in optimum manufacturing of the product (Dumitrescu and Szecsi, 2002). DFMA was seen as a way to increase the productivity significantly without any significant investment (Fabricius, 1994). Over the years, DFMA has aided in the development of efficient Computer Aided Manufacturing (CAM) tools, and integration of CAM tools into CAD is quite common these days. Modularity is one of the guidelines for the DFMA process, and it has received a growing interest as a product design strategy.

In the modular product architecture, there is a one-to-one interaction with the functional elements in the function structure to the physical components of the product (Ulrich and Tung, 1991). Implementation of modularity offers several advantages to the manufacturing firms as it enables them to manage a large variety of products and produce them at lower costs and shorter lead times (Gershenson et al., 2003; Mikkola and Gassmann, 2002; Kusiak, 2002). For example, modular product design decouples developmental tasks and enables modules to be developed concurrently in order to reduce development time (Ulrich and Tung, 1991). Although a lot of research involving modularization of product architecture has been carried out at the detailed design stage, very few methodologies were found in the design literature that deal with modularity at the conceptual design stage (Gupta and Okudan, 2007).

Recently, challenges by global competition and mass customization have led to a paradigm shift from the design of single products to the design of the product families (Rai and Allada, 2003). Hence, product variety management has gained primary importance in this era of mass customization, leading to the inception of the Design for Variety (DFV) strategy (Martin and Ishii, 1996, 1997, 2000, and 2002). As a result, manufacturing companies have started viewing variety issues as the key to profitability and seek to optimize the costs of producing all the varieties of products together (Martin and Ishii, 1996). This has further led to the development of product family design, enabling manufacturing firms to offer two or more highly differentiated products, which share a substantial fraction of their components (Jiao and Tseng, 1999).

Overall, several opportunities exist in the design literature, each individually aiding the designers in designing cost-efficient and innovative designs early in the development of electro-mechanical products. It is estimated that the combined effect of all the aforementioned design approaches will be tremendous, and the proposed research aims at developing a computational framework aiding designers at the conceptual design stage by amalgamating DFA, Modularization and DFV methodologies. Overall goal of this research is to develop a stand-alone computational tool that aids the designers in satisfying customer needs, reducing the conceptual designing time and cost for the manufacturing firms, designing products based on ease of assembling, modularizing the product architecture and lastly, to generate variety. Section 1.2 provides an overview of the proposed research framework.

1.2 Overview of the Proposed Framework

The proposed research framework seeks to generate computer aided modularized conceptual designs by considering assembly and variety issues. This research is inspired by four distinct research tools that have been slightly modified and integrated together into a robust framework. Figure 1-1 shows the four research tools adapted by the proposed framework.

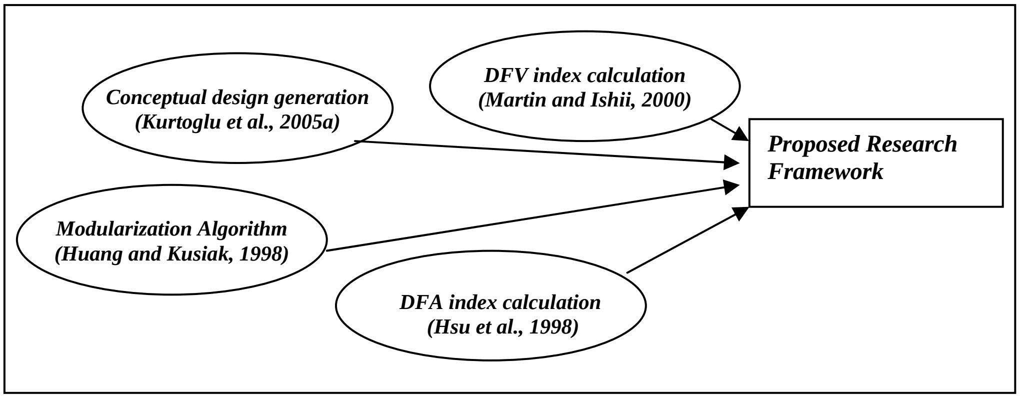


Figure 1-1: Research tools inspiring the proposed framework

The research comprises of five major steps that need to be executed before the modularized conceptual design is completely generated. A pre-defined design repository also needs to be populated by inputting 1) graph grammar rules, 2) component specifications like image, DFA index, and the rule associated with the component and 3) interaction among the various rules. Figure 1-2 briefly shows the various steps involved in the proposed framework, which are explained in-depth in Chapter 3.

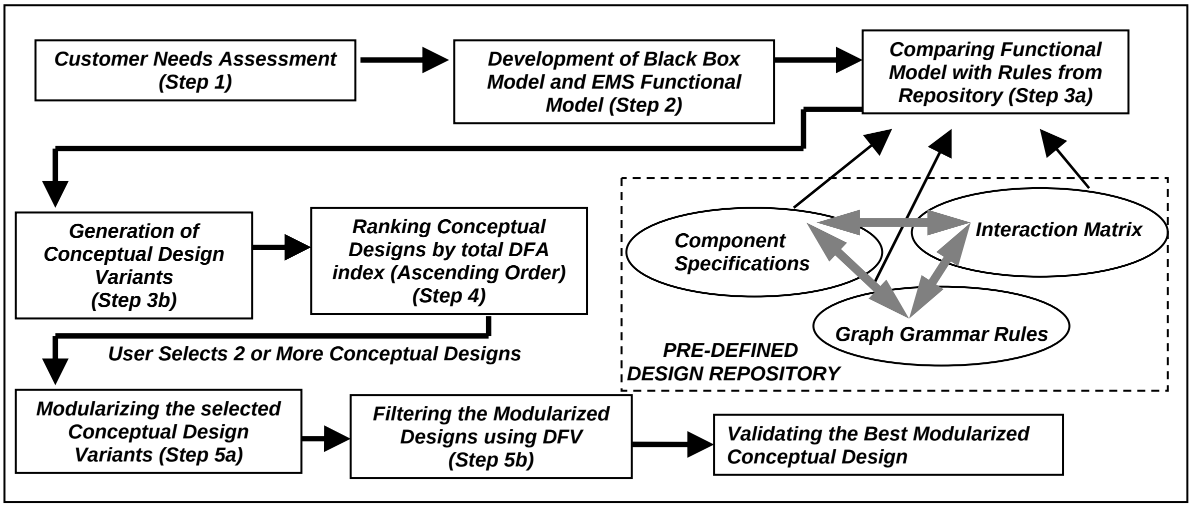

Customer Needs Assessment (Step 1)
Development of Black Box Model and EMS Functional Model (Step 2)
Comparing Functional Model with Rules from Repository (Step 3a)
Generation of Conceptual Design Variants (Step 3b)
Ranking Conceptual Designs by total DFA index (Ascending Order) (Step 4)
Component Specifications
Interaction Matrix
Graph Grammar Rules
PRE-DEFINED DESIGN REPOSITORY
User Selects 2 or More Conceptual Designs
Modularizing the selected Conceptual Design Variants (Step 5a)
Filtering the Modularized Designs using DFV (Step 5b)
Validating the Best Modularized Conceptual Design

Figure 1-2: Steps of operation in the proposed research framework

The interactions amongst the design team, Graphical User Interface (GUI) as well as design repository are highlighted in Figure 1-3.

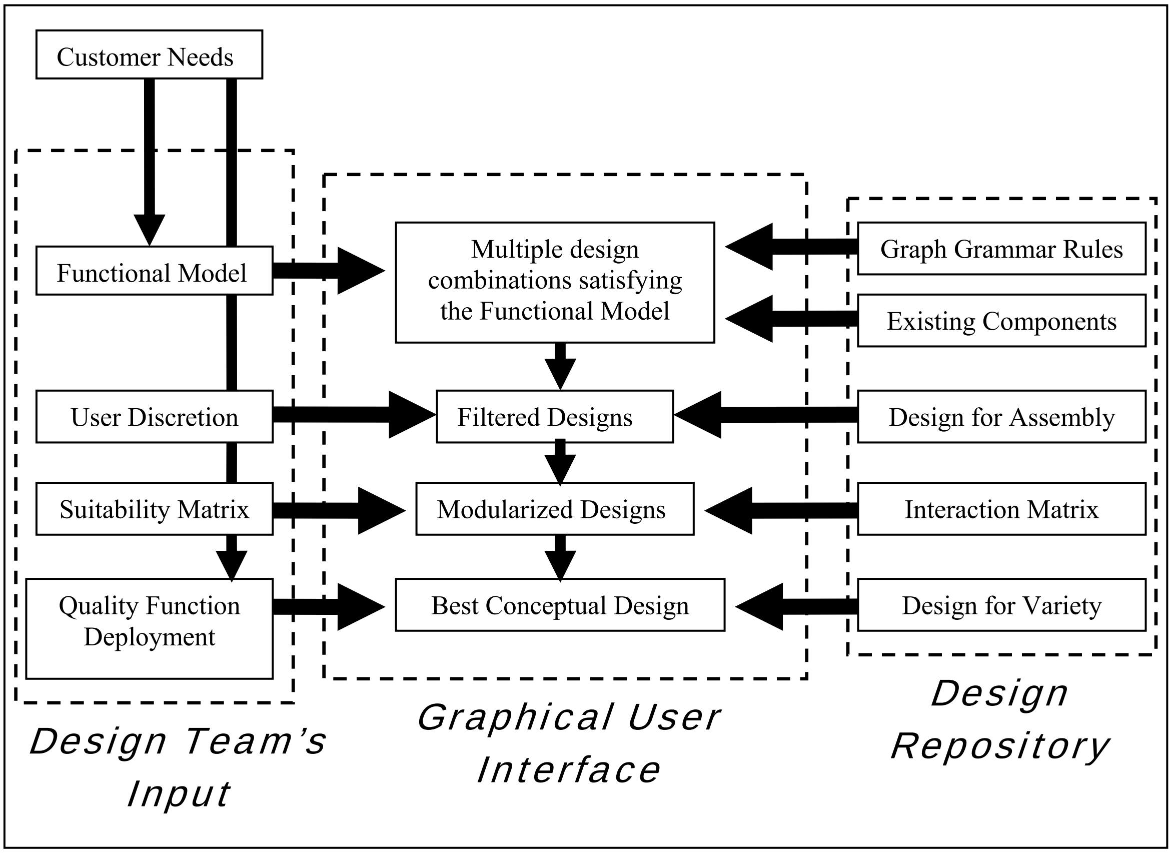


Figure 1-3: Interactions within the proposed software framework.

As shown in Figure 1-3, the proposed framework primarily comprises of a GUI that is linked to a design repository. The design team obtains the customer needs and generates a functional model that is input in to the GUI. The design repository comprises of component specifications, graph grammar rules and interaction matrix, which supplies all the necessary information required to calculate the DFA index. Once the design team selects the designs to be modularized, they would need to input the suitability matrix that combines with the interaction matrix present in the design repository to generate the modularized designs. Furthermore, the design team utilizes the customer needs to generate Quality Function Deployment (QFD) matrix that is input into the

GUI. This QFD matrix is utilized to generate the DFV index that is used to select the best conceptual design. Section 1.3 discusses the roadmap for this research thesis.

1.3 Thesis Roadmap

In the design literature, Design for Assembly, Modularity, and Design for Variety have individually been found to benefit the manufacturing firms and enabling them to gain a competitive advantage over their peers. As a result, integration of these three methodologies within a conceptual design software framework is predicted to be of tremendous benefit to the firms. Chapter 2 contains a detailed literature review on the three product design strategies and it also highlights their benefits to handle a variety of manufacturing related problems. Probing into the literature also reveals that a lot of research has focused on trying to make the CAD technology accessible to the designers at the conceptual stage. However, there has been no mention of any efficient way to filtering the resultant conceptual designs that are generated as a result of Computer Aided Conceptual Design (CACD) tools. Chapter 2 also discusses the function-based approach to conceptual product designs that has been gaining popularity these days.

Chapter 3 provides a detailed step-by-step explanation of the proposed research framework along with figures and examples, which enable the reader to gain a deeper understanding of the software research framework. In Chapter 4, the working of the proposed framework is illustrated through two conceptual design examples. The first example involves the generation of the conceptual design of an electronic toothbrush by inputting the corresponding functional model into the software framework. The second example involves the design of a mounting system for holding a Variable Message Sign (VMS).

Finally, the results obtained from the research framework are validated by using a Design Envelopment Analysis (DEA) based concept selection tool. A comparison between the two research tools is provided in order to demonstrate the effectiveness and validity of the proposed framework. The results of the testing and validation stage are discussed in Chapter 5. The thesis concludes in Chapter 6, where the potential scope for improvements is discussed and recommendations are provided for future research.

Chapter 2

Literature Review

In this chapter, an extensive literature review of existing conceptual design tools is presented in order to highlight the uniqueness of the proposed research framework. Chapter 2 begins with an introduction to different product design stages and gradually focusing specifically on the conceptual design stage. Section 2.1 highlights the importance of customer needs assessment, while Section 2.2 explains the functional basis of design. Section 2.3 discusses some of the existing computer aided design tools utilized for concept generation. Section 2.4 highlights the importance of design repositories for storing the details of existing components in order to generate innovative conceptual designs. A brief overview of product modularity is provided in Section 2.5 along with brief descriptions of modularity methodologies at different stages of design. Section 2.6 explains the Design for Manufacturing and Assembly (DFMA) strategy, while Section 2.7 provides a brief overview of a relatively newer concept: Design for Variety (DFV).

As mentioned previously, foremost step in the development of any electro-mechanical product is its design. In the literature, design has been categorized into four distinct phases (Pahl and Beitz, 1998). The first phase involves the collection of information about the requirements to be embodied in the design and the constraints involved. The second phase, which is also known as conceptual design phase, involves the establishment of functions that make up a product or a system. The interaction between the mechanisms corresponds to the function inputs and outputs. In the third phase, the designer determines the layout and form of the product in accordance with technical and economic considerations. Lastly, in the detailed design stage, the arrangement, form, dimensions and surface of all individual components are finally laid out and all components are formally identified.

Conceptual design is the most challenging of all the four design stages since the product knowledge at this stage is imprecise, uncertain, ambiguous or probabilistic in nature (Nepal et al., 2005). During the conceptual design stage, design is generally carried out in the form of functions or mechanisms as opposed to the actual physical components during the detailed phase of the design. Designers generate ideas at the conceptual stage and immediately turn them into two dimensional (2D) quick sketches with tools like paper and pencil (Qin et al., 2003). The designers then need to determine the physical nature of design, as well as verify the design functions at this stage. Few computational tools exist at this stage, causing the designers to be limited to few options like drawing based on personal experience or experience of co-workers, utilizing patent searches and reverse-engineering of products to create quality designs (Bryant et al., 2005a). At the detailed design stage however, multiple computational tools like Computer Aided Design (CAD) and Finite Element Analysis (FEA) are easily available to aid the designer.

It is observed that the costs incurred during the production of the product, its use and disposal, are generally committed by early design decisions (Seo et al., 2002). The studies reported by Dowlatshahi (1992) indicate that design of the product influences 70% to 85% of the total cost of the product. This can be reduced by giving due consideration to the life cycle cost issues accruing during the conceptual design phase (Seo et al., 2002). Ullman (1997) observed that 75% of the final design costs are obligated during the conceptual design stage. Moreover, conceptual design stage is the best time to impact the economies of a product. Obtaining customer preferences is imperative before the generation of conceptual designs. Section 2.1 reviews the methodologies existing in literature, which deal with the assessment of customer needs.

2.1 Customer Needs Assessment

To a large extent, the success of any product depends on the design team's awareness of the customer needs. Hence, customer driven design and manufacturing is an important strategy for the manufacturing firms, and all decisions must relate to the satisfaction of customers by listening to the 'voice of the customer' (Lin and Che, 1998). In the design literature, Quality Function Deployment (QFD) has emerged as a systematic approach to design based on a close awareness of customer desires, coupled with the integration of corporate functional groups (Rosenthal, 1992). QFD is a popular tool at the conceptual design stage as it converts customer demands into quality characteristics and systematically develops a quality plan for deployment of the finished product. QFD was developed by Yoji Akao in Japan in 1966. Akao pointed out that, when QFD is appropriately applied, it can reduce the development time by one-half to one-third (Akao, 1990). QFD uses principles from Concurrent Engineering (CE) in that cross-functional teams are involved in all phases of product development. The QFD process is classified into four phases, and uses a matrix to translate customer requirements from initial planning stages through production control (Becker Associates Inc., 2000).

The product planning phase of QFD is also known as House Of Quality (HOQ), which documents customer requirements, warranty data, competitive opportunities, product measurements, competing product measures, as well as technical ability to meet each customer's requirements (Hauser and Clausing, 1988). HOQ involves constructing a collection of sub-matrices, each containing information relating to others (Eggen, 2003).

Martin and Ishii (2000) have incorporated a modified QFD structure in their DFV related research. The modified structure consists of two phases: 1) QFD phase – I, which lists the customer needs and their relationship to engineering metrics, and 2) QFD phase – II, which maps the engineering metrics to the components used in the design.

Overall, QFD is found to be a widely used tool for the assessment of customer needs and obtaining good customer data is found to be critical to the success of the entire QFD process. Once the customer needs have been obtained, the next step involves the development of the functional structure of the product to be developed. Section 2.2 introduces the functional basis of the design terminology.

2.2 Functional Basis of Design

The use of function has been recognized as an important part of the design process and early representation of function was performed in the field of artificial intelligence (Szykman et al., 1999). According to Yoshikawa (1980), the information retrieval process that involves a one-to-one mapping between functional space and attribute space is one of the most important models of the design process. There is no clear and uniform definition of function obtained in the design literature this far. However, the definition given by Rodenacker (1991) that defines function as an input and output of material, energy and information, is widely accepted in the design research (Muogboh, 2003). Another definition of function, given by Pahl and Beitz (1998), characterizes function as a general input/output relation of a system whose main purpose is to perform a task. Functional knowledge is found to be important in predicting, observing, describing and verifying the device behavior (Iwasaki et al., 1995).

Stone and Wood (2000) have standardized the initial phase of the design process by formally deriving a functional model. In their research, they outline the methods to take customer needs information, create a black box, and then create a functional model using a common language representing the overall function in terms of sub-functions and verb-object forms (Strawbridge et al., 2002). Researchers at University of Missouri-Rolla (UMR), University of Texas at Austin (UTA), and National Institute of Standards and Technology (NIST) have developed a standard vocabulary of describing the basic functional blocks for the design of electro-mechanical products (Hirtz et al., 2002). This effort resulted in the inception of the functional basis language that spans the space of all functions and flows (Kurtoglu et al., 2005a).

Functional basis consists of two sets of terminology: 1) containing action verbs to describe function and 2) containing nouns to describe flow (Bohm et al., 2005). Function basis standard vocabulary was given by Little et al. (1997) and Stone (1997), and comprises of a set of functions and flows with clear definitions that are combined in verb – object form in order to describe a sub function. Generally these are broken down into eight function classes and three flow classes and are shown in Tables 2-1 and 2-2 respectively.

Table 2-1: Function classes and flow classes (Little et al., 2000)

Class	***Basic***	***Flow restricted***	***Synonyms***
Branch	Separate		Switch, Divide, Release, Detach, Disconnect, Disassemble, Subtract,
		Remove	Cut, Polish, Sand, Drill, Lathe
	Refine		Purify, Strain, Filter, Percolate, Clear
	Distribute		Diverge, Scatter, Disperse, *Diffuse*, Empty
	Dissipate		Absorb, Dampen, Dispel, *Diffuse*, Resist
Channel	Import		Input, Receive, *Allow*, Form Entrance, *Capture* Discharge, Eject, Dispose, Remove
	Export		
		Transfer	
		Transport	Lift, Move, Channel
		Transmit	Conduct, Transfer, Convey
	Guide		Direct, Straighten, Steer
		Translate	Oscillate
		Rotate	Turn, Spin
		Allow DOF	Constrain, Unlock
Connect	Couple		Join, Assemble, *Attach*
	Mix		Combine, Blend, Add, Pack, Coalesce
Control Magnitude	Actuate		Start, Initiate
	Regulate		Control, *Allow*, *Prevent*, Enable/Disable, Limit, Interrupt, Valve
	Change		Increase, Decrease, Amplify, Reduce, Magnify, Normalize, Multiply, Scale, Rectify, Adjust
	Form		Compact, Crush, Shape, Compress, Pierce
Convert	Convert		Transform, Liquefy, Solidify, Evaporate, Condense, Integrate, Differentiate, Process
Provision	Store		Contain, Collect, Reserve, *Capture*
	Supply		Fill, Provide, Replenish, Expose
	Extract		
Signal	Sense		Perceive, Recognize, Discern, Check, Locate
	Indicate		Mark
	Display		
	Measure		Calculate
Support	Stop		Insulate, Protect, *Prevent*, Shield, Inhibit
	Stabilize		Steady
	Secure		*Attach*, Mount, Lock, Fasten, Hold
	Position		Orient, Align, Locate

Table 2-2: Flow classes (Stone, 1997)

Class	***Basic***	***Sub-basic***
Material	Human	
	Gas	
	Liquid	
	Solid	
	Status	Auditory
Signal		Olfactory
		Tactile
		Taste
		Visual
	Control	
Class	***Basic***	***Sub-basic***
Energy	Human	
	Acoustic	
	Biological	
	Chemical	
	Electrical	
	Electromagnetic	Optical
		Solar
	Hydraulic	
	Magnetic	
	Mechanical	Rotational
		Translational
		Vibrational
	Pneumatic	
	Radioactive	
	Thermal	

Overall, functional basis of design has evolved into a widely adapted methodology for representing the product in terms of series of functions and flows. Functional Basis is intended to be broad enough to span the entire mechanical design space while not being repetitive (Bryant et al., 2005a). On the other hand, Japanese researchers have also explored a consistent language for describing the functionality of products, and relating it to the product behavior using behavioral actors. These behavioral actors are introduced in Section 2.2.1.

2.2.1 Behavioral Actors

Probing into the design literature reveals a tight coupling between function and behavior at the conceptual design stage. According to Szykman et al. (1999), the functional representation takes a top-down approach to representing a product. The overall function is described first, and then the behavior of each component is described in the context of this function (Chandrasekaran, 1994). According to Kumara et al. (1989), causal reasoning is important to delineate the structure-function and behavior of each component in an electro-mechanical product. Iwasaki et al. (1995, 1997) propose the Causal Functional Representation Language (CFRL) for representing device functions, with well-defined semantics in terms of behavior. Tomiyama et al. (1993) and Umeda et al. (1996) have proposed a Function-Behavior State (FBS) model for the representation of design. According to Zhang et al. (2002), function characterizes the abstract of a behavior, and the behavior characterizes the implementation of the function, which is supported by a structure. The authors have developed a functional reasoning strategy based on a heuristic search method, which allows the automatic reasoning out of physical behaviors from the desired functions. Kitamura and Mizoguchi (1998, 1999) have explored a consistent language for describing the functionality of products and relating it to the product behavior. Sasajima et al. (1996) propose a method and vocabulary for representing components captured from behavior and function view points in their Function and Behavior Representation Language (FBRL).

In another research by Deng et al. (1999), this functional modeling strategy was further strengthened by employing the behavioral scenario for not only describing the function, structure and behavioral process of the product, but also explicitly describing its environment. This enabled the researchers to view the product dynamically and describe the desired product comprehensively. According to Deng et al. (1999), the behavior scenario refers to a virtual scenario, which involves the desired product so that it can exhibit its intended behaviors and fulfill its required functions. In their research, the authors have proposed a Functional – Environmental – Behavior - Structure (FEBS) model. The functional information used in this model describes the intention or purpose of the product to be designed. The assumption is that the mechanical function is achieved through some action, and the material, flow and signals are referred to as the attributes of that action. The environment information provides the description of the environmental elements and their attributes, whereas the behavior information is used to describe the driving inputs as well as functional outputs of the overall behavior and each sub-behavior. The structural information describes the sub-assembly or components that are the behavior actors for the sub-behaviors. In their subsequent research work, Deng et al. (2000) found that the initial FEBS model was inflexible and computationally expensive due to the fixed top-down modeling strategy from initial functional decomposition to the causal behavioral process generation. The authors incorporated the Knowledge-Based Functional Design Automation (KBFDA) system that allowed the searching of a behavior whose functional output matched the desired function. If no matching behavior was found, the system would decompose the desired function into sub-functions using domain specific functional rules, and repeat the searching process. They also used a decision making method to rank and select the behaviors that meet the functional requirements. This approach was then refined in order to develop a new functional modeling framework called the B-FES modeling framework that implied a behavior driven modeling strategy. In this new approach, structure was implicitly associated with a behavior, and the behavior was represented in terms of functional requirement, behavior actor and functional output.

Overall, a behavior driven modeling approach is useful to automatically generate conceptual designs, when only the overall functional requirement and available environmental attributes are known to the designer. It is however believed that by incorporating functional basis of design along with behavior driven modeling strategy, the repeatability can be significantly improved. Repeatability of the process is found to be important particularly when the tool is to be computerized, and can be easily achieved by developing graph grammar rules. These rules are discussed in Section 2.2.2.

2.2.2 Graph Grammar Rules

Shape grammars are a set of shape rules that were originally used in architectural research by Stiny (1980). Yoshikawa and Ando (1987) utilized similar rules to propose a method for preserving the designer's intention in CAD systems as functional data in explicit form. These rules provide a flexible yet ideally structured approach to engineering design methods (Cagan, 2001), and are constructed by experienced designers to capture knowledge about a certain type of artifact (Kurtoglu et al., 2005b). An important off-shoot of the shape grammar research is graph grammar research, where graph grammars are also comprised of rules for transforming nodes and arcs within a graph (Kurtoglu et al., 2005b). These techniques are used to create a formal language for generating and updating complex designs from a simple initial specification. In the research by Kurtoglu et al. (2005a, 2005b) grammar rules are created in order to observe the common uses of components found in the design repository. It is found that by using these rules, repeatable conceptual designs can be automatically generated. Graph grammar rules that are used in the conceptual design of products are found to work in a similar way as the causal behavioral rules which were discussed in Section 2.2.1. We believe that since the graph grammar rules incorporate the functional basis, they can be standardized to a greater degree in comparison with the causal behavioral rules.

Overall, it can be concluded that a set of governing rules is imperative in order to generate computer aided conceptual designs. Section 2.3 discusses the shortcomings of CAD packages at the conceptual stage, and describes some existing design tools in the literature.

2.3 Computer-Aided Design Tools for Concept Generation

In the last two decades, Computer Aided Design (CAD) has undergone major breakthroughs and improvements such as the use of advanced graphic engines for solid modeling and automatic imposition of geometric constraints (Muogboh, 2003). As a result, modern CAD packages are well equipped with several additional capabilities like Feature-Based Design (FBD), Design For Assembly (DFA) and Design For Manufacturing (DFM). These functionalities however make the CAD packages perfectly suitable for product development only at the detailed design stage, with virtually no CAD software existing to support the design at the conceptual design stage. According to Tor et al. (2002), traditional CAD technology does not have the built-in intelligence to perform functional reasoning. Very little advancementd have been made to allow the modern CAD systems to capture and model product functionality during the conceptualization phase (Muogboh, 2003). According to Wang et al. (1995), it is possible to automate the entire design process if the conceptual design stage can be automated.

The inability of modern CAD tools to capture product functionality has led the designers to usually carry out the conceptual design phase without the help of any CAD tools (Tor et al., 2002). One of the main issues in developing a computer based functional design tool is the complexity involved in modeling the functional facets of design that represent the designer's intention or purpose (Zhang et al., 2003). The Behavior driven Functional – Environmental Structural (B-FES) modeling framework (Tor et al., 2002) has been utilized to create a functional design software system called FuncDesigner (Zhang et al., 2003). This software supports the design of mechanical products through functional reasoning steps including causal behavioral reasoning and functional decomposition. Kumara and Ham (1990) propose a method for capturing the conceptual design process by modeling the human associative memory as an Aritificial Neural Network.

Recently, research has been conducted to incorporate the CAD environment within the computer aided generation of conceptual designs. Liu et al. (1999) address the issue of developing physical embodiments from a set of spatial configurations by composing a method for transforming functional solutions to physical embodiments. Graph grammar rules and causal behavioral rules are observed to be widely accepted forms of rules for converting functional solutions to physical embodiments. These functional solutions, however, need to be stored and codified for re-usability. This necessity has given rise to the concept of design repositories, which are widely incorporated in current Computer Aided Conceptual Design (CACD) tools. Section 2.4 introduces design repositories, and discusses the importance of repositories for generating innovative conceptual designs.

2.4 Importance of Design Repositories for Concept Generation

Over the years, researchers have been trying to capture the best design practices that can be codified and recorded for reuse at a later stage by storing them in design repositories. A design repository is defined by Bohm et al. (2005) as a heterogeneous product design database in which various design solutions can be searched and re-used. These repositories have the capacity to store and retrieve design knowledge, and accordingly enable the designer to have easy access to a wide array of design solutions beyond their stored knowledge.

A web based repository has been created by University of Missouri – Rolla (UMR) and in collaboration with the University of Texas at Austin (UTA) and has been refined and populated over the years (Bohm and Stone, 2004). During the development of these design repositories, one important consideration is the need to have a standardized function and component taxonomy. According to Hirtz et al. (2002), functional basis infrastructure has evolved as a standardized design language that was developed in concert with NIST. The UMR repository includes descriptive product information such as functionality, bill of materials and Design Structure Matrix (DSM) on approximately 50 consumer products (Bryant et al., 2005a). Design tools such as Function Component Matrix (FCM) and DSM can be generated easily from single / multiple products and used in a variety of ways to enhance the design process (Bohm and Stone, 2003, 2004). When information related to product or component is added to these design repositories, several specific attributes are captured and stored in a relational database. Relationships can be drawn against different components based on the values of their attributes as entered by the user. The repository developed by Bohm et al. (2005) employs a morphological search tool that offers the designers with alternatives while they perform the activities related to conceptual design. In the research by Bohm et al. (2005), the authors proved the effectiveness of the morphological search tool by showing that it was capable of automatically replicating a substantial amount of solutions which otherwise would need to be developed manually using the different conceptual design techniques. However, their search algorithm lacked the capability of combining multiple sub-functions and components to be returned based on a single sub-function input. The research by Bohm et al. (2006) describes the detailed working of the UMR repository schema (version 2.0) and the development of several relational databases. The UMR design repository project was guided by the repository initiative at the NIST (Bohm et al., 2006). The research by Szykman (Szykman, 2002; Murdock et al., 1997) describes the design, development, architecture, functionality and interfaces of the NIST design repository.

These design repositories are being employed extensively in research involving the development of CACD tools. In the component basis of design (Kurtoglu et al., 2005b), the authors discuss two separate computer based conceptual designing tools that utilize the University of Missouri – Rolla (UMR) design repository for concept generation. The first research is a matrix based approach (Bryant et al., 2005a, 2005b), where the Functional Component Matrix (FCM) is extracted from the repository and determines how components are chosen to meet the functional specification of the design problem. By pre-multiplying FCM with the filter matrix, the FCM is reduced to a reasonable set of component combinations. The second research (Kurtoglu et al., 2005b) utilizes the function structure data (i.e., FCM) and graph grammar rules in order to generate Configuration Flow Graphs (CFG) as the output.

In a recent research at UMR, Vivek et al., (2006) document the initial efforts to develop a 3D visualization tool, which is a part of an effort to create an automated concept generation tool.

Since the accessibility of the concept generator via the web is one of the primary goals of this research, the authors utilize the low-memory Virtual Reality Modeling Language (VRML) instead of a traditional CAD package.

Overall, it is observed that several successful research attempts have been made to automatically generate conceptual designs. Some researchers have even attempted to incorporate the CAD tools within the existing conceptual design tools in order to enable the designer to better visualize the product design. However, no conceptual design tool was found to incorporate modularity within the existing product architecture. Section 2.5 seeks to enable the readers to have a better understanding of the modular product architecture, by providing a brief overview of modularity.

2.5 Modular Product Architecture

Implementation of modular product architecture is observed as one of the responses to the ever increasing challenges offered by mass customization and globalization, as it enables firms to manage a large variety of products (Gershenson et al., 2003; Mikkola and Gassmann, 2003; Kusiak, 2002). In the modular architecture, each functional element of the product is implemented by exactly one sub-assembly, with few interactions between the sub-assemblies (Ulrich and Eppinger, 1995). This enables a design change to be made easily to one sub-assembly without affecting the others (Gershenson et al., 2003; Kusiak, 2002). The following sections discuss the various aspects of product modularity in detail. Section 2.5.1 discusses the benefits offered by modularity.

2.5.1. Benefits of Modularity

Several benefits are offered to a manufacturing firm by implementing modularity in the product architecture. Modularity permits components to be produced separately and used interchangeably in different formats without compromising system integrity (Mikkola and Gassmann, 2003). Modularity allows customers to mix and match elements to come up with a final product that suits their tastes and needs (Baldwin and Clark, 1997). It allows for flexible designs over the life of a product, which can respond to changes in functional requirements over time (Tate et al., 1998). These changes could be either changes in customer requirements, or the need for products to perform different functions at different times. Antonio et al. (2004) state that modularity enables firms to distribute their products more broadly by modularizing, thereby customizing the products to meet government regulations of different geographical regions. Modularity delays the tasks of product differentiation until the product has moved through the distribution channels and is ready to be distributed to clients, therefore reducing inventory costs (Antonio et al., 2004; Feitzinger and Lee, 1997). Moreover, it increases ease of reuse, recycling and disposal as it allows the grouping of components into easily detachable modules (Zhang and Gershenson, 2003). Reuse of existing modules reduces the cost of product development (Krikke et al., 2004). Modular product design decouples developmental tasks and enables the modules to be developed concurrently in order to reduce development time (Ulrich and Tung, 1991). It is also easy to update the modular products as they have functional modules. Despite its benefits, modular designs have certain drawbacks as well and Section 2.5.2 highlights these issues.

2.5.2. Issues Related to Modularity

There are certain issues involved with modular products as well. Firstly, modular products need to be designed with redundant physical components and less function sharing, so that they are compatible across other products, which might result in increased variable costs (Ulrich and Tung, 1991). Another drawback is the trade-off between standardization and modularization, while selecting a platform (Jose and Tollenaere, 2005). Manufacturing companies can optimize as well as customize products by the use of different modules, thereby allowing them more diversity of products, but resulting in higher costs. On the other hand, by increasing the number of common modules, these companies reduce the number of possible combinations of their products, but it allows them more cost savings. Figure 2-1 graphically demonstrates this trade-off.

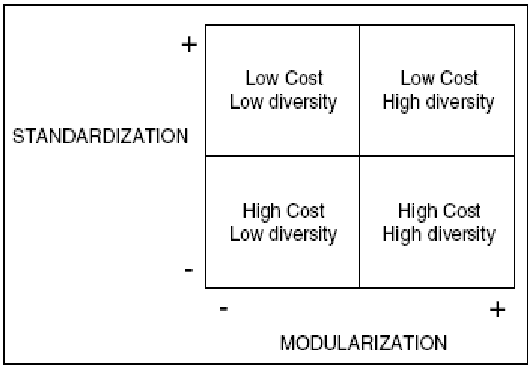


Figure 2-1: Modular versus standard parts (Adopted from Jose and Tollenaere, 2005)

Whitney (2004) claims that modularity is not always desirable, especially in the case of high powered mechanical products. According to the author, more modular the product becomes, the more it is likely to be larger, heavier, less energy efficient and harder to control its side effects. The following section discusses the taxonomy of the modular architecture.

2.5.3. Taxonomy of Modular Product Architecture

A considerable amount of research has been conducted in order to modularize the product architecture and several distinct taxonomies are found in the design literature. The first classification, published in 1984, distinguished between basic, auxiliary, special and adaptive modules, based on the importance of functions embedded within a given module to the working of a larger system (Pahl and Beitz, 1984; Salvador et al., 2004). Years later, a more detailed classification was developed by Ulrich and Tung (1991), defining five categories of modularity based on the kind of pairing between components, for generating variants of the product. These were: 1) component swapping modularity, 2) component-sharing modularity, 3) fabricate-to-fit modularity, 4) bus modularity, and 5) sectional modularity. In component-swapping modularity, different product variants within the same product family are obtained by pairing different components with the same basic product. In component-sharing modularity, the modular product is matched with different basic modules, creating different product variants of different product families. In fabricate-to-fit modularity, a product includes a component with some continually varying feature. In the case of bus modularity, a product component has two or more interfaces that can be matched with any selection of components from a set of component types. Finally, sectional modularity allows the linking of components chosen from a standard set of component types, to be connected in any arbitrary way, as long as the components are attached to each other only at their interfaces. In this taxonomy, the classification criteria could be improved with increased clarity. Figure 2-2 demonstrates the different types of modularity, and is adopted from http://www.icaen.uiowa.edu/~coneng/lectures (viewed on: 8th November, 2008).

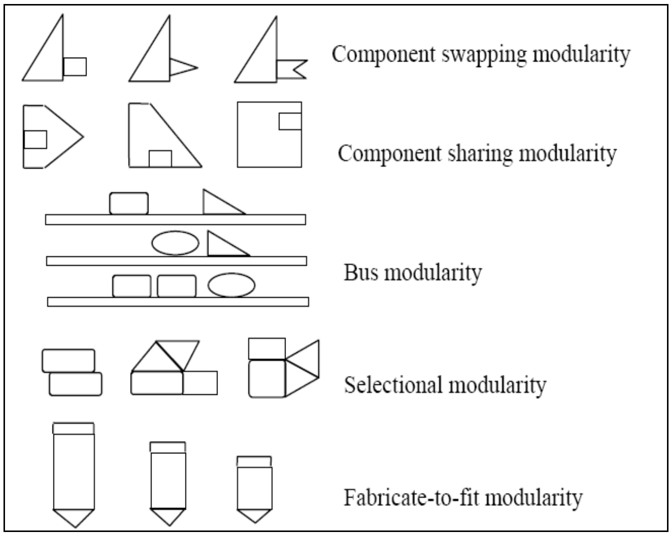

Component swapping modularity
Component sharing modularity
Bus modularity
Selectional modularity
Fabricate-to-fit modularity

Figure 2-2: Different types of modularity.

Ulrich (1995) proposed a modification to his earlier classification, by defining the degree of standardization of the interface between components as the classification criterion. Based on this, a distinction was found between bus and sectional modularity. However, fabricate-to-fit modularity no longer seemed a distinctive approach, but in actuality, it has important implications for manufacturing as well as marketing. Component-sharing modularity is also portrayed as an accidental outcome of product modularization; instead, it is one of the basic reasons for which the firms pursue product modularization. Therefore this classification missed some important manufacturing implications of modularity (Salvador et al., 2004).

Allen and Carlson – Skalak (1998) categorized the modular design methods into two main categories: function-based methods and matrix-based methods. The function based methods (Pahl and Beitz, 1998; Stone et al., 2000) include a formal function model and require some intrinsic knowledge on directing the functional module decomposition. Matrix based methods (Zhang and Gershenson, 2003; Huang and Kusiak, 1998; Sosale et al., 1997; Coulter et al., 1998) seek to achieve the maximum modularity by redesigning or reconfiguring the product structure matrix, which represents the product architecture (Guo and Gershenson, 2004). Section 2.5.4 provides a detailed comparison between the modular product architecture and the integral product architecture.

2.5.4. Modular Architecture versus Integral Architecture

Degree of integrality or modularity is a key feature of any product architecture, and the major difference lies in the mapping between the functional models of the product and the physical components (Sosa et al., 2003). Mikkola and Gassmann (2003) state that although most of the studies on modularity are qualitative and exploratory in nature, they offer limited insights to how the firms measure the degree of modularity embedded in the product architecture. They have developed a modularity function, which is used to measure the level of modularity as a function of components, interfaces, degree of coupling and substitutability. Sosa et al. (2003) introduced a method to identify whether the system is modular or integral based on the interaction of components in the system DSM. The authors define modular systems to be one, whose design interface with other systems is clustered among only a few physically adjacent systems. Integrative systems, on the other hand, are defined as those whose design interfaces span most of the systems which together make up the product. The authors also highlight the importance of identifying the modularity or the integrality of the system, as the system architecture affects the design team interactions. However, the authors do not say when a modular system is more appropriate than an integral system. Hölttä et al. (2005) showed that if the technical drivers like power consumption, or weight are the main drivers of a design, then an integral system provides a more suitable architecture than a modular system. On the other hand, a modular system is preferred when the business drivers such as commonality and flexibility are the main concerns during the design. The authors quantified the degree of product modularity based on its internal connectivity structure which can be represented using the DSM. Guo and Gershenson (2004) developed a metric to measure the product modularity using the component-component connectivity matrix. It is based on the definition of the modularity where the module is tightly connected within the module and loosely connected to the rest of the system, and is useful when module boundaries are unambiguous (Hölttä et al., 2005). Section 2.5.5 discusses modularity at the detailed stage of electro-mechanical product design.

2.5.5. Modularity at the Detailed Design Stage

Gupta and Okudan (2007) conducted an extensive review of existing modularity techniques, and found that most research involving product modularization was carried out at the detailed stage, with very few methodologies focusing on the conceptual design stage. Some detailed design modularizing methodologies are discussed in subsequent paragraphs.

Kreng and Lee (2004) proposed a matrix based method, which requires three matrices to be constructed: 1) function interaction matrix, 2) physical interaction matrix, and 3) component–component correlation matrix. The modular driver is then identified, which enables the inter-functional teams to indicate how much the driver affects the components, and accordingly a relationship matrix is built between modular drivers and components. A non-linear programming model is then developed, and optimization is done using a grouping genetic algorithm to search for the optimal or near-optimal modular design. The authors have adopted a heuristic approach in order to deal with the challenging issue of combinatorial explosion effectively. The proposed heuristic approach also develops a suitable cross-over method for clustering objects.

Salhieh and Kamrani (1999) proposed a methodology for identifying components being developed in parallel by decomposing the product into its basic functional and physical elements. Once the basic functional and physical elements are achieved, system level specifications are identified, which are one-to-one relationships between components with respect to their functional and physical characteristics. This is followed by measuring the degree of association between components using a similarity index between components. Components having high degree of association are grouped together as modules, and an optimization model is used to identify the independent modules that can be designed simultaneously.

Hsiao and Liu (2005) applied the Interpretive Structural Modeling (ISM) technique to identify the hierarchical structure of component interactions within a module. Their proposed methodology begins with product market planning, which classifies the various requirements of different markets. The next stage involves QFD analysis, to find the relative importance of each component towards different markets. This is followed by the cluster retrieval method, which is used to integrate components and treat them as a single entity. Hierarchical graphs are formed by identifying the set of components in the matrix that cannot reach or be reached by other components present outside the set. This set is then removed from the matrix and this process is repeated for the remaining matrix until a unique set of nodes is obtained.

Huang and Kusiak (1998) developed a matrix based methodology in order to interpret three different types of modularity: concept swapping, concept sharing and bus modularity. Their methodology is called Decomposition Approach (DA), and involves the generation of suitability and interaction matrices. Whereas the suitability matrix represents the suitability for components for inclusion in a module, interaction matrix represents the interaction between the components. A decomposition approach is then followed to generate modularity matrix by triangularization of interaction matrix (Kusiak et al., 1994) and the rearrangement of suitability matrix such that the sequence of rows and columns in both the matrices remains the same. Modularity matrix is then analyzed and classified based on the aforementioned three different kinds of modularization, and results are generated. Section 2.5.6 discusses the industrial implementation of modular architecture.

2.5.6. Industrial Implementation of Modularity

Various industries have been implementing modularity in their product architectures for over two decades. However, despite the numerous applications, science base of modularity is found to be almost non-existent, especially in engineering design (Kusiak, 2002). According to Kusiak (2002), the gain of corporate profits due to improvements of the quality of generated modules with formal approaches would be tremendous. Table 2-3 summarizes relevant examples, where manufacturing firms have benefited significantly by implementing modularity. Section 2.5.7 provides a detailed comparison of different modularization methodologies as obtained from the design literature.

Table 2-3: Industrial implementation of modularity (Gupta and Okudan, 2007)

Company	Product	Benefits achieved by modularity	References
Nippondenso Co.	Panel Meter	Significant reduction in the number of parts through redesign of the old panel meter into six standard modules.	Aoki (1980)
Sony	Walkman	By mixing and matching modular components of a few basic product designs, over 160 creations were made possible.	Sanchez (1996)
Sony	Handy Cam	Several upgraded models of video cameras were leveraged from an initial system design by successive introduction of improved modular components.	Sanchez (1996)
Volkswagen	VW, Audi, Skoda, and Seat.	Claims to save $1.7 billion annually on development and production costs through platform and component commonality by sharing between its four major brands.	Bremmer (1999)
Ford Motor Co.		Ford has a similar shared platform ambition within its new Generic Architecture Process program, and has similar expectations of large monetary savings in development and production costs.	Bremmer (1999)
Mercedes		Able to be more responsive to orders, without owning inventories. This is due to the purchasing of modules only when the car is being assembled. By using delayed product differentiation, Mercedes plans to supply cars at a shorter lead-time.	Gershenson et al. (2003)

2.5.7. Comparison of Different Modularization Methodologies

It is observed that mostly all modular design methodologies that exist in the design literature have very different criteria on the rationale for modularization. Each individual methodology is important in developing an understanding of product modularity modeling; however, no one can optimize perfectly according to all the criteria (Hölttä et al., 2005). As a result, researchers have tried to compare different modularity methods in an attempt to find out the best modularization methodology which on the whole suits the given criterion.

Hölttä and Salonen (2003) compared three modularity methods (Stone et al., 2000; Pilmer and Eppinger, 1994; Ericsson and Erixon, 1999). The authors tested the three methods on four products: an intra-oral camera, an electronic pipette and two medical injector heads. The main modularization criteria for the Function Heuristic Method (FHM) developed by Stone et al. (2000) are functionality and module interfaces. Pimmler and Eppinger (1994) showed that spatial, energy, information and material interactions of components or functions are represented in the DSM. In this approach, the components or functions are mapped against each other with their interactions marked in the matrix; a clustering algorithm is then applied to group the components, which are possible module candidates. The Modular Function Deployment (MFD) approach (Ericsson and Erixon, 1999) is more management oriented rather than engineering oriented. Hölttä and Salonen (2003) obtained different results from each method, even after providing identical inputs. The authors found that MFD is best suited to define design variants and to decide on make-buy decisions; DSM technique is best suited for modularizing a complex system with many interactions for a person to handle. In order to modularize a product family, FHM approach was found to be the most reasonable. The authors also found DSM technique to be the most repeatable and it could be easily converted to a computerized algorithm. One of the limitations of the research was the lack of integration of all the methods with a quantitative modularity measure. This made it difficult to reach an objective conclusion for the best method for modularity improvement (Guo and Gershenson, 2004).

Guo and Gershenson (2004) compared four different modularity methods (Gershenson et al., 2003; Stone et al., 2000; Sosale et al., 1997; Coulter et al., 1998) based on the hypothesis that for the same initial design and same modularity measure, the design methodology that generated more modular products more efficiently, was the best design. The authors compared the modularity methods by using four products: a Kodak single-use camera, a Conair supermax hairdryer, an Adhesive Tech mini glue gun and a Regent halogen clamp lamp. Based on the assumptions that 1) the modularity matrix was the full representation of the product's modular structure and 2) that redesign of the matrix yielded the same optimal modular product architecture as the redesign of the modular product architecture itself, the authors were able to compare between function-based and matrix-based methods. The authors found that Gershenson et al.'s modular approach (Gershenson et al., 2003), which was aimed at reducing the total life cycle cost by designing a more modular product, was the most reliable modular product design method. This modular approach also showed highest modularity improvement; statistically fastest redesign and stable results. Stone et al.'s FHM approach (Stone et al., 2000), which consisted of the identification of modules from a function structure with flows of energy, materials and signals, generated better results for mini glue gun and lamp. Coulter et al.'s approach (Coulter et al., 1998), whose main goal for modular product design was improved product reliability with respect to physical structure and material reliability. This method was also found to be a stable redesign method. The modular design objective found in Sosale et al. (1999) was the weighted average of all the life-cycle engineering objectives and this approach generated better results for camera and hair dryer. Comparison done by Guo and Gershenson (2004) was mainly aimed at clarifying the fundamental relationship between product modularity and product life cycle costs.

Gupta and Okudan (2008a) compared the FHM by Stone et al. (2000), B-FES approach by Zhang et al. (2006) and DA by Huang and Kusiak (1998). All three methodologies were tested on the conceptual design using Oral-B Vitality^{TM} electronic toothbrush (www.oral-b.com) and different number of modules were obtained by each methodology. Under the hypothesis that the modularizing methodology offering highest ease of assembly and clustering components into modules to effectively meet future customer needs is the "best" methodology, DA was found to be the best approach. Section 2.5.8 discusses the modularization methodologies focusing at the conceptual design stage.

2.5.8. Modularization Methodologies at the Conceptual Design Stage

Most of the focus on product modularity implementation is observed at the detailed design stages instead of the conceptual design stage (Gupta and Okudan, 2007). According to Nepal et al. (2005), prior modularization methodologies lacked in applying scientific approach to capture the attributes of prospective modules at the conceptual phase. Kusiak (2002) points out that ideally, the model for the creation of a module should be general so that designers can form these modules during any phase of the product life cycle. The author, however, argues that though it is desirable to form modules early in the design process, the information necessary to form them might not yet be available, causing the early generated modules to fail in meeting the constraints that become apparent later in the design process. Hence, information availability is crucial during module identification. The type and amount of information available warrants the classification of modularity (Pahl and Beitz, 1998; Kusiak, 2002).

According to Nepal et al. (2005), fuzzy logic can be widely used in handling the uncertainties in the product design at the conceptual stage. The authors have proposed a formal integrated fuzzy-logic based approach for optimizing the manufacturing and reusability related costs of prospective modules, while modularizing the product architecture. Tsai and Wang (1999) proposed a methodology that utilizes fuzzy cluster identification to select modules for parallel designing. Mattson and Magleby (2001) discuss concept selection techniques for managing modular product development during the conceptual stage. According to Zhang et al. (2006), the functional modeling approach has become a leading technique in modeling a design and related requirements from its functional aspects in order to allow reasoning about its function for various activities, particularly at the conceptual design stage. They proposed a fast method for generating new concept variants during the conceptual design of a family of products. Stone et al.'s FHM approach (Stone et al., 2000) enables modular designs to be executed early in the product development process. Huang and Kusiak's DA allows for the optimal forming of modules even at the conceptual stage, where there is insufficiency of available information (Huang and Kusiak, 1998).

Overall, although a few research methodologies that focus on the implementation of modularity at the conceptual stage were found; no widely adopted measure to quantitatively compare these approaches was found in the design literature. Besides, none of the existing modularization methodologies were found to incorporate manufacturing and assembly considerations during the development of the modular product architecture. It is also observed that several conceptual design combinations are generated by utilizing the computer-aided concept generation tools mentioned in Section 2.4. However, in order to select the best conceptual design technique, an effective and unbiased filtering tool is required. In the proposed framework, ease of manufacturing and assembly is chosen as the primary filtering criterion. Section 2.6 provides a brief overview of the Design for Manufacture and Assembly (DFMA) in order to enable the readers to easily comprehend the proposed research framework.

2.6 Design for Manufacture and Assembly

In the design literature, researchers have stressed on the importance of taking a careful account of the manufacturing and assembly problems in the early phases of product design (Boothroyd, 1994). This is because the traditional "overall the wall" approach resulted in several manufacturing and assembly problems leading to requests for design change and a considerable delay in the final product release. Over the years, a rising number of companies started to address the DFMA principles in order to improve the manufacturing situation by changing the product design (Fabricius, 1994). DFMA is also seen as a way to increase the productivity significantly without any major investment.

Predicting the life cycle impact of life design decisions in the early design process is critical to the success of any product. Hermann et al. (2004) state that ignoring downstream life cycle issues results in the generation of poor designs, which may cause unforeseen problems and excessive costs. On the other hand, accurate predictions of the life cycle needs early in the design process not only reduces the number of redesign iterations, time to market, development and manufacturing costs but also improves customer's experience (Hermann et al., 2004). Therefore, it is imperative that changes and interventions with regards to manufacturing aspects should take place at the design stage in order to result in optimum manufacturing of the product (Dumitrescu and Szecsi, 2002). In the research by Fabricius (1994), the author proposed a seven step procedure consisting of guidelines to improve the manufacturability of the product. This procedure was applied to several pilot projects in big and small companies, which resulted in the generation of a variety of electro-mechanical as well as mechanical products.

Design for Assembly (DFA) is closely linked to DFMA and the significant benefits obtained by the use of DFA were not realized until systematic analysis tools became available around 1980 (Boothroyd, 1994). Apart from seeking to simplify the product structure, DFA seeks to reduce the total number of parts, thereby reducing the total cost of the parts. Simpson et al. (1995) state that although the DFA analysis by Boothroyd and Dewhurst (1989) is useful late in the design embodiment phase, it is difficult to implement it at the early conceptual and embodiment phases where the information defining the product's physical form is vague and incomplete. In the research by Simpson et al. (1995), the authors transformed the eight governing principles of Boothroyd and Dewhurst's DFA analysis into abstracted DFA principles for use within conceptual and embodiment phases.

A quantitative DFA index is proposed by Hsu et al. (1998), which can be associated with each of the components present in an electro-mechanical product. The DFA index is calculated based on the ranking system developed by Rampersad (1994). The quantification is also partly based on the classification system of Boothroyd and Dewhurst (1989). It is believed that this DFA index would serve as an effective filtering criterion for the proposed framework in order to select the best conceptual design. Modularization of the best conceptual design by considering only the degree of interaction between components is not enough, and it is desired to also incorporate variety issues as one of the modularization criteria. Section 2.7 provides a brief overview of the existing Design for Variety issues as found in the design literature.

2.7 Design for Variety

It is observed that DFMA has significantly improved the quality and profitability of US manufacturing companies over the past two decades (Martin and Ishii, 1996). However, researchers state that despite the improvements in product quality, applying DFMA to single products is not enough. Instead, companies seek to optimize the costs of producing all the varieties of products together, as they see variety issues as the key to profitability (Martin and Ishii, 1996). The authors propose a systematic design methodology leading to a wide coverage of customer preferences, while reducing manufacturing costs, shortening production cycle as well as enhancing product line flexibility. The authors introduced the Design for Variety (DFV) strategy, which seeks to estimate the cost of providing variety at the earlier stages of design.

In a later research work by Martin and Ishii (1997), the authors improved their previous set of indices capturing the amount of variety within a design, by correlating them with indirect costs. As a result, they developed three indices: 1) Commonality Index (CI) that measures number of unique parts, 2) Differentiation Index (DI) that measures where product is differentiated and value added, and 3) Setup Index (SI), measuring the product switch-over costs. Apart from the quantitative tools, the authors also developed qualitative tools like process sequence graphs, which may help industries to graphically understand how variety affects their manufacturing operations (Martin and Ishii, 1997).

In subsequent research, Martin and Ishii (2000) developed product platform architecture to obtain reduced design effort and time-to-market for future generations of the product. The authors proposed the use of two indices to measure the product's architecture: 1) Generational Variety Index (GVI) that measured the amount of redesign required for future designs of the product, 2) Coupling Index (CI), measuring the coupling among future components. In later work, Martin and Ishii (2002) improved the DFV structured methodology to aid the development of a product platform architecture that incorporates standardization and modularization to reduce future design costs and efforts. Several other research efforts have also explored issues dealing with the strategic benefits of developing product platforms and management of product families.

Farrel and Simpson (2001) discussed how the strategic incorporation of product platforms into the design process can leverage individually customized products using the example of the design of a yoke cross-section. Hsiao and Liu (2005) state that main advantage of developing a product family is that it enables a company to offer two or more products that are highly differentiated yet share a substantial fraction of their components. Gonzales-Zugasti and Otto (2000) define two ways to create product families using a platform-based strategy: 1) integral platform, where all the products in the family share a single and monolithic part of the product, and 2) modular platform, which uses a set of modules as a product platform to be used across the product family. This common platform along with other variant-specific modules is used to generate the product family variants.

Modular platform strategy is found to have several inherent advantages. Rai and Allada (2003) propose that by reducing the number of module types and instances for each module type required to produce the entire product family, efforts required to design, produce, distribute, operate and maintain the module instances throughout the life cycle of the product family can be lowered. Jiao and Tseng (1999) assert that a Product Family Architecture (PFA) performs as an integration platform for extending the traditional boundaries of product design to encompass a larger scope spanning from sales and marketing to distribution and services. The authors developed a PFA model by grouping similar products into families based on their product topology along with functional requirements, and then established an optimal PFA based on this grouping.

Fujita et al. (1999) reviewed product variety based on multiple considerations like views of customer’s needs, functions, manufacturing modules and hierarchical representation of systems. Using an integer programming formulation, the authors formulated the product variety design and proposed an optimization approach to designing modular products from existing modules. The research by Fujita et al. (1999) is based on determining the optimum number of configurations of those modules, which have already been designed. On the other hand, Gonzales-Zugasti and Otto (2000) developed a method that helps designers in determining the right combination of unique and common modules that are required for each product. Hence, both existing and new modules can be accommodated using this research methodology. Gonzales-Zugasti and Otto (2000) defined a module matrix to represent the mix of module instances used in a product family. Using a genetic algorithm, the authors found an optimum degree of commonality, which also included the optimum configuration of the common modules.

Overall, incorporating variety issues along with modularity is important for manufacturing firms in gaining a competitive edge against their peers. In the proposed research framework, it is desired that future technological changes along with changes in customer preferences be incorporated as a modularization criterion. This would enable all the components that are more likely to change in the future to be clustered into a single module. As a result, manufacturing firms would only need to replace the specific module of the product in order to satisfy the future needs. In the next chapter, a detailed and step-by-step explanation of the proposed research framework is provided.

Chapter 3

Proposed Framework for Generating Conceptual Designs

In this chapter, a detailed step-by-step explanation of the proposed research framework is discussed. The current software framework has been developed using Java Swing within the NetBeans IDE 5.5.1 programming environment. MySQL database is used for storing all the various database tables within the design repository, and Java Database Connectivity (JDBC) connection is used to open the MySQL tables within the Java environment.

Foremost step in the proposed framework as shown in Figure 1-3, is the assessment of the customer needs. Obtaining the customer needs is imperative, as it enables the designers to focus the conceptual design more on the specific needs and tastes of the consumers. Customer needs are generally obtained by conducting surveys and enquiring potential consumers about their likes, dislikes and expectations from similar products in the market. By satisfying all the customer needs, the manufacturing company is likely to earn huge profits by selling the product since these consumers will be the end users of their product. However, fulfillment of all the customer needs may not be technically feasible under the constraints of time and cost. Hence, the customer needs are ranked based on their importance using different tools, e.g. Borda Count (Borda, 1782), and this helps in identifying the important customer needs that should be fulfilled. Quality Function Deployment (QFD) (Akao, 1990) is an important tool to translate the customers desires into engineering metrics or functions which can then be incorporated into the product architecture. The proposed framework utilizes a two-phase QFD tool (Martin and Ishii, 2000) for generation of the DFV index, which is explained in the later sections. Once the customer needs are obtained, the designers can develop a black box model, which enables them to list the input / output functions and flows that make up the product. Figure 3-1 shows a general black box model for an electro-mechanical product.

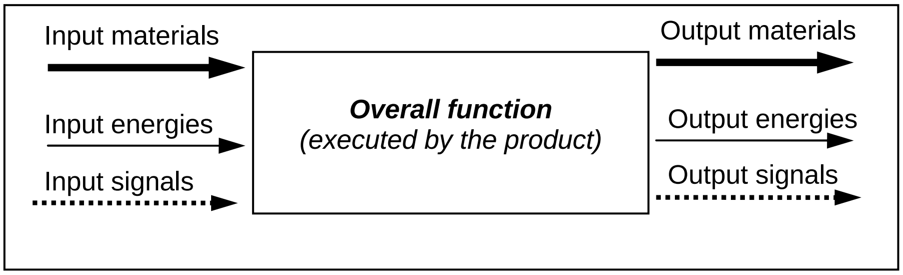


Figure 3-1: Black box model of an electro-mechanical product

In the black-box model, the overall function to be performed is mentioned inside the boundary of the box, while input energies, materials and signal flows are identified to the left of the box. The output energy, material and signal flows are indicated to the right of the box, which identifies the remainder flows obtained after the overall function has been executed.

The next step involves the generation of the Energy Material Signal (EMS) functional model. The EMS functional model is obtained by decomposing the overall function into simpler sub-functions and flows, which are generally described in a verb-object form. These sub-functions and flows are obtained from a standard set of vocabulary referred to as functional basis (Stone, 1997; Little et al., 1997). Section 2.2 provides an overview of the functional basis, while Tables 2-1 and 2-2 highlight the sub-functions and flows that make up the functional basis set. Figure 3-2 highlights a general EMS model comprising of sub-functions and flow of materials, signals and energies.

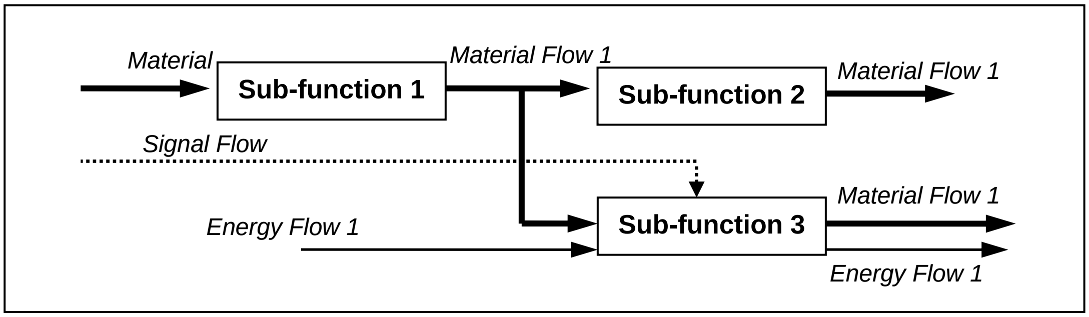
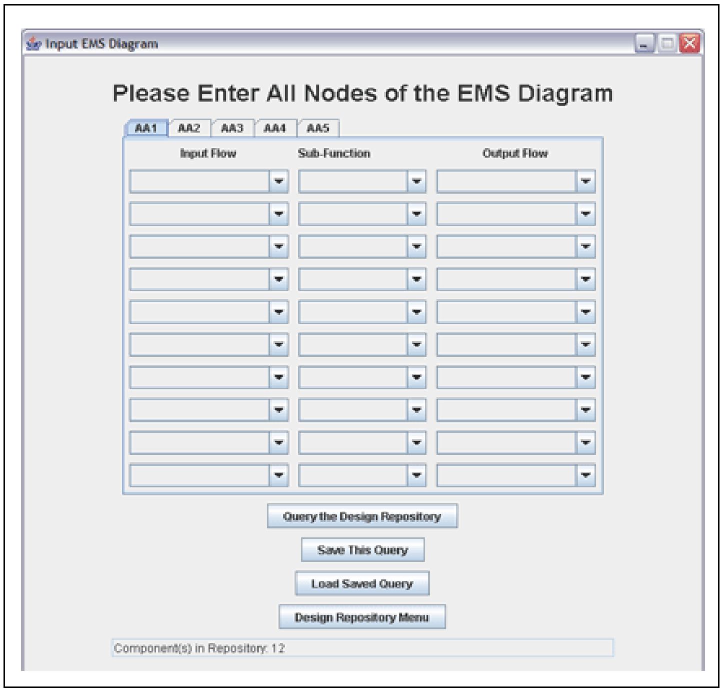


Figure 3-2: EMS model of an electro-mechanical product.

The EMS functional model needs to be input into the proposed software framework in order to generate the conceptual designs by querying the design repository. As a result, a Graphical User Interface (GUI) is developed, inspired from Kurtoglu et al. (2005a, 2005b), using which the functional model can be easily input into the proposed software framework. Figure 3-3 shows the GUI screen.



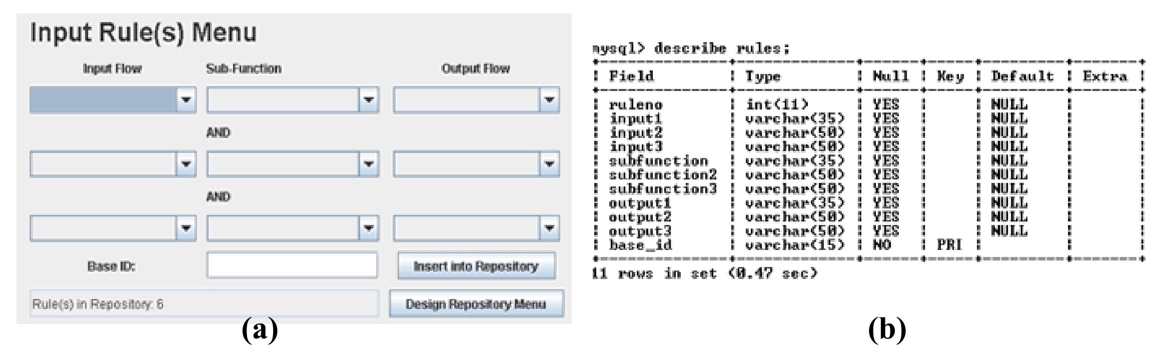
Component(s) in Repository: 12

Figure 3-3: GUI screen for inputting the EMS functional model

While inputting the functional model, each sub-function can be considered as a node, with an input flow and an output flow. These input/output flows can be energy, material or signals, and are obtained from the standard functional-basis vocabulary. Currently, the GUI has the capacity of accommodating up to 50 nodes, as it has 5 tabs (i.e., AA1 – AA5) with each tab accommodating up to 10 nodes. The software also allows the designer to save the query and load a previously saved query. After the completed EMS model is input into the framework, the design repository is queried in order to generate all possible conceptual design variants that satisfy the overall functional model. In order to automatically generate the conceptual designs, each node of the EMS model needs to be compared with all the nodes of each rule, in order to obtain a direct match. Once the rule gets triggered, all the components associated with that rule are retrieved, resulting in the generation of multiple conceptual designs satisfying the same overall function. The design repository therefore needs to be populated with two major databases: one for storing the different graph grammar rules, and the other for storing the component specifications (e.g., DFA index, image and the rule associated with that component). Since multiple components satisfy a single rule, several conceptual designs satisfying a given function may be generated. Section 3.1 discusses the structure of the design repository.

3.1 Design Repository

As defined previously in Section 2.4, a design repository is a heterogeneous product design database in which various design solutions can be searched and re-used for future needs. In the proposed framework, the design repository comprises of three major data tables within the database: ‘rules’, ‘DFA’ and ‘interaction’. The ‘rules’ table is used to store input/output flows and sub-functions corresponding to each user-defined rule. Each rule can have at most three nodes of sub-functions and input/output flows associated with it, and needs to have a unique associated base id (Figure 3-4). For the proposed framework, graph grammar rules are adapted from the research by Kurtoglu et al. (2005a). Some of these rules are available at the research team's website: http://www.me.utexas.edu/~adl/cfg_grammar.htm (viewed on: 8th November, 2008).

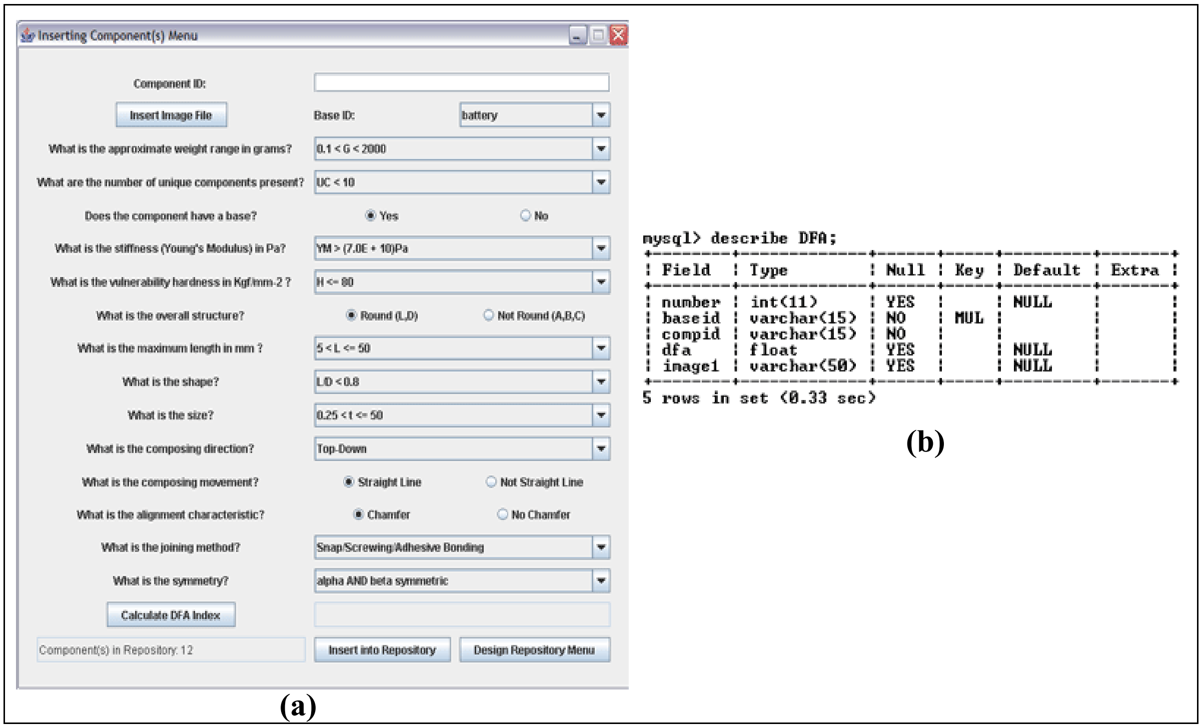


```
mysql> describe rules;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ruleno       | int(11)     | YES  |     | NULL    |       |
| input1       | varchar(35) | YES  |     | NULL    |       |
| input2       | varchar(50) | YES  |     | NULL    |       |
| input3       | varchar(50) | YES  |     | NULL    |       |
| subfunction  | varchar(35) | YES  |     | NULL    |       |
| subfunction2 | varchar(50) | YES  |     | NULL    |       |
| subfunction3 | varchar(50) | YES  |     | NULL    |       |
| output1      | varchar(35) | YES  |     | NULL    |       |
| output2      | varchar(50) | YES  |     | NULL    |       |
| output3      | varchar(50) | YES  |     | NULL    |       |
| base_id      | varchar(15) | NO   | PRI |         |       |
+--------------+-------------+------+-----+---------+-------+
11 rows in set (0.47 sec)
```

Figure 3-4: (a) GUI for inputting the rules into repository (b) MySQL table

The second table in the database is the 'DFA' table, which stores the DFA index, image file of the component and also has a unique component ID associated with each component stored in the design repository. For the generation of the conceptual design, each component has to be associated with a rule. Hence, for each component ID, a corresponding base ID has to be selected, which is the foreign key to the rules table. Figure 3-5 shows the GUI and structure of inputting the components in the design repository.

The third table within the database is 'interaction' table, which stores the interaction amongst the different rules stored within the repository. The designer needs to determine whether there is any interaction between the rules and also needs to indicate the direction of interaction. For example, if there is an interaction between rule1 and rule2, it is denoted by 1, otherwise 0. The interaction table is useful for modularization of the conceptual designs based on the Decomposition Approach (DA) by Huang and Kusiak (1998). The interaction matrix is developed while the designer examines the dissected components during population of the repository.



Inserting Component(s) Menu

Component ID:

Insert Image File — Base ID: battery

What is the approximate weight range in grams? 0.1 < G < 2000

What are the number of unique components present? UC < 10

Does the component have a base? ● Yes ○ No

What is the stiffness (Young's Modulus) in Pa? YM > (7.0E + 10)Pa

What is the vulnerability hardness in Kgf/mm-2 ? H <= 80

What is the overall structure? ● Round (L,D) ○ Not Round (A,B,C)

What is the maximum length in mm ? 5 < L <= 50

What is the shape? L/D < 0.8

What is the size? 0.25 < t <= 50

What is the composing direction? Top-Down

What is the composing movement? ● Straight Line ○ Not Straight Line

What is the alignment characteristic? ● Chamfer ○ No Chamfer

What is the joining method? Snap/Screwing/Adhesive Bonding

What is the symmetry? alpha AND beta symmetric

Calculate DFA Index

Component(s) in Repository: 12

Insert into Repository — Design Repository Menu

(a)

```
mysql> describe DFA;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| number | int(11)     | YES  |     | NULL    |       |
| baseid | varchar(15) | NO   | MUL |         |       |
| compid | varchar(15) | NO   |     |         |       |
| dfa    | float       | YES  |     | NULL    |       |
| image1 | varchar(50) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
5 rows in set (0.33 sec)
```

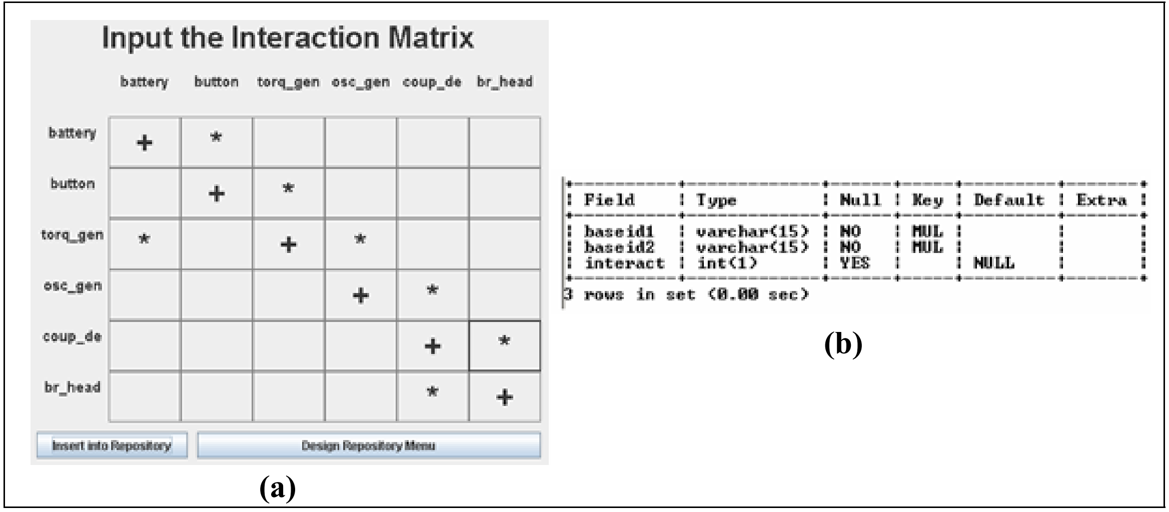
(b)

Figure 3-5: (a) GUI to input components into the repository (b)MySQL table

The interactions can be either flow of electricity between the components, or the force exerted by one component towards the other. The direction of the interaction is important as some interactions are unidirectional, while others are bi-directional. Accordingly, they are implemented in the decomposition algorithm. Figure 3-6 shows the GUI of the interaction table, which can be accessed from the design repository. As mentioned previously, there are several cases where more than one component is associated with the same rule. This is particularly useful when the rules are compared with the EMS diagram in order to result in the generation of multiple solutions satisfying the same set of functions. However, to select the best solution, these different component combinations are ranked based on a certain evaluation criterion.

In the proposed framework, DFA index is chosen for ranking the different conceptual designs, since it is desired to have a minimum assembly time while assembling the components.

Modularization of the product architecture would further cluster the components together into bigger sub-assemblies or modules, thereby further reducing the assembly time. However, it is our hypothesis that a good modularization methodology should cluster components to form modules such that components having higher interactions with each other, and also which are highly likely to change based on future customer needs, are clustered into single modules. DFV is therefore utilized as the secondary filtering criteria. DFA index calculation is explained in Section 3.2.



Input the Interaction Matrix

	battery	button	torq_gen	osc_gen	coup_de	br_head
battery	+	*				
button		+	*			
torq_gen	*		+	*		
osc_gen				+	*	
coup_de					+	*
br_head					*	+

```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| baseid1   | varchar(15) | NO   | MUL |         |       |
| baseid2   | varchar(15) | NO   | MUL |         |       |
| interact  | int(1)      | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure 3-6: (a) GUI to input the interaction matrix into repository (b)MySQL table

3.2 DFA Index Calculation

In order to generate feasible conceptual designs, which offer the ease of assembly, a DFA index is associated with each component present in the repository. The DFA index is calculated based on the ranking system developed by Rampersad (1994). The quantification is also partly based on the classification system of Boothroyd and Dewhurst (1989). In order to calculate the DFA index, values are obtained for 13 different criteria (Rampersad, 1994; Hsu et al., 1998). These criteria include: 1) weight, 2) number of unique components, 3) stiffness, 4) length, 5) presence of base component, 6) vulnerability hardness, 7) shape, 8) size, 9) composing movement, 10) composing direction, 11) symmetry, 12) alignment and 13) joining method. These criteria are discussed in detail in subsequent paragraphs. The formula for calculating the DFA index is as follows (Hsu et al., 1998):

$$\text{DFA index} = 10\,(\sum P_i - \sum V_{min,i}) / (\sum V_{max,i} - \sum V_{min,i})$$

Here,

Pi : point value for each criterion, i = 1…..13

$V_{min,i}$: minimum value for each criterion

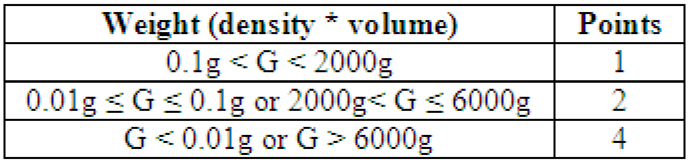
$V_{max,i}$: maximum value for each criterion

The calculated DFA index can have a value from 0 to 10, with zero being the most favorable value for ease of assembly and 10 being the worst value for the ease of assembly.

a) Effect of Weight (W)

The weight of the component determines handling, and accuracy of its positioning by the material handler. It determines the grasping force during handling, carrying capacity and also the dynamic properties of the robot. Table 3-1 shows the three weight categories and the points awarded to each category. The weight is found out by multiplying the density of the material(s) used with the total volume of the assembly. If the assembly consists of several different materials, the total density is taken as the average of the individual densities (Rampersad, 1994).

Table 3-1: Effect of weight on DFA analysis

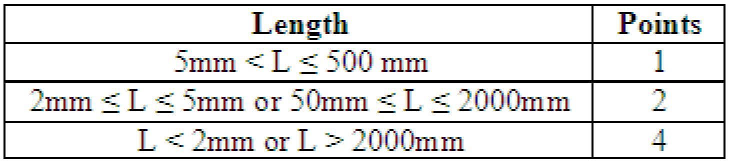


Weight (density * volume)	Points
0.1g < G < 2000g	1
0.01g ≤ G ≤ 0.1g or 2000g< G ≤ 6000g	2
G < 0.01g or G > 6000g	4

b) Effect of Length (L)

Length is the size of the longest side of the encompassed shape of the component. It is useful for describing the state of the component. For example, if the length of the component in the direction of movement is greater than one-eighth of the diameter of vibratory drum, then the component may be too large for a vibratory feeder (Boothroyd and Dewhurst, 1989). Table 3-2 highlights the three categories of length and their effect on automated assembly.

Table 3-2: Effect of length on DFA analysis

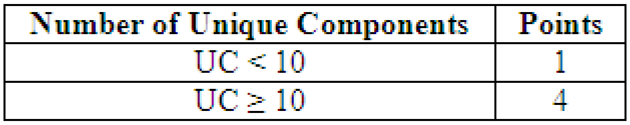


Length	Points
5mm < L ≤ 500 mm	1
2mm ≤ L ≤ 5mm or 50mm ≤ L ≤ 2000mm	2
L < 2mm or L > 2000mm	4

c) Effect of the number of Unique Components (UC)

According to Rampersad (1994), the limit of the total number of components per assembly is established at 20. Furthermore, the limit to the number of unique or different components per assembly is established to be 10. This is due to the fact that more the number of components are, the higher would be the assembly time and assembly costs. Table 3-3 shows the two categories for the number of unique components.

Table 3-3: Effect of unique components on DFA analysis

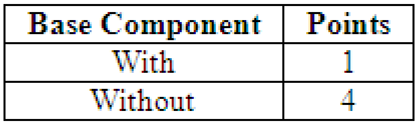


Number of Unique Components	Points
UC < 10	1
UC ≥ 10	4

d) Effect of Base Component (BC)

According to Rampersad (1994), a good base part is necessary for a component, which functions as a product barrier both during composition and transport between various assembly cells. Table 3-4 shows the classification with regards to the presence of a base component.

Table 3-4: Effect of a base component on DFA analysis

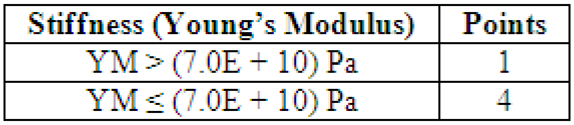


Base Component	Points
With	1
Without	4

e) Effect of stiffness – Young's Modulus (YM)

A material with high elasticity modulus (Young's Modulus) is described as stiff and a material with low modulus is described as flexible (Rampersad, 1994). Consideration for stiffness is important in the design for assembly, since it influences the feeding, handling and composing. Table 3-5 shows the classification with regards to stiffness of the component.

Table 3-5: Effect of stiffness on DFA analysis

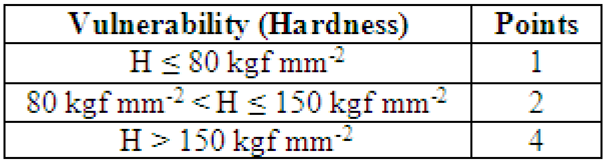


Stiffness (Young's Modulus)	Points
YM > (7.0E + 10) Pa	1
YM ≤ (7.0E + 10) Pa	4

f) Effect of Vulnerability - Hardness (VH)

Vulnerability entails the damage or wear that can be caused within a component by dynamic loading such as dropping, vibration and bumping (Rampersad, 1994). Vulnerability of the component is dependent on its elongation and is important in feeding of the components. According to Longmoen (1985), vulnerability is related to the drop. Table 3-6 shows the categories related to vulnerability and their associated points. Components having hardness between 80 and 150 kgf/mm^2 are vulnerable at a fall of 50 mm, while those having hardness value greater than 150 kgf/mm^2 get deformed after a fall of 50mm (Rampersad, 1994).

Table 3-6: Effect of vulnerability on DFA analysis

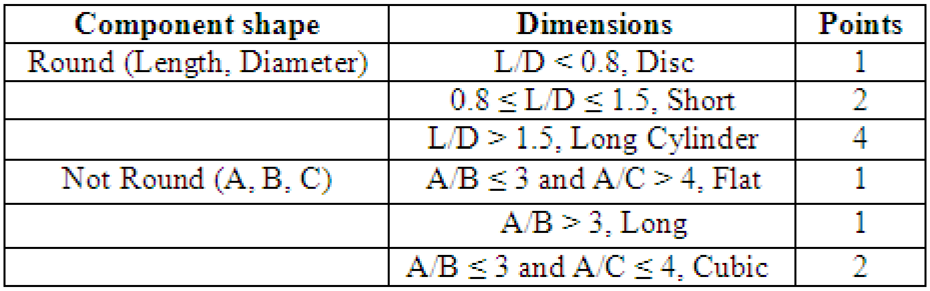


Vulnerability (Hardness)	Points
$H \leq 80$ kgf mm^{-2}	1
80 kgf mm^{-2} $< H \leq 150$ kgf mm^{-2}	2
$H > 150$ kgf mm^{-2}	4

g) Effect of Shape

Shape of the component particularly influences feeding and composing. Shape of the component is generally classified either as round or non-round (Rampersad, 1994). In the case of round components, they are classified as disc, short cylinder and long cylinder. Long cylinders and discs are found to have better orientation properties than short cylinders. Non-round components are classified as flat, long or cubic. Typically, flat and cubically shaped components have better orientation properties than long components. Table 3-7 highlights the shape classification where Length (L) and Diameter (D) represent round components while A, B and C represent the three orthogonal sides of a flat, long or cubically shaped component.

Table 3-7: Effect of shape on DFA analysis

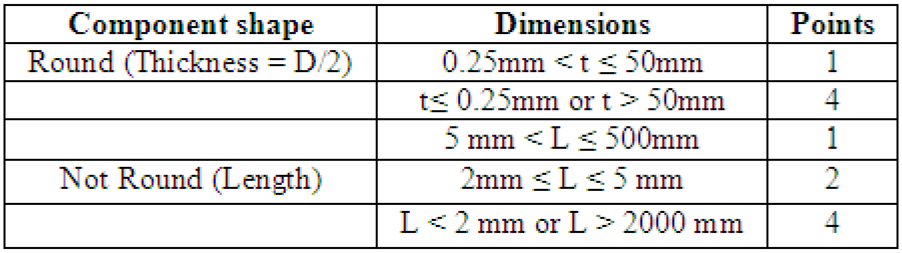


Component shape	Dimensions	Points
Round (Length, Diameter)	$L/D < 0.8$, Disc	1
	$0.8 \leq L/D \leq 1.5$, Short	2
	$L/D > 1.5$, Long Cylinder	4
Not Round (A, B, C)	$A/B \leq 3$ and $A/C > 4$, Flat	1
	$A/B > 3$, Long	1
	$A/B \leq 3$ and $A/C \leq 4$, Cubic	2

h) Effect of Size

Size of the components is important as it influences the feeding. The size is generally divided into length and thickness. Thickness is the size of the shortest side, which in case of round components equals half the diameter (Rampersad, 1994). Table 3-8 shows the size classification obtained by considering the Length (L) and thickness (t) of the components.

Table 3-8: Effect of size on DFA analysis

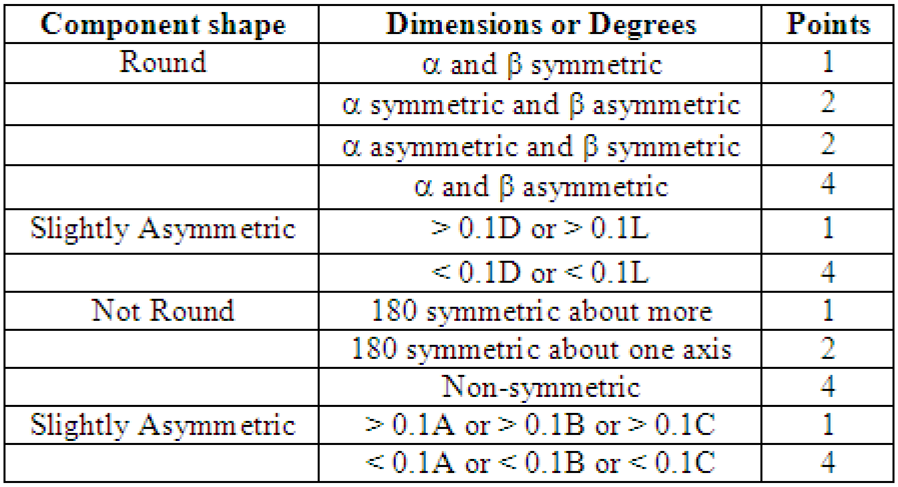


Component shape	Dimensions	Points
Round (Thickness = D/2)	$0.25mm < t \le 50mm$	1
	$t \le 0.25mm$ or $t > 50mm$	4
	$5 mm < L \le 500mm$	1
Not Round (Length)	$2mm \le L \le 5 mm$	2
	$L < 2 mm$ or $L > 2000 mm$	4

i) Effect of Symmetry

Symmetry of the component also influences the feeding, and has been classified into two types: symmetric and asymmetric (Boothroyd and Dewhurst, 1989). Symmetric components have simple orientation characteristics, which have advantages for feeding and composing (Rampersad, 1994). Each component has two orientation directions: around its longitudinal axis and transverse axis, resulting in alpha symmetry and beta symmetry. Alpha (α) symmetry is the rotation symmetry of a component around an axis perpendicular to the assembly direction, while Beta (β) symmetry is the rotation symmetry around the axis in the assembly direction. Table 3-9 shows the classification where Length (L) and Diameter (D) represent round components while A, B and C represent the three orthogonal sides of flat, long or cubically shaped components.

Table 3-9: Effect of symmetry on DFA analysis.

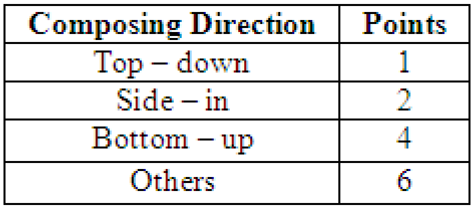


Component shape	Dimensions or Degrees	Points
Round	α and β symmetric	1
	α symmetric and β asymmetric	2
	α asymmetric and β symmetric	2
	α and β asymmetric	4
Slightly Asymmetric	> 0.1D or > 0.1L	1
	< 0.1D or < 0.1L	4
Not Round	180 symmetric about more	1
	180 symmetric about one axis	2
	Non-symmetric	4
Slightly Asymmetric	> 0.1A or > 0.1B or > 0.1C	1
	< 0.1A or < 0.1B or < 0.1C	4

j) Effect of Composing Direction (CD)

Composing direction is the direction in which the component is moved during assembly. According to Rampersad (1994), vertical composing movement from above is more favorable than either from the sides or from below. Table 3-10 shows the classification with regards to the different composing directions.

Table 3-10: Effect of composing direction on DFA analysis

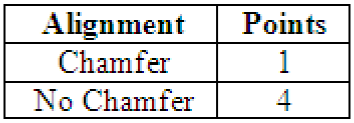


Composing Direction	Points
Top – down	1
Side – in	2
Bottom – up	4
Others	6

k) Effect of Alignment

Consideration for alignment is necessary for assemblies in order to avoid stagnation during composing (Rampersad, 1994). As a result, in the case of a peg-hole assembly, a chamfer needs to be provided on the peg and/or hole for proper alignment. Table 3-11 highlights the points awarded for each of the two criterion.

Table 3-11: Effect of alignment on DFA analysis

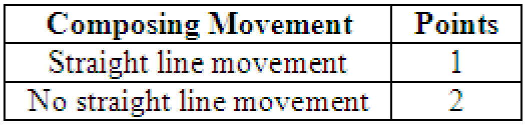


Alignment	Points
Chamfer	1
No Chamfer	4

l) Effect of Composing Movement (CM)

The trajectory followed by a robotic manipulator during composing is important as problems may occur if the trajectory is complex (Rampersad (1994). Hence, a straight line trajectory is preferred. Table 3-12 shows the points awarded for the two criteria.

Table 3-12: Effect of composing movement on DFA analysis

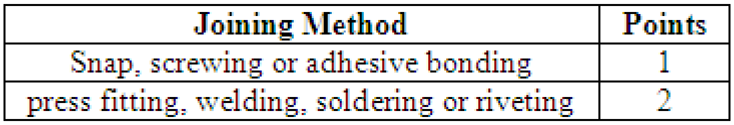


Composing Movement	Points
Straight line movement	1
No straight line movement	2

m) Effect of Joining Method (JM)

The joining method may either be a permanent method such as welding, or could be a temporary method like screwing. The joining method is important in the DFA consideration, as it has an effect on the total assembly time. Table 3-13 shows the different joining methods taken into consideration for DFA, and the associated points.

Table 3-13: Effect of joining method on DFA analysis



Joining Method	Points
Snap, screwing or adhesive bonding	1
press fitting, welding, soldering or riveting	2

Once the overall component combinations are generated, the total DFA index is calculated for each design by summing the DFA indexes of each component present in that design. The design offering the least total DFA index is ranked one, and similarly the other combinations are ranked in ascending order based on their DFA indexes. Section 3.3 illustrates the rule-based searching method utilized in the proposed research framework.

3.3 Rule-Based Searching Technique

The proposed research framework utilizes the rule-based search technique. Each rule comprises of up to three nodes of input/output flows and sub-functions. A general graph grammar rule may be represented as follows:

R1: IF ((***input1*** = A1 & ***sub-function1*** = B1 & ***output1*** = C1) & (***input2*** = A2 & ***sub-function2*** = B2 & ***output2*** = C2) & (***input3*** = A3 & ***sub-function3*** = B3 & ***output3*** = C3)) THEN RETRIEVE ALL COMPONENTS FROM TABLE DFA WITH ***baseid*** = BID;

Here, a number is associated with each rule (e.g., R1). A1, A2, A3 are the three input flows, while C1, C2, C3 are the three output flows and B1-B3 are the three sub-functions of the three nodes. The rules are stored in a catalogued order and are compared with each set of nodes in the EMS functional model input GUI (Figure 3-3). Whenever a complete match is found between the antecedent of the graph grammar rule and nodes of the GUI, the appropriate rule gets triggered. The firing of the rule results in retrieving all the possible component combinations from the design repository that has the base ID equal to the base ID associated with the fired rule.

An exhaustive search is conducted, which seeks to combine every component alternative retrieved from a single fired rule to each of the component alternatives retrieved from all the other fired rules, resulting in the full set of possible combinations. For example, let us assume that a query results in the overall firing of three rules: R1, R2 and R3, where R1 is associated with M1 components from the repository, R2 retrieves M2 components, and R3 retrieves M3 components. The total number of combinations generated would be M1*M2*M3. In the proposed framework, all these conceptual design combinations are stored in a temporary mySQL table 'final_info', which are ranked in ascending order of their total DFA index value. The proposed framework allows the designer to visualize all the generated conceptual design alternatives by viewing their images. The designer then selects two conceptual designs for modularization. In the proposed software framework, it is decided to have two concept alternatives for the sake of simplicity; however, by making slight modifications in the software code, we may easily allow the designer to select more than two designs for modularization. Section 3.4 discusses the modularization approach utilized for the proposed framework.

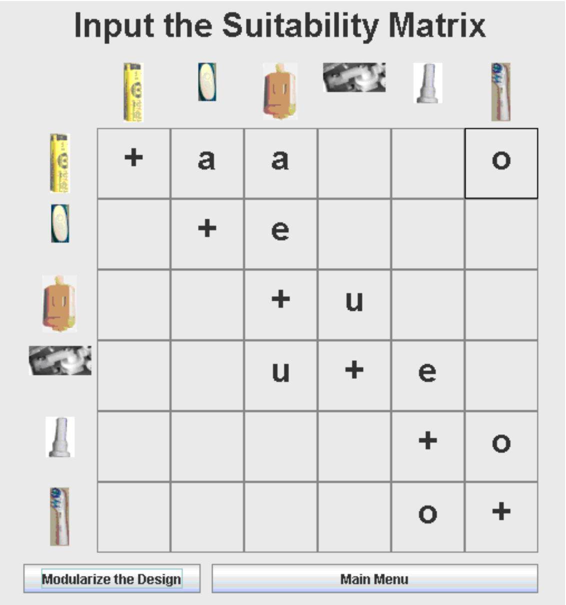
3.4 Modularization Approach

In the proposed research framework, Decomposition Approach (DA) is choosen for modularization, since DA is found out to be the most suitable modularization technique under the objective metric of "Design for Assembly" and "Design for Variety" (Gupta and Okudan, 2008a). DA is a matrix based modularization approach and the two input matrices are interaction and suitability matrices. Suitability matrix represents the suitability for components for inclusion in a module, while the interaction matrix represents the interaction between the components. The interaction matrix is generated for the user selected design, based on the 'interaction' table present in the repository. The suitability matrix is generated at runtime since it may be possible that two components interact with each other, but they may not be suitable for inclusion within a module. The suitability matrix is also dependent on the customer needs to a great extent.

Suitability matrix is generated by asking the designer whether the components within each of the two selected designs are suitable to be included within the same module. Figure 3-7 (a) shows the different levels of input for the suitability matrix (Huang and Kusiak, 1998).

(a)

Input Letter	Corresponding Meaning
a	strongly desired
e	desired
o	strongly undesired
u	undesired



Input the Suitability Matrix

+	a	a			o
	+	e			
		+	u		
		u	+	e	
				+	o
				o	+

(b)

Figure 3-7: (a) Levels of input for suitability matrix (b) GUI for suitability matrix

Unlike the interaction matrix, direction is not important for the suitability database. As shown in Figure 3-7(b), the suitability level for a component that is to be included along with another component is indicated in the appropriate cell corresponding to both the components. The suitability may be also left blank, if it does not make any difference if the two components are included within the same module.

Once the suitability and interaction matrices are defined, DA (Huang and Kuisak, 1998) is applied in order to transform the interaction and suitability matrices, allowing one to explore the potential modules amongst components. The interaction and suitability matrices are generated for each conceptual design variant, which are defined as follows:

- The interaction matrix A is defined as: $[a_{ij}]_{m \times m}$ where a_{ij} is the interaction between component i and component j.
- The suitability matrix B is defined as: $[b_{ij}]_{m \times m}$ where b_{ij} represents the suitability for components i and j for inclusion in a module.

The triangularization approach presented in Kusiak et al. (1994) is applied in order to transform the interaction matrix A to its triangularized equivalent (i.e., A'). The suitability matrix is accordingly arranged from B to B' so that the sequence of rows and columns in the matrix B' are same as those in A'. The next step involves the combination of both the matrices: A' and B' to form the modularity matrix, which is represented as [A'|B']. Components are then removed from the module, if they satisfy all the following conditions:

- The component (i.e., k) and any other component (i.e., l) in the same module are undesired to for inclusion in the module.
- Component k interacts with the remaining components in the module to a degree, which is less than component l.
- None of the resultant module is empty.

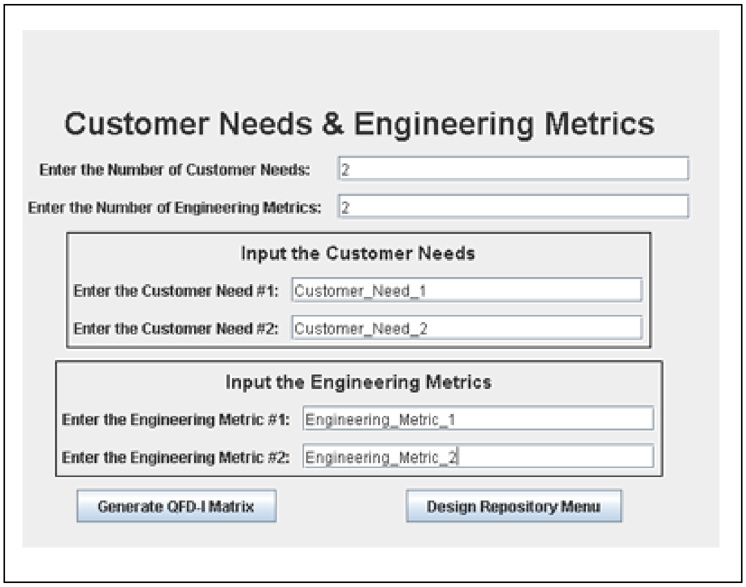
These components removed are placed at the end of the modularity matrix and this process is repeated until no more components can be deleted. On the other hand, those components that are strongly desired for simultaneous inclusion in two modules are duplicated and this process continues until no more components can be duplicated.

The two user selected modularized conceptual variants are then analyzed according to variety considerations and the secondary filtering is done to select the best modularized conceptual design. Section 3.5 discusses the DFV index calculation.

3.5 DFV Index Calculation

This filtering criterion is known as DFV index, or the Generational Variety Index (GVI) adopted from Martin and Ishii (2000). It is basically an indicator of the amount of redesign required for each component within a product in order to meet the future market requirements. DFV index is basically an estimate of the required changes in a component due to external or non-controllable factors (Martin and Ishii, 2000). In the proposed framework, it is desired that the modules should be formed in a way such that all the customer needs likely to change in the future, are clustered into a single module. This would enable the manufacturing firm to just replace the particular module in order to satisfy the future needs. Hence, the two design alternatives are evaluated and the design which generates the least DFV index is selected as the best design. In order to quantify these customer needs, required for generating the DFV index, a two phase QFD technique is adopted from the research by Martin and Ishii (2000).

In the first QFD phase, a relationship is developed between the customer needs and the Engineering Metrics (EM). The engineering metrics are measurable items, which are translations of subjective customer needs into quantifiable engineering specifications. Figure 3-8 shows the GUI for inputting the customer needs as well as engineering metrics, while Figure 3-9 shows a general QFD-I matrix.



Customer Needs & Engineering Metrics

Enter the Number of Customer Needs: 2

Enter the Number of Engineering Metrics: 2

Input the Customer Needs

Enter the Customer Need #1: Customer_Need_1

Enter the Customer Need #2: Customer_Need_2

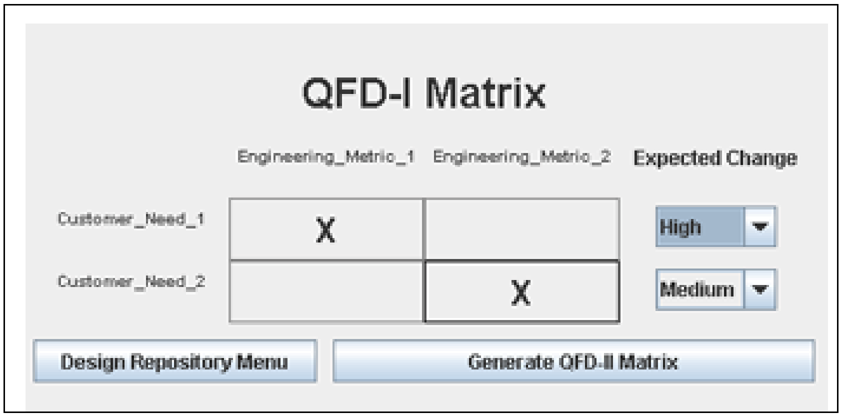
Input the Engineering Metrics

Enter the Engineering Metric #1: Engineering_Metric_1

Enter the Engineering Metric #2: Engineering_Metric_2

Figure 3-8: GUI for inputting the customer needs and engineering metrics

The next step involves qualitatively estimating (e.g., High/Medium/Low) the range of change of customer needs (Martin and Ishii, 2000). This step enables the design team to begin thinking about how the customer needs change with time.



QFD-I Matrix

	Engineering_Metric_1	Engineering_Metric_2	Expected Change
Customer_Need_1	X		High
Customer_Need_2		X	Medium

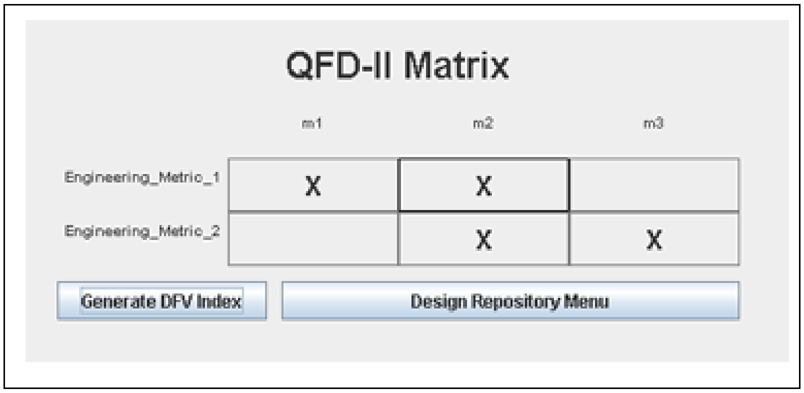
Figure 3-9: QFD phase I matrix

It is desired that the customer needs which are forecasted to change significantly with time, should be restricted to only one module and not many modules. Otherwise, all modules need to be re-designed in order to accommodate future customer needs. If the range of change of customer needs is high, then it is denoted by 3. If the range of change is medium, then it is denoted as 2, while low range of change of customer needs is denoted 1. Table 3-14 summarizes the rating system proposed for the DFV index calculation.

Table 3-14: The rating system for DFV index calculation

Range of change of customer needs	**Rank associated**
High	3
Medium	2
Low	1

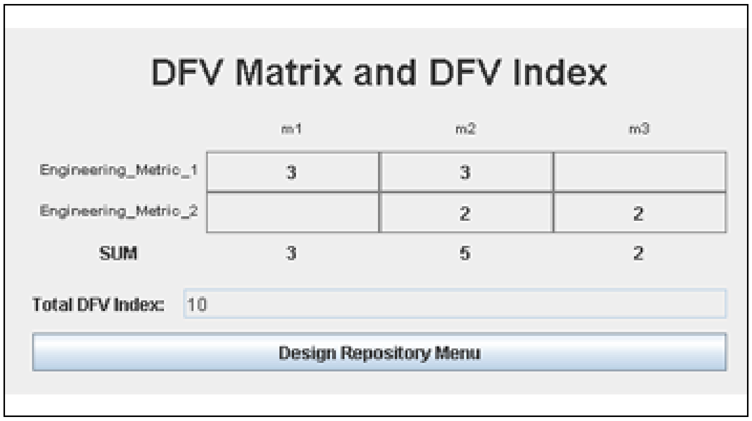
The next step is to generate the QFD Phase II matrix, which maps the engineering metrics from phase I to the modules used in the design. A general QFD Phase II matrix is shown in Figure 3-10.



	m1	m2	m3
Engineering_Metric_1	X	X	
Engineering_Metric_2		X	X

Figure 3-10: A general QFD phase II matrix

The final step involves the development of the DFV matrix and calculation of the DFV index**.** The total GVI value is obtained by summing the last row of the DFV matrix for each concept variant. The conceptual design having lower DFV index value is best overall concept, since lesser redesign effort would be required within its architecture in order to meet the future customer requirements. Figure 3-11 shows a general DFV matrix with the index value associated.



DFV Matrix and DFV Index

	m1	m2	m3
Engineering_Metric_1	3	3	
Engineering_Metric_2		2	2
SUM	3	5	2

Total DFV Index: 10

Figure 3-11: A general DFV matrix

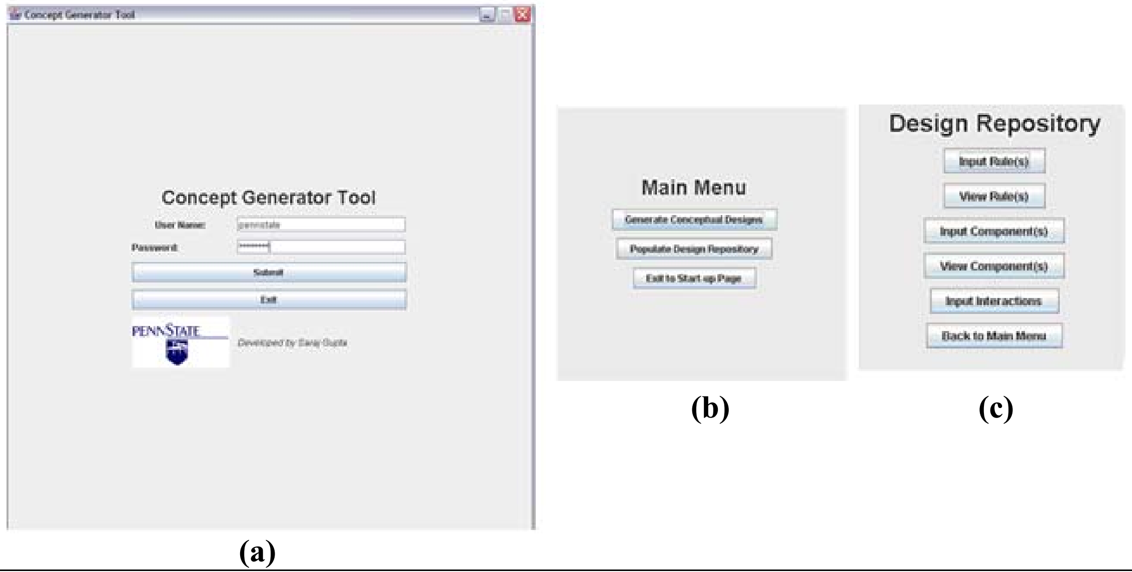
Chapter 4 demonstrates the working of the proposed software framework with the help of two electro-mechanical product design examples. Firstly, the working of the framework is demonstrated on an electronic toothbrush design example. Secondly, its working is illustrated with the design of a mounting system for a Variable Message Sign (VMS).

Chapter 4

Demonstration of the Proposed Framework through Design Examples

In this chapter, working of the proposed framework is demonstrated with the help of two design examples. The first example is an electronic toothbrush design, while the second example involves the design of a mount system for a Variable Message Sign (VMS).

The software comprises of an authentication screen (Figure 4-1a), which grants the user, access to the software only if the correct username password combination is entered. Once the right username / password combination is entered, the Main Menu screen is displayed (Figure 4-1b). It is assumed that the design repository is initially empty, and therefore the preliminary step is to populate the design repository. When the user selects the "Populate Design Repository" button in the Main Menu, he/she is guided to the "Design Repository" menu (Figure 4-1c).



User Name: pennstate

Password: ********

Figure 4-1: GUI's of (a) Authentication screen (b) Main menu (c) Design repository

In the Design Repository menu, the user can input components, graph grammar rules and interaction between the rules. The designer can also view the existing rules and components. Section 4.1 discusses the implementation of the proposed framework on an electronic toothbrush design.

4.1 Electronic Toothbrush Design Example

This section illustrates the working of the proposed research framework as implemented on an electronic toothbrush design example (Gupta and Okudan, 2008b; Gupta and Okudan, 2008c). The foremost step is the population of 'DFA' and 'rules' tables for the conceptual design of the electronic toothbrush. Prior to populating the design repository, dissection of the two different electronic toothbrush models was conducted in order to examine the components that make up the toothbrush. The two models are: 1) Oral-B™ Vitality Series® Dual Clean and 2) Crest™ Spin Brush Pro. Components are determined for the two electric toothbrushes as follows:

(a) Crest^{TM} - Spin Brush Pro:

This electronic toothbrush is produced by Crest^{TM} (http://www.crest.com; viewed on: 8th November, 2008), and comprises of six major components obtained by dissection. These are highlighted in Figures 4-2 and 4-3.

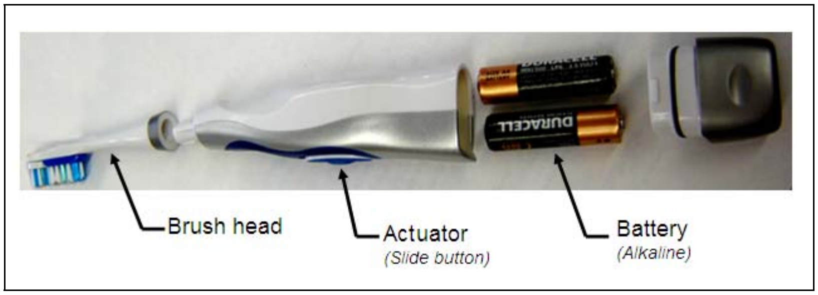


Figure 4-2: Crest Spin Pro toothbrush after preliminary dissection

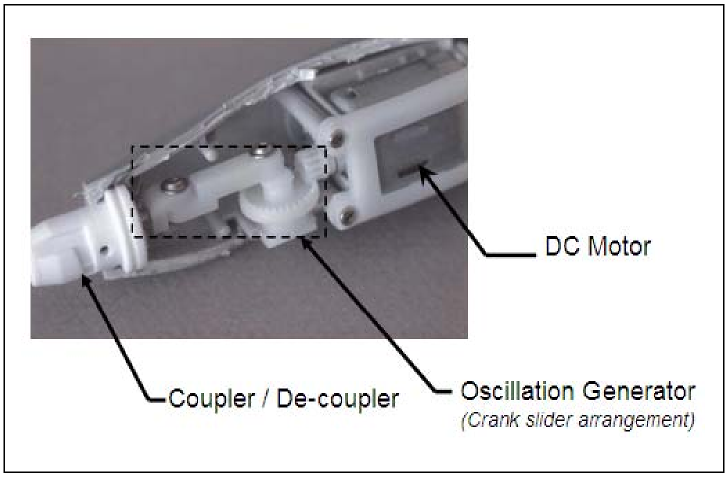


Figure 4-3: Crest Spin Pro toothbrush after secondary dissection

(b) Oral-B^{TM} Vitality - Dual Clean:

This electronic toothbrush model is produced by Oral-B^{TM} (http://www.oral-b.com; viewed on: 8th November, 2008), and comprises of six major components obtained by dissection. These components are highlighted in Figures 4-4a and 4-4b.

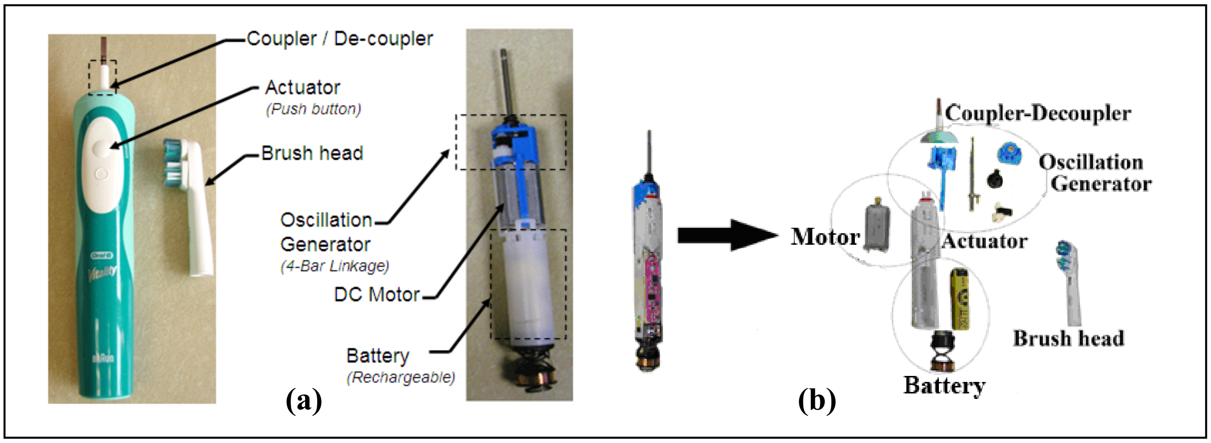
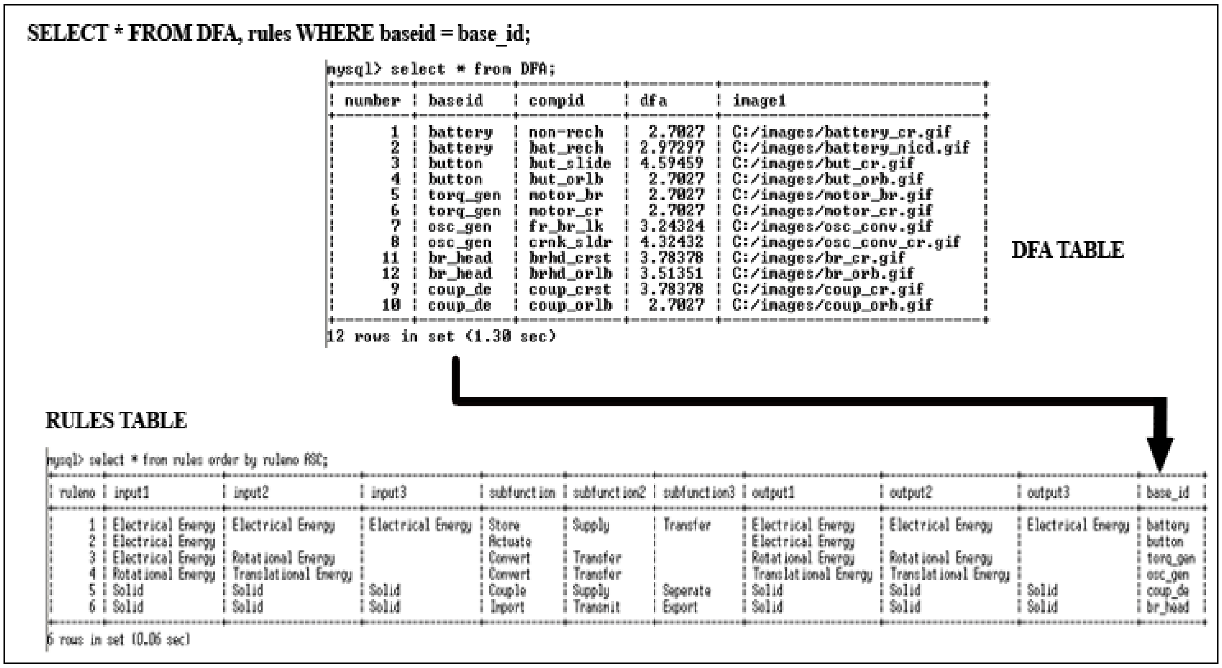


Figure 4-4: (a) Oral-B toothbrush major components (b) secondary dissection

There are six major components in each of the above two toothbrush models, which correspond to six different graph grammar rules, with two component alternative associated with every rule. The graph grammar rules have to be input into the repository before inserting the components as each component needs to be associated with a rule. The rules are entered into the ‘rules’ table using the input rule(s) GUI (Figure 3-4). Although each component alternative is associated with the same overall rule, it would have a different component ID, DFA index value, as well as the component image. These parameters corresponding to each component are entered into the ‘DFA’ using the component(s) inserting GUI (Figure 3-5). Figure 4-5 shows the rows associated with ‘DFA’ table and ‘rules’ table, along with the relationship between the two tables.



SELECT * FROM DFA, rules WHERE baseid = base_id;

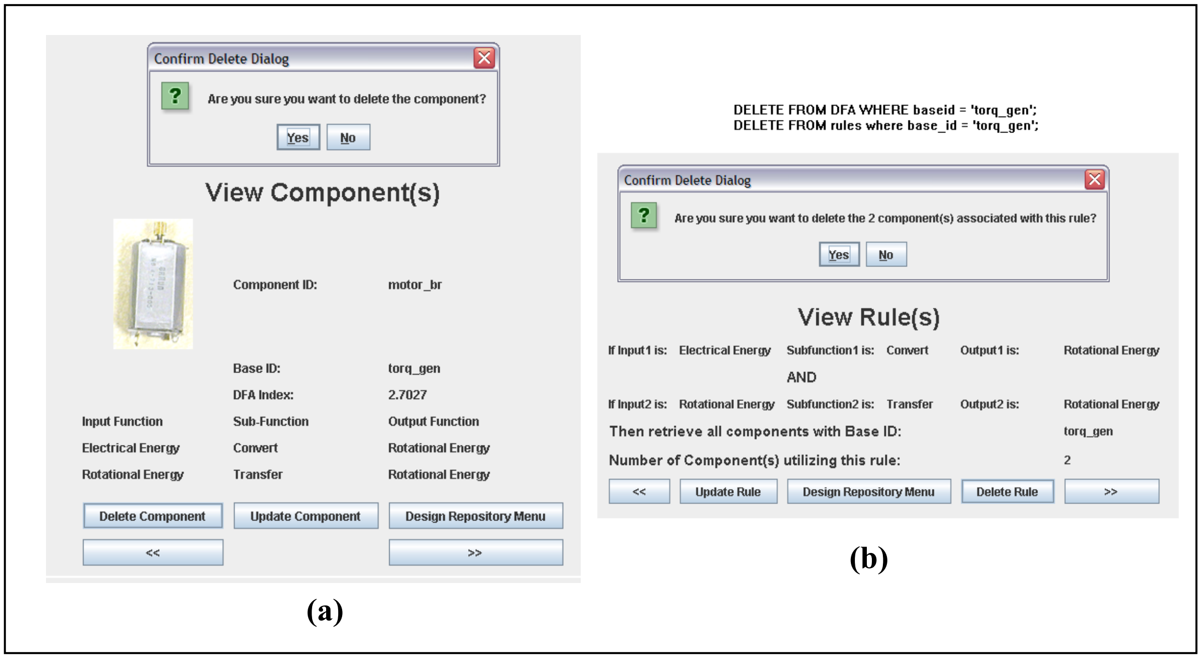
```
mysql> select * from DFA;
+--------+----------+------------+---------+-----------------------------+
| number | baseid   | compid     | dfa     | image1                      |
+--------+----------+------------+---------+-----------------------------+
|      1 | battery  | non-rech   |  2.7027 | C:/images/battery_cr.gif    |
|      2 | battery  | bat_rech   | 2.97297 | C:/images/battery_nicd.gif  |
|      3 | button   | but_slide  | 4.59459 | C:/images/but_cr.gif        |
|      4 | button   | but_orlb   |  2.7027 | C:/images/but_orb.gif       |
|      5 | torq_gen | motor_br   |  2.7027 | C:/images/motor_br.gif      |
|      6 | torq_gen | motor_cr   |  2.7027 | C:/images/motor_cr.gif      |
|      7 | osc_gen  | fr_br_lk   | 3.24324 | C:/images/osc_conv.gif      |
|      8 | osc_gen  | crnk_sldr  | 4.32432 | C:/images/osc_conv_cr.gif   |
|     11 | br_head  | brhd_crst  | 3.78378 | C:/images/br_cr.gif         |
|     12 | br_head  | brhd_orlb  | 3.51351 | C:/images/br_orb.gif        |
|      9 | coup_de  | coup_crst  | 3.78378 | C:/images/coup_cr.gif       |
|     10 | coup_de  | coup_orlb  |  2.7027 | C:/images/coup_orb.gif      |
+--------+----------+------------+---------+-----------------------------+
12 rows in set (1.30 sec)
```

```
mysql> select * from rules order by ruleno ASC;
+--------+-------------------+---------------------+-------------------+-------------+--------------+--------------+---------------------+---------------------+-------------------+----------+
| ruleno | input1            | input2              | input3            | subfunction | subfunction2 | subfunction3 | output1             | output2             | output3           | base_id  |
+--------+-------------------+---------------------+-------------------+-------------+--------------+--------------+---------------------+---------------------+-------------------+----------+
|      1 | Electrical Energy | Electrical Energy   | Electrical Energy | Store       | Supply       | Transfer     | Electrical Energy   | Electrical Energy   | Electrical Energy | battery  |
|      2 | Electrical Energy |                     |                   | Actuate     |              |              | Electrical Energy   |                     |                   | button   |
|      3 | Electrical Energy | Rotational Energy   |                   | Convert     | Transfer     |              | Rotational Energy   | Rotational Energy   |                   | torq_gen |
|      4 | Rotational Energy | Translational Energy |                  | Convert     | Transfer     |              | Translational Energy | Translational Energy |                  | osc_gen  |
|      5 | Solid             | Solid               | Solid             | Couple      | Supply       | Seperate     | Solid               | Solid               | Solid             | coup_de  |
|      6 | Solid             | Solid               | Solid             | Import      | Transmit     | Export       | Solid               | Solid               | Solid             | br_head  |
+--------+-------------------+---------------------+-------------------+-------------+--------------+--------------+---------------------+---------------------+-------------------+----------+
6 rows in set (0.06 sec)
```

Figure 4-5: Relationship between ‘DFA’ table and ‘rules’ table in MySQL.

Once the user has entered all the rules and components into the two tables within the repository, he/she can view these components and rules through the ‘Design Repository’ menu. The user can also update or delete an existing component and/or rule through the ‘Design

Repository' menu. As shown in Figures 4-6a and 4-6b, when the user selects a component to be deleted, a confirmation dialog window is displayed to confirm the deletion of the component.



Confirm Delete Dialog

Are you sure you want to delete the component?

View Component(s)

Component ID: motor_br

Base ID: torq_gen

DFA Index: 2.7027

Input Function	Sub-Function	Output Function
Electrical Energy	Convert	Rotational Energy
Rotational Energy	Transfer	Rotational Energy

```
DELETE FROM DFA WHERE baseid = 'torq_gen';
DELETE FROM rules where base_id = 'torq_gen';
```

Confirm Delete Dialog

Are you sure you want to delete the 2 component(s) associated with this rule?

View Rule(s)

If Input1 is: Electrical Energy Subfunction1 is: Convert Output1 is: Rotational Energy

AND

If Input2 is: Rotational Energy Subfunction2 is: Transfer Output2 is: Rotational Energy

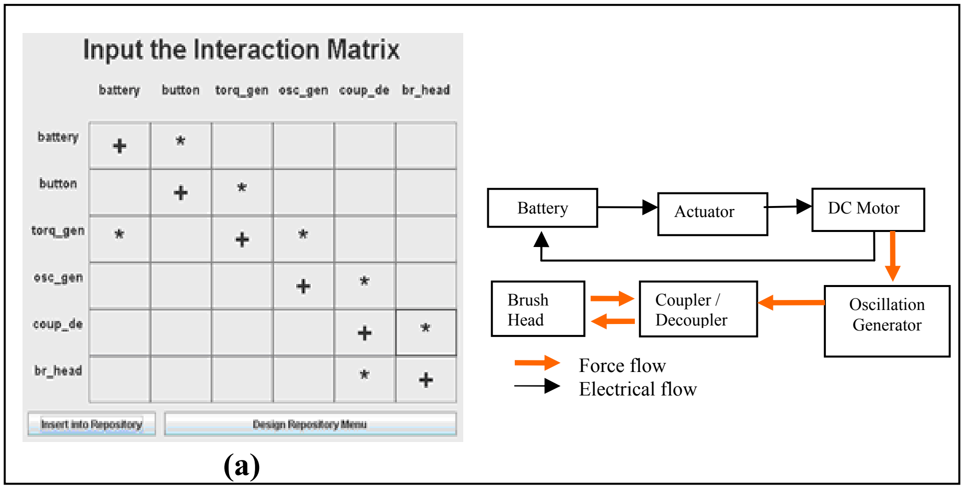
Then retrieve all components with Base ID: torq_gen

Number of Component(s) utilizing this rule: 2

Figure 4-6: Deleting (a) motor component, and (b) its corresponding rule

In the case of deleting a rule, all the components associated with that rule also need to be deleted; therefore, the user has to confirm the deletion of the rule along with all the components associated with the rule. The component viewing GUI (Figure 4-6a) also allows the user to update a component, where the user can change the DFA index, component ID or the rule associated with the component. The rules viewing GUI also allows the user to modify an existing rule.

The third table within the design repository which needs to be populated is the interaction table. This highlights the interactions between the various rules stored in the design repository. Figure 4-7 shows the interaction GUI for the electronic toothbrush example. The user can start generating designs once the design repository is populated to his/her satisfaction. In order to generate the conceptual design combinations, the user goes one step back to the 'Design Repository Menu' (Figure 4-1c) and then back to the 'Main Menu' (Figure 4-1b). The user can then select the 'Generate Conceptual Designs' button in the 'Main Menu'. Prior to generating the conceptual designs, the designer needs to prepare a black-box model and EMS model to clearly identify the major functions and flows that occur within the product. The user also needs to input the interactions amongst the rules before generating the conceptual design. Section 4.1.1 discusses the generation of the conceptual designs by querying the EMS diagram.



Input the Interaction Matrix

	battery	button	torq_gen	osc_gen	coup_de	br_head
battery	+	*				
button		+	*			
torq_gen	*		+	*		
osc_gen				+	*	
coup_de					+	*
br_head					*	+

Figure 4-7: (a) GUI for inputting the interaction matrix (b) interactions between rules

4.1.1. Generation of Conceptual Designs

The designer needs to be well aware of the customer needs and target population before designing a new product or even re-designing an existing product. Therefore, the first step involves the customer needs assessment for the design of an electronic toothbrush. Table 4-1 shows the customer needs for an electronic toothbrush, which were obtained by analyzing the customer complaints associated with Oral-B™ and Crest™ toothbrushes, in super-store websites like Target, Walmart etc.

Table 4-1: Customer needs assessment for an electronic toothbrush

Non-bulky
Cheap
Portable
Less noisy
Compatible with different brush heads

Once customer needs are obtained and analyzed, the next step involves the development of the black-box model, as well as the EMS functional structure. In an electronic toothbrush, electricity and human force are provided as input energies to the toothbrush, along with a particular brush head, and as a result, the bristles execute a translatory motion. Noise can be assumed to be an outcome of the toothbrush. Figure 4-8 shows the black-box model of an electronic toothbrush, which displays the input/output flows, and the overall function performed by the toothbrush. The overall function is broken into sub-functions connected by flows of energies, materials and signals in order to generate the EMS model (Figure 4-9).

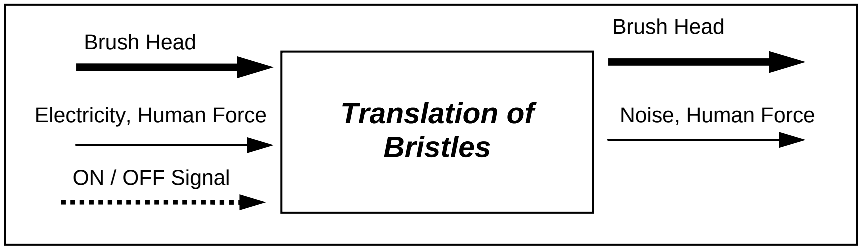


Figure 4-8: Black box model of an electronic toothbrush

The EMS diagram generated in Figure 4-9 is then entered into the software framework, using the ‘Input EMS Diagram’ GUI. Each row in the GUI screen comprises of an input flow, sub-function and output flow, and these correspond to each node within the EMS diagram. Figure 3-3 shows the GUI screen for inputting the nodes. In several instances, the generated EMS diagrams are quite exhaustive, and re-entering them again into the system may take a lot of time.

Therefore, the GUI also allows the user to save the overall query as text file, and load the file if another similar design needs to be generated at a later stage. Figure 4-10 shows the saved query file for the toothbrush EMS diagram.

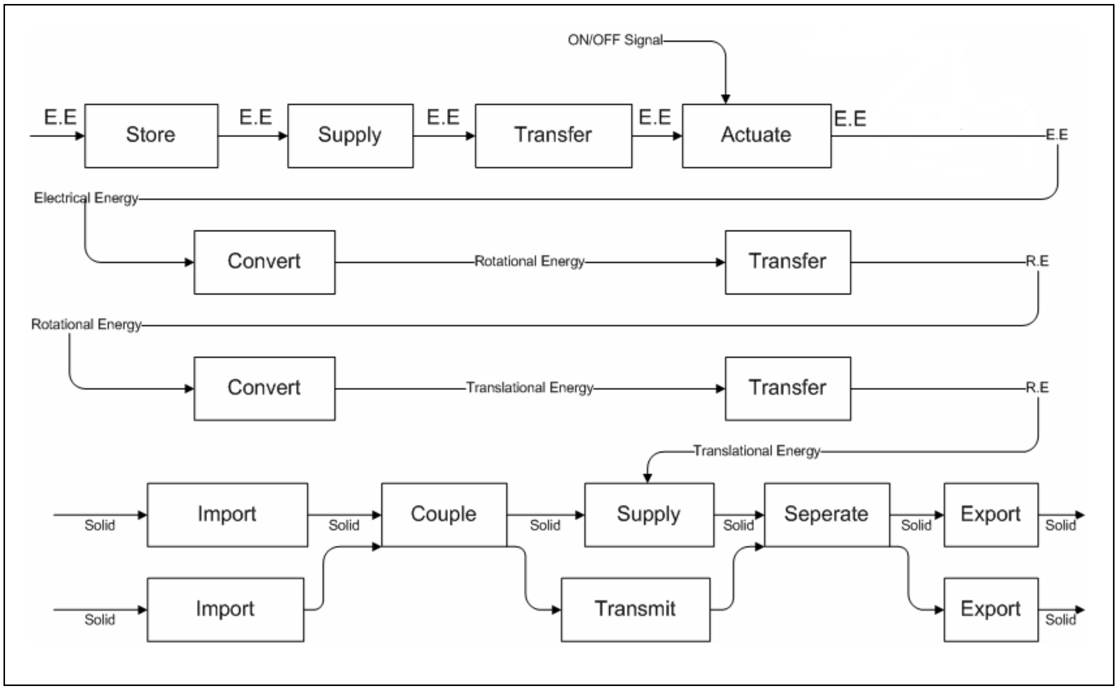
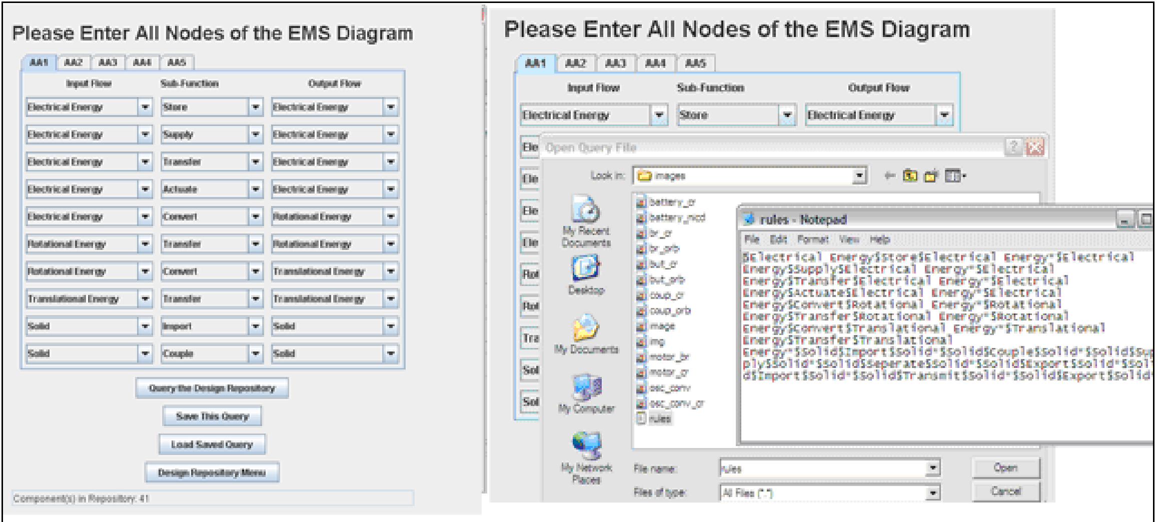


Figure 4-9: Function structure of the electronic toothbrush.

Once all the nodes have been input, the user clicks on “Query the Design Repository” button, and the rule-based searching logic is utilized to generate conceptual design alternatives. Rule-based searching is discussed in Section 3.2.3. There are 2 possible variants for each of the six major components, which correspond to the Oral-B^{TM} and Crest^{TM} models respectively. As a result, 2^6 conceptual design combinations are possible for satisfying the EMS diagram of the electronic toothbrush. The software framework lists all the 64 possible combinations, calculates the total DFA index associated each combination, and finally ranks the conceptual design in the ascending order based on aggregated DFA index values.



Please Enter All Nodes of the EMS Diagram

Input Flow	Sub-Function	Output Flow
Electrical Energy	Store	Electrical Energy
Electrical Energy	Supply	Electrical Energy
Electrical Energy	Transfer	Electrical Energy
Electrical Energy	Actuate	Electrical Energy
Electrical Energy	Convert	Rotational Energy
Rotational Energy	Transfer	Rotational Energy
Rotational Energy	Convert	Translational Energy
Translational Energy	Transfer	Translational Energy
Solid	Import	Solid
Solid	Couple	Solid

Component(s) in Repository: 41

Please Enter All Nodes of the EMS Diagram

Input Flow	Sub-Function	Output Flow
Electrical Energy	Store	Electrical Energy

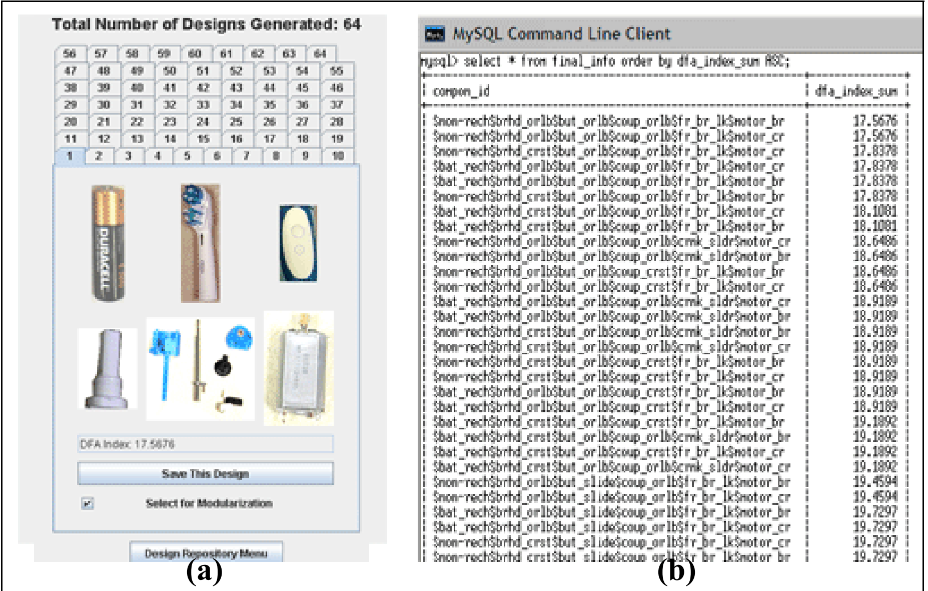
File name: rules

Files of type: All Files (*.*)

```
$Electrical Energy$Store$Electrical Energy*$Electrical
Energy$Supply$Electrical Energy*$Electrical
Energy$Transfer$Electrical Energy*$Electrical
Energy$Actuate$Electrical Energy*$Electrical
Energy$Convert$Rotational Energy*$Rotational
Energy$Transfer$Rotational Energy*$Rotational
Energy$Convert$Translational Energy*$Translational
Energy$Transfer$Translational
Energy*$Solid$Import$Solid*$Solid$Couple$Solid*$Solid$Sup
ply$Solid*$Solid$Seperate$Solid*$Solid$Export$Solid*$Sol
d$Import$Solid*$Solid$Transmit$Solid*$Solid$Export$Solid
```

Figure 4-10: (a) EMS diagram input into GUI (b) loading previously saved query

Figure 4-11a shows 64 possible combinations, while Figure 4-11b shows the temporary MySQL table containing the 64 combinations. The GUI allows the user to save individual design combinations to conduct further analysis using third-party softwares.



Total Number of Designs Generated: 64

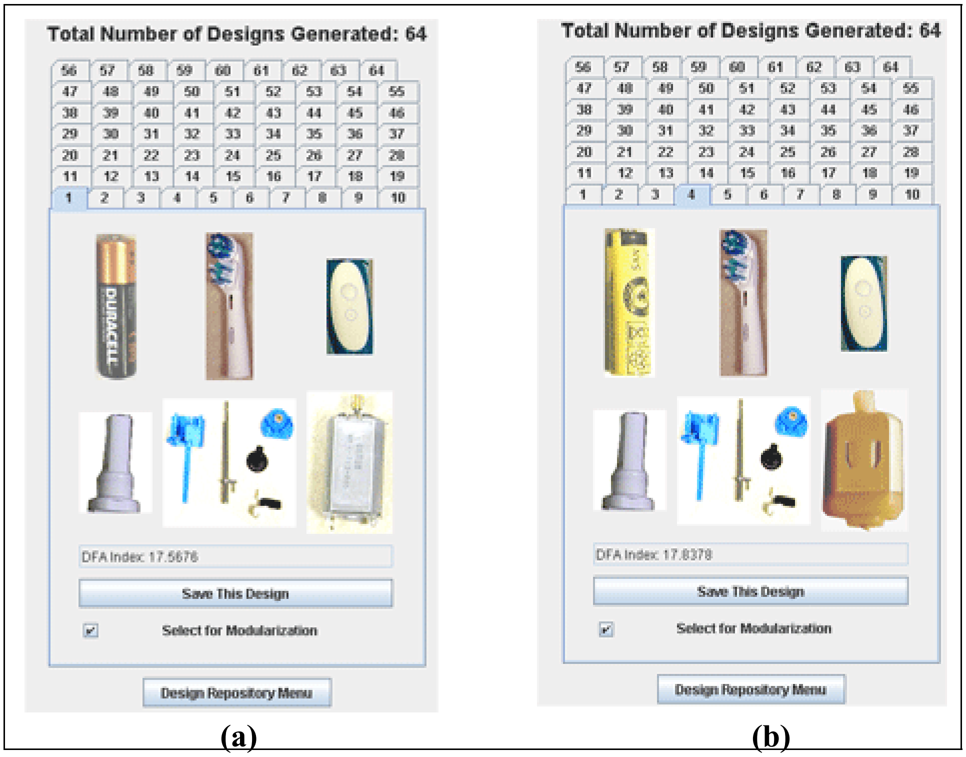
DFA Index: 17.5676

☑ Select for Modularization

```
mysql> select * from final_info order by dfa_index_sum ASC;
+--------------------------------------------------------+---------------+
| compon_id                                              | dfa_index_sum |
+--------------------------------------------------------+---------------+
| $non-rech$brhd_orlb$but_orlb$coup_orlb$fr_br_lk$motor_br  |       17.5676 |
| $non-rech$brhd_orlb$but_orlb$coup_orlb$fr_br_lk$motor_cr  |       17.5676 |
| $non-rech$brhd_crst$but_orlb$coup_orlb$fr_br_lk$motor_cr  |       17.8378 |
| $bat_rech$brhd_orlb$but_orlb$coup_orlb$fr_br_lk$motor_cr  |       17.8378 |
| $bat_rech$brhd_orlb$but_orlb$coup_orlb$fr_br_lk$motor_br  |       17.8378 |
| $non-rech$brhd_crst$but_orlb$coup_orlb$fr_br_lk$motor_br  |       17.8378 |
| $bat_rech$brhd_crst$but_orlb$coup_orlb$fr_br_lk$motor_cr  |       18.1081 |
| $bat_rech$brhd_crst$but_orlb$coup_orlb$fr_br_lk$motor_br  |       18.1081 |
| $non-rech$brhd_orlb$but_orlb$coup_orlb$cmk_sldr$motor_cr  |       18.6486 |
| $non-rech$brhd_orlb$but_orlb$coup_orlb$cmk_sldr$motor_br  |       18.6486 |
| $non-rech$brhd_orlb$but_orlb$coup_crst$fr_br_lk$motor_br  |       18.6486 |
| $non-rech$brhd_orlb$but_orlb$coup_crst$fr_br_lk$motor_cr  |       18.6486 |
| $bat_rech$brhd_orlb$but_orlb$coup_orlb$cmk_sldr$motor_cr  |       18.9189 |
| $bat_rech$brhd_orlb$but_orlb$coup_orlb$cmk_sldr$motor_br  |       18.9189 |
| $non-rech$brhd_crst$but_orlb$coup_orlb$cmk_sldr$motor_br  |       18.9189 |
| $non-rech$brhd_crst$but_orlb$coup_orlb$cmk_sldr$motor_cr  |       18.9189 |
| $non-rech$brhd_crst$but_orlb$coup_crst$fr_br_lk$motor_br  |       18.9189 |
| $non-rech$brhd_crst$but_orlb$coup_crst$fr_br_lk$motor_cr  |       18.9189 |
| $bat_rech$brhd_orlb$but_orlb$coup_crst$fr_br_lk$motor_br  |       18.9189 |
| $bat_rech$brhd_orlb$but_orlb$coup_crst$fr_br_lk$motor_cr  |       18.9189 |
| $bat_rech$brhd_crst$but_orlb$coup_crst$fr_br_lk$motor_br  |       19.1892 |
| $bat_rech$brhd_crst$but_orlb$coup_orlb$cmk_sldr$motor_br  |       19.1892 |
| $bat_rech$brhd_crst$but_orlb$coup_crst$fr_br_lk$motor_cr  |       19.1892 |
| $bat_rech$brhd_crst$but_orlb$coup_orlb$cmk_sldr$motor_cr  |       19.1892 |
| $non-rech$brhd_orlb$but_slide$coup_orlb$fr_br_lk$motor_br |       19.4594 |
| $non-rech$brhd_orlb$but_slide$coup_orlb$fr_br_lk$motor_cr |       19.4594 |
| $bat_rech$brhd_orlb$but_slide$coup_orlb$fr_br_lk$motor_br |       19.7297 |
| $bat_rech$brhd_orlb$but_slide$coup_orlb$fr_br_lk$motor_cr |       19.7297 |
| $non-rech$brhd_crst$but_slide$coup_orlb$fr_br_lk$motor_cr |       19.7297 |
| $non-rech$brhd_crst$but_slide$coup_orlb$fr_br_lk$motor_br |       19.7297 |
```

Figure 4-11: (a) The GUI showing 64 combinations (b) temporary MySQL table

It may be observed from Figure 4-12a that the design having the least assembly time (DFA Index: 17.56) comprises of five components from the Oral-B Vitality Dual-Clean toothbrush, while only the alkaline battery is from the Crest Spin Brush Pro model. Therefore, Oral-B's product architecture design strategy is found to be more efficient than its competitor. Further analysis reveals that Oral-B can minimize the assembly time of their Vitality Dual-Clean model by replacing the soldered re-chargeable battery with alkaline batteries, which would also reduce the overall cost of their toothbrush model. Figure 4-12b shows the 4th design combination which has major components as well as the soldered re-chargeable battery. It is assumed that the designer selects the first conceptual design (DFA: 17.5676) and fourth conceptual design (DFA: 17.8378) for modularization. Figure 4-12 shows the two designs selected by the designer in order to proceed to the modularization phase of the proposed software framework.



(a)

Total Number of Designs Generated: 64

DFA Index: 17.5676

☑ Select for Modularization

(b)

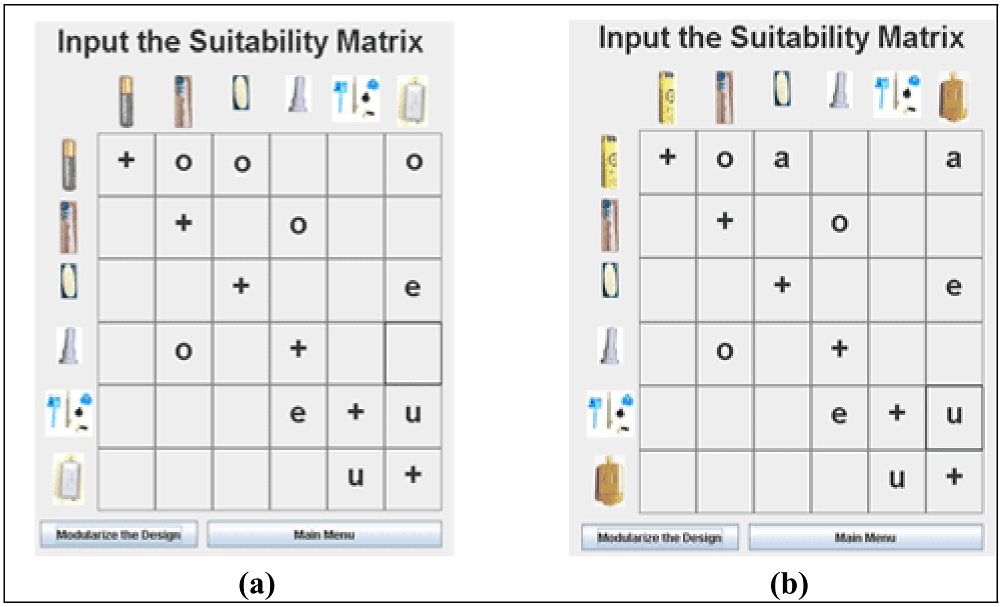
Total Number of Designs Generated: 64

DFA Index: 17.8378

☑ Select for Modularization

Figure 4-12: Conceptual Designs (a) #1 and (b) #4 are selected for modularization

In order to modularize the two conceptual design variants, the interaction and the suitability matrices need to be developed. The interaction matrix is already present in the design repository (Figure 4-7), which is stored in the MySQL table 'interaction' as the interactions amongst the various rules. The suitability matrix on the other hand, is populated by analyzing the customer needs, and determining the extent of suitability of including any two components within a module. Some components are not desired to be included along with other components to form a module. For example, brush head is not desired to be included along with the motor of the toothbrush as removable brush heads are needed to incorporate variety within the toothbrush design. Figure 4-13 shows the suitability matrices that are input for the two conceptual designs.



Input the Suitability Matrix

	+	o	o			o
		+		o		
			+			e
		o		+		
				e	+	u
					u	+

(a)

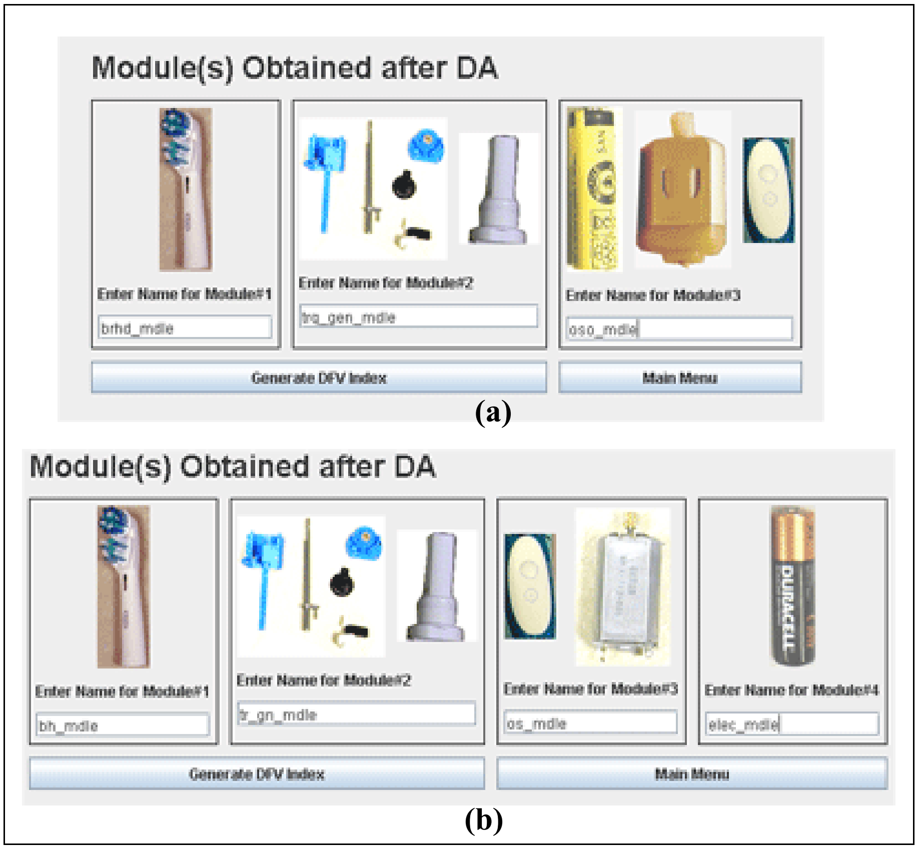
Input the Suitability Matrix

	+	o	a			a
		+		o		
			+			e
		o		+		
				e	+	u
					u	+

(b)

Figure 4-13: Suitability matrix for conceptual designs (a) #1 and (b) #4

Once both the interaction and suitability matrices are generated, the triangularized modularity matrix is developed and modules are formed. Figure 4-14 shows the modules generated for the two conceptual variants after applying the DA algorithm. The application of DA on conceptual design #4 resulted in the generation of three modules as shown in Figure 4-14a. The same steps of the DA algorithm are executed for the conceptual design #1, and Figure 4-14b shows the four modules generated.



Module(s) Obtained after DA

Enter Name for Module#1: brhd_mdle
Enter Name for Module#2: trq_gen_mdle
Enter Name for Module#3: oso_mdle

(a)

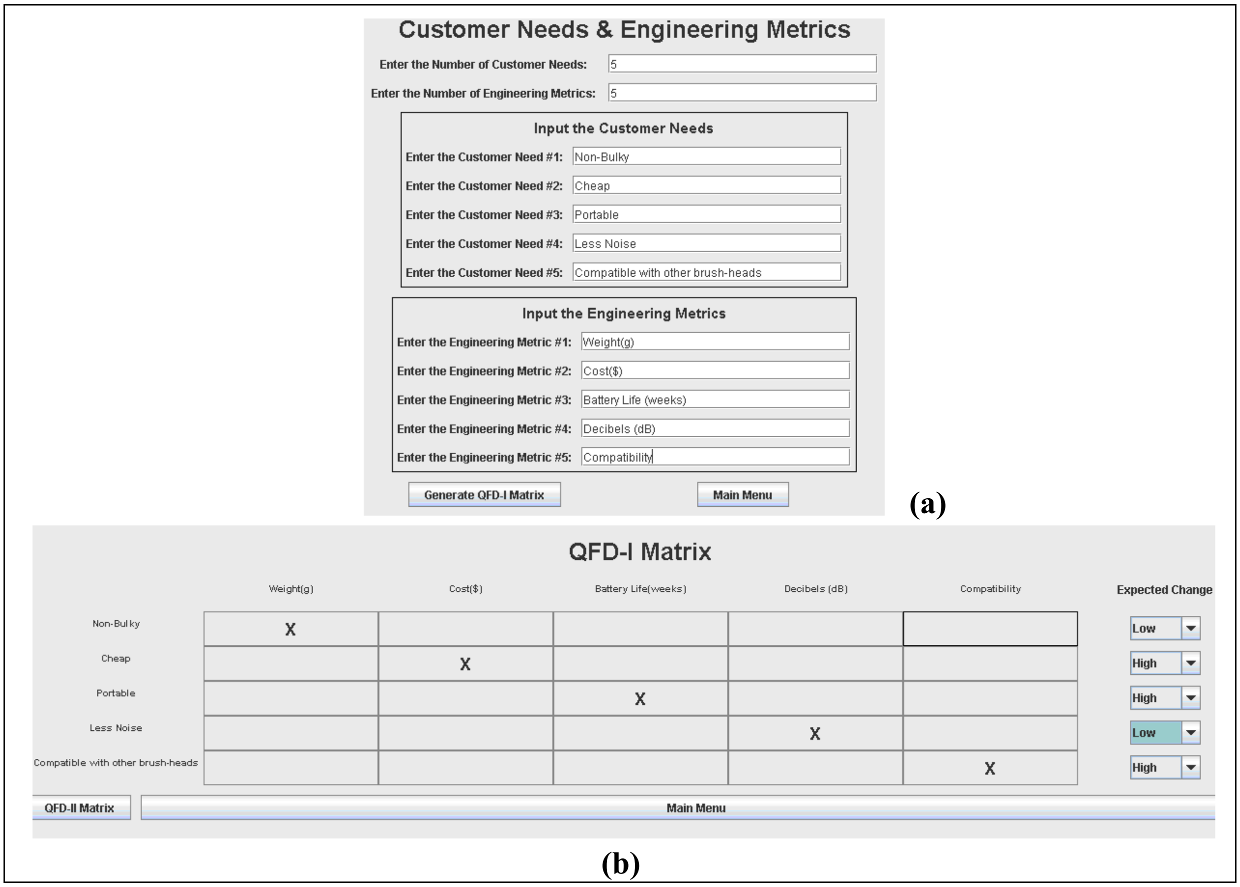
Module(s) Obtained after DA

Enter Name for Module#1: bh_mdle
Enter Name for Module#2: tr_gn_mdle
Enter Name for Module#3: os_mdle
Enter Name for Module#4: elec_mdle

(b)

Figure 4-14: Modules obtained for conceptual designs (a) #4 and (b) #1.

One major change in this conceptual variant is that the designer prefers the battery unit not to be included in the same module as the motor and the actuator. The main reason is that the batteries used are non-rechargeable and need to be replaced often. A secondary filtering needs to be done using the DFV index in order to determine the best conceptual design. The customer needs for the electronic toothbrush mentioned in Table 4-1 require to be converted to measurable engineering metrics using the QFD phase I matrix. Figures 4-15a and 4-15b demonstrate the generation of QFD Phase I matrix for the conceptual design of toothbrush.



Customer Needs & Engineering Metrics

Enter the Number of Customer Needs: 5

Enter the Number of Engineering Metrics: 5

Input the Customer Needs

Enter the Customer Need #1: Non-Bulky

Enter the Customer Need #2: Cheap

Enter the Customer Need #3: Portable

Enter the Customer Need #4: Less Noise

Enter the Customer Need #5: Compatible with other brush-heads

Input the Engineering Metrics

Enter the Engineering Metric #1: Weight(g)

Enter the Engineering Metric #2: Cost($)

Enter the Engineering Metric #3: Battery Life (weeks)

Enter the Engineering Metric #4: Decibels (dB)

Enter the Engineering Metric #5: Compatibility

(a)

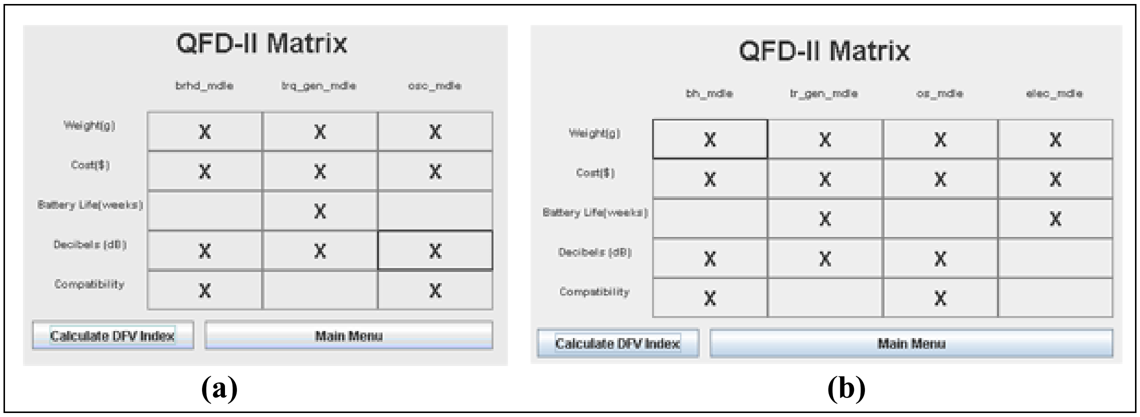
QFD-I Matrix

	Weight(g)	Cost($)	Battery Life(weeks)	Decibels (dB)	Compatibility	Expected Change
Non-Bulky	X					Low
Cheap		X				High
Portable			X			High
Less Noise				X		Low
Compatible with other brush-heads					X	High

(b)

Figure 4-15: (a) GUI to input customer needs (b) QFD phase I matrix

QFD Phase II maps the engineering metrics from phase I to the modules used in the design. The resultant matrix obtained is shown in Figure 4-16a for concept #1 and Figure 4-16b for concept #2. Based on the ranking system highlighted in Table 3-14, a quantitative value is assigned to the range of change for each customer need. For this design example, it is assumed that the designer estimates the range of change for the customer needs after a span of two years. Since it is assumed that the more the number of different modules associated with the same highly changing customer need, the higher would be the re-design effort in order to satisfy that customer need in the future. Hence, the conceptual variant having lesser DFV index value is selected as the best design. Figure 4-17 shows the QFD phase II matrices along with the DFV index values.



QFD-II Matrix

	brhd_mdle	trq_gen_mdle	osc_mdle
Weight(g)	X	X	X
Cost($)	X	X	X
Battery Life(weeks)		X	
Decibels (dB)	X	X	X
Compatibility	X		X

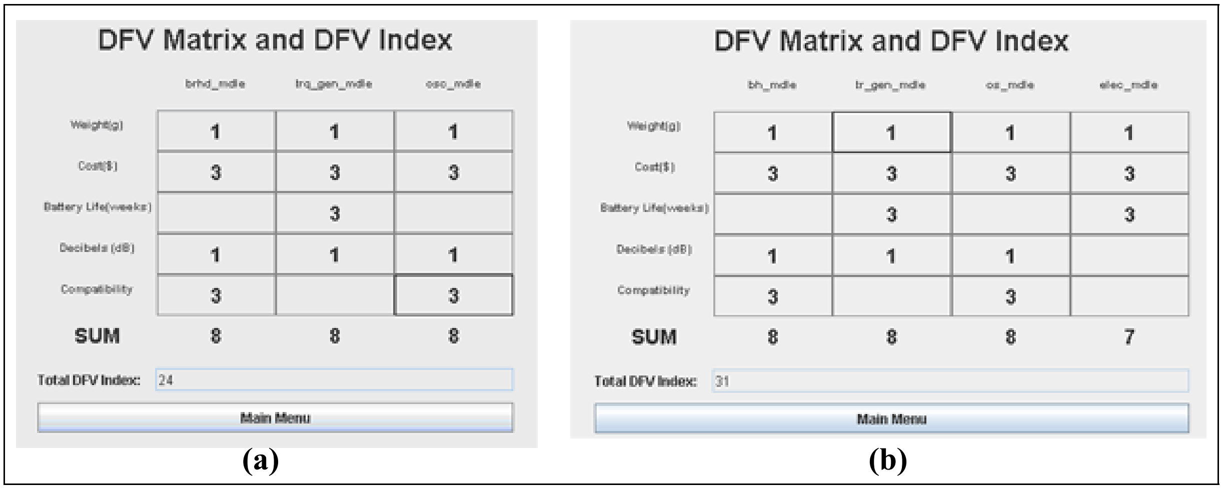
(a)

QFD-II Matrix

	bh_mdle	tr_gen_mdle	os_mdle	elec_mdle
Weight(g)	X	X	X	X
Cost($)	X	X	X	X
Battery Life(weeks)		X		X
Decibels (dB)	X	X	X	
Compatibility	X		X	

(b)

Figure 4-16: QFD Phase-II matrix for conceptual designs (a) #4 and (b) #1



DFV Matrix and DFV Index

	brhd_mdle	trq_gen_mdle	osc_mdle
Weight(g)	1	1	1
Cost($)	3	3	3
Battery Life(weeks)		3	
Decibels (dB)	1	1	1
Compatibility	3		3
SUM	8	8	8

Total DFV Index: 24

(a)

DFV Matrix and DFV Index

	bh_mdle	tr_gen_mdle	os_mdle	elec_mdle
Weight(g)	1	1	1	1
Cost($)	3	3	3	3
Battery Life(weeks)		3		3
Decibels (dB)	1	1	1	
Compatibility	3		3	
SUM	8	8	8	7

Total DFV Index: 31

(b)

Figure 4-17: DFV Matrix for conceptual designs (a) #4 and (b) #1

The DFV index value for conceptual design #4 is found to be 24, while for conceptual design #1, the DFV value is 31. Hence, conceptual design #4 is chosen as the best overall concept, since lesser redesign effort would be required within its architecture in order to meet the future customer requirements. Figure 4-18 shows the final design obtained from the proposed software framework that uses two distinct filtering criteria: DFA and DFV in order to reduce the solution space into a single conceptual design. Section 4.2 discusses the implementation of the proposed framework on the design of a mounting system for a Variable Message Sign (VMS).

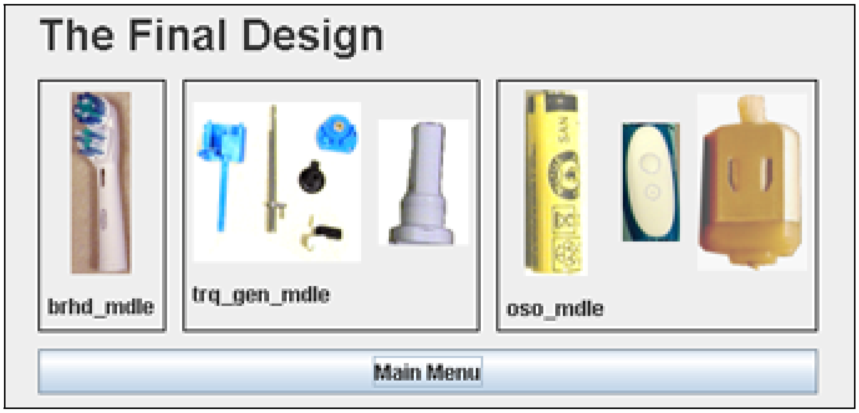


Figure 4-18: Final design obtained for the electronic toothbrush design

4.2 Design of a Mounting System for Variable Message Sign

In order to illustrate the working of the proposed framework on the VMS mounting system design, it is assumed that a hypothetical company has developed a portable VMS, and its design team needs to design a mounting system to secure the VMS (Gupta et al., 2008; Gupta and Okudan 2008d). Furthermore, the mounting system design must provide a variety of mounting options including mounting on a vehicle as well as on ground. It should also be easily setup in less than two minutes, comply with Department of Transportation (DOT) regulations, and must cost under $200 to manufacture. Figure 4-19 highlights some examples of VMS and their mounting system, which are used in numerous applications to alert commuters. It is assumed that the design repository has now been populated with components, rules and interactions corresponding to the electronic toothbrush design example discussed in Section 4.1.

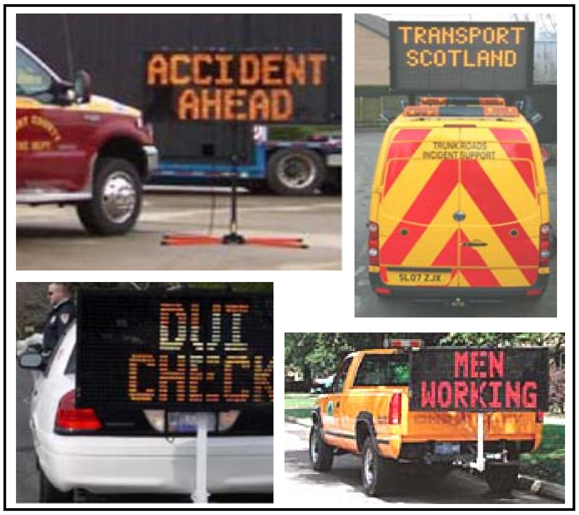
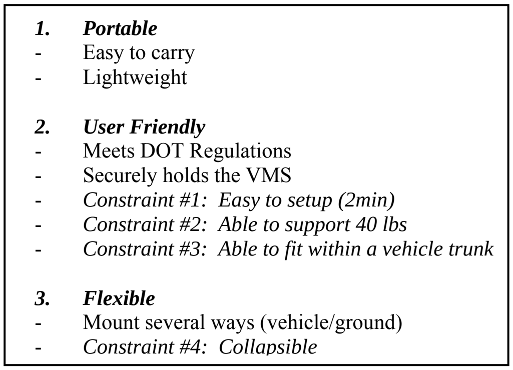


Figure 4-19: Examples of VMS and their mounting systems for various applications

As mentioned previously, foremost step in the proposed research framework is customer needs assessment. Customer needs were gathered by browsing several VMS supplier websites and through group discussion. The needs were then classified into four groups: Portable, User Friendly, Flexibility and Cost. Figure 4-20 highlights these customer needs.



1. Portable
- Easy to carry
- Lightweight

2. User Friendly
- Meets DOT Regulations
- Securely holds the VMS
- *Constraint #1: Easy to setup (2min)*
- *Constraint #2: Able to support 40 lbs*
- *Constraint #3: Able to fit within a vehicle trunk*

3. Flexible
- Mount several ways (vehicle/ground)
- *Constraint #4: Collapsible*

Figure 4-20: Hierarchical customer needs for VMS mounting system

The next step involves the preparation of a black-box model of the VMS mounting system. The main objective of the mounting system is to secure the VMS, and this objective is reflected in the black-box model (Figure 4-21). The black box model is then used to create the EMS diagram, which is obtained by decomposing the overall function into simpler sub-functions and flows. This EMS diagram of the VMS mounting system is shown in Figure 4-22, which needs to be input into the EMS GUI (Figure 3-3) in order to generate the conceptual designs by querying the design repository.

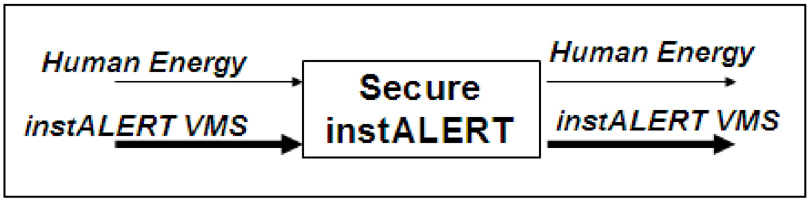


Figure 4-21: Black box for the VMS mount system

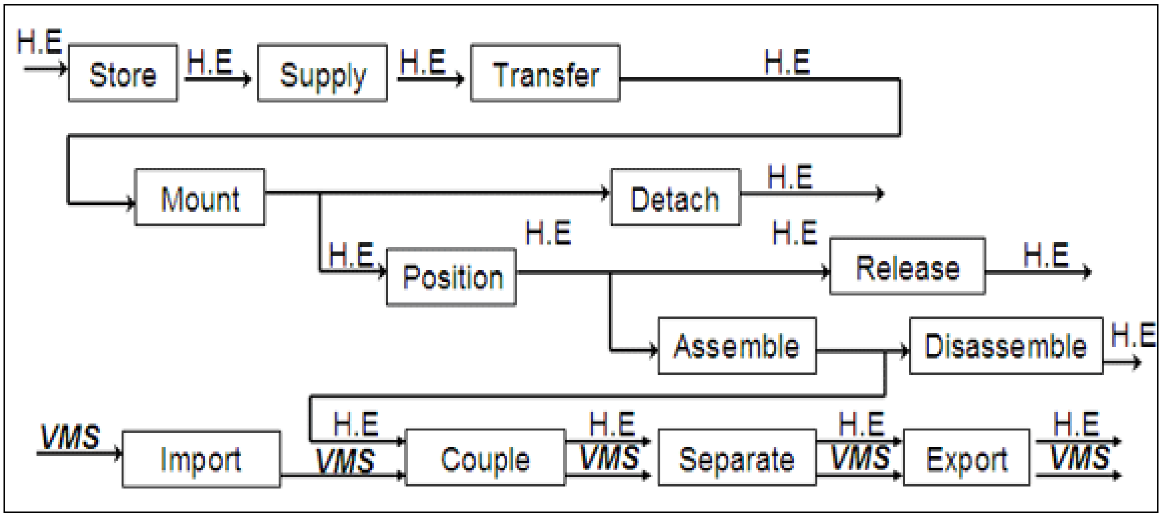
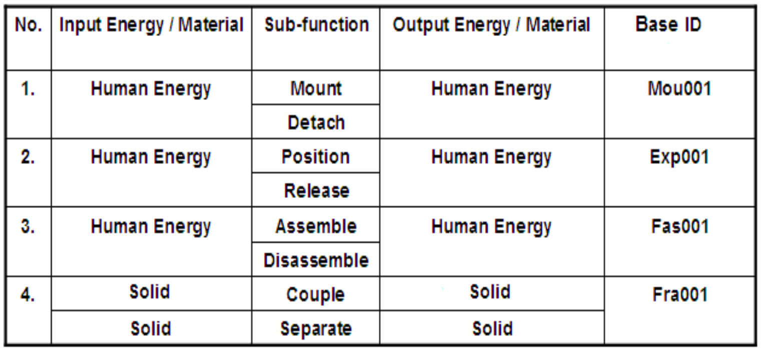


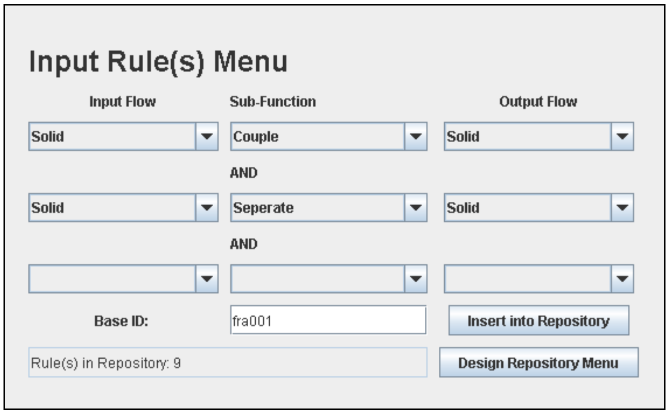
Figure 4-22: EMS model of the VMS mounting system

Analysis of EMS diagram reveals four different steps associated with the mounting system design. First step governs the use of Human Energy (HE) to fixture the base mount of the mounting system, either on the car or on ground. The second step involves utilization of HE for height adjustment of the base mount system so that VMS can be easily noticed by the passing commuter vehicles. Once the height of base mount is adjusted, VMS fixture can be assembled with the base mount. Lastly, the VMS is coupled with the VMS fixture so that it is secured thoroughly on to the ground/car. Corresponding to these steps, four graph grammar rules are generated, which are shown in Table 4-2. Figure 4-23 shows the GUI for inputting the rules into the design repository, while Figure 4-24 shows the GUI for viewing the saved rules.

Table 4-2: Graph grammar rules for the VMS mounting system



No.	Input Energy / Material	Sub-function	Output Energy / Material	Base ID
1.	Human Energy	Mount Detach	Human Energy	Mou001
2.	Human Energy	Position Release	Human Energy	Exp001
3.	Human Energy	Assemble Disassemble	Human Energy	Fas001
4.	Solid Solid	Couple Separate	Solid Solid	Fra001



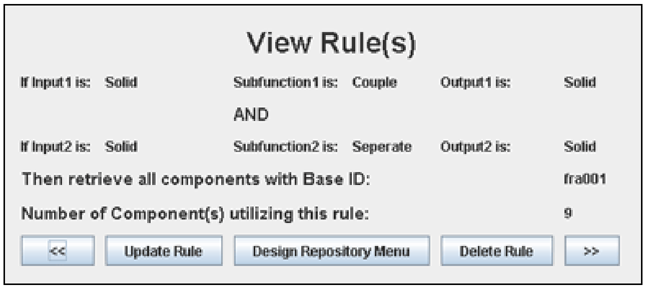
Input Rule(s) Menu

Input Flow	Sub-Function	Output Flow
Solid	Couple	Solid
	AND	
Solid	Seperate	Solid
	AND	

Base ID: fra001

Rule(s) in Repository: 9

Figure 4-23: GUI for inputting the VMS rules into the design repository



View Rule(s)

If Input1 is: Solid   Subfunction1 is: Couple   Output1 is: Solid

AND

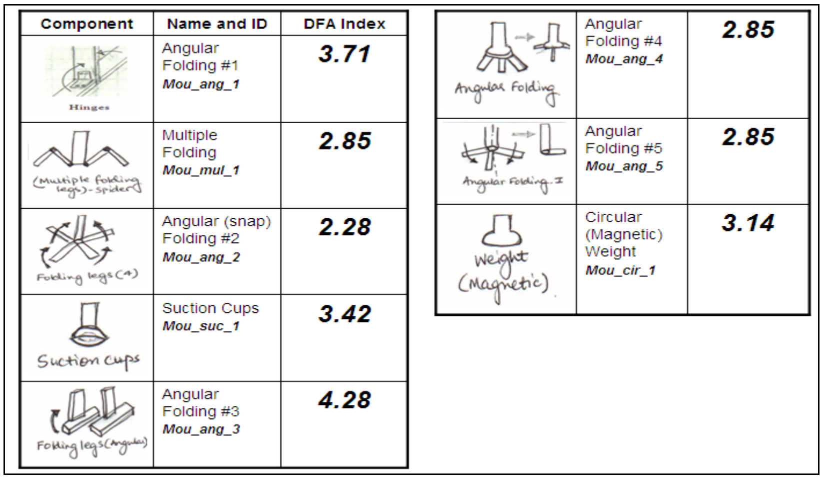
If Input2 is: Solid   Subfunction2 is: Seperate   Output2 is: Solid

Then retrieve all components with Base ID: fra001

Number of Component(s) utilizing this rule: 9

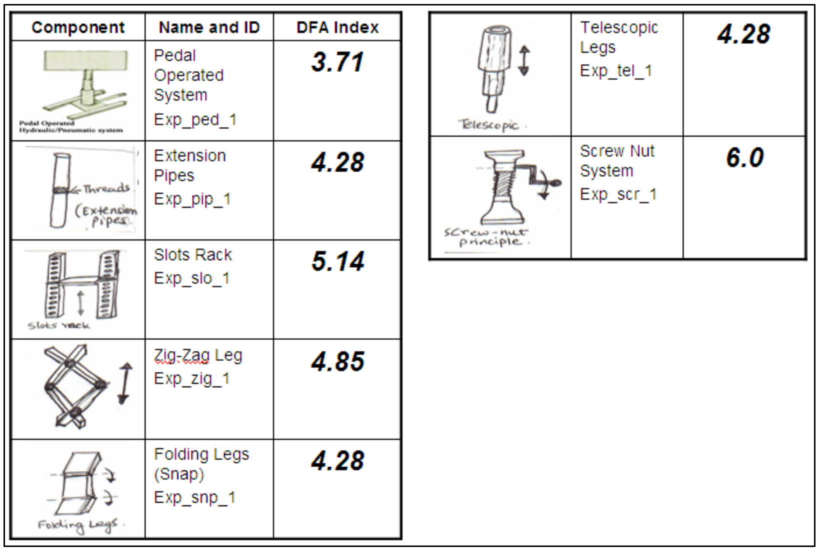
Figure 4-24: GUI for viewing the VMS rules saved in the design repository

The design team then begins to brainstorm all the components that satisfy the different sub-functions and flows shown in the EMS diagram. These components will be stored in the design repository for later reuse. After identifying the components, a unique ID is associated with each of them and their DFA indexes are calculated based on the 13-point ranking system (Hsu et al., 1998; Rampersad, 1994). Figures 4-25 to 4-28 show the identified components for the mounting system along with their associated image, rule, unique ID and DFA index value. Figure 4-29 shows the mySQL table corresponding to the stored components along with their images and DFA index values. Figure 4-30 shows the GUI for inputting the components into the design repository, while Figure 4-31 shows the GUI for viewing the components.



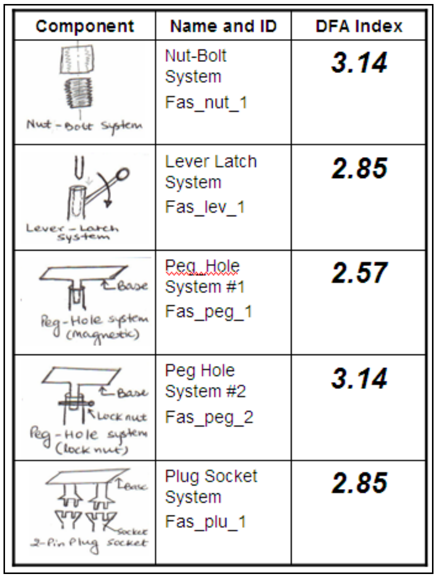
Component	Name and ID	DFA Index
Hinges	Angular Folding #1 *Mou_ang_1*	3.71
(Multiple folding legs) - spider	Multiple Folding *Mou_mul_1*	2.85
Folding legs (4)	Angular (snap) Folding #2 *Mou_ang_2*	2.28
Suction cups	Suction Cups *Mou_suc_1*	3.42
Folding legs (Angular)	Angular Folding #3 *Mou_ang_3*	4.28
Angular Folding	Angular Folding #4 *Mou_ang_4*	2.85
Angular Folding - I	Angular Folding #5 *Mou_ang_5*	2.85
Weight (Magnetic)	Circular (Magnetic) Weight *Mou_cir_1*	3.14

Figure 4-25: Components for "mount to ground/car" function



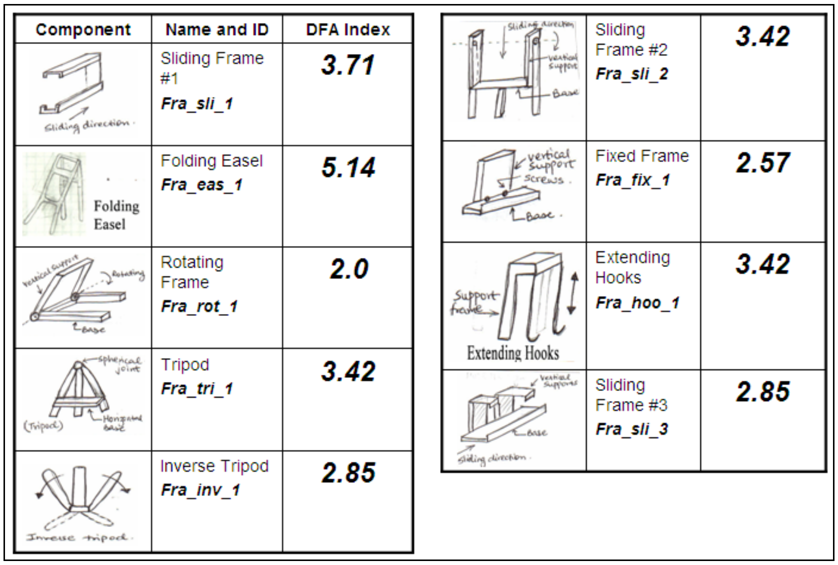
Component	Name and ID	DFA Index
Pedal Operated Hydraulic/Pneumatic system	Pedal Operated System Exp_ped_1	3.71
Threads (Extension pipes)	Extension Pipes Exp_pip_1	4.28
Slots rack	Slots Rack Exp_slo_1	5.14
	Zig-Zag Leg Exp_zig_1	4.85
Folding Legs.	Folding Legs (Snap) Exp_snp_1	4.28
Telescopic.	Telescopic Legs Exp_tel_1	4.28
screw-nut principle.	Screw Nut System Exp_scr_1	6.0

Figure 4-26: Components for “extending/collapsing” function



Component	Name and ID	DFA Index
Nut-Bolt System	Nut-Bolt System Fas_nut_1	3.14
Lever-Latch system	Lever Latch System Fas_lev_1	2.85
Base Peg-Hole system (magnetic)	Peg Hole System #1 Fas_peg_1	2.57
Base Lock nut Peg-Hole system (lock nut)	Peg Hole System #2 Fas_peg_2	3.14
Base Socket 2-Pin Plug socket	Plug Socket System Fas_plu_1	2.85

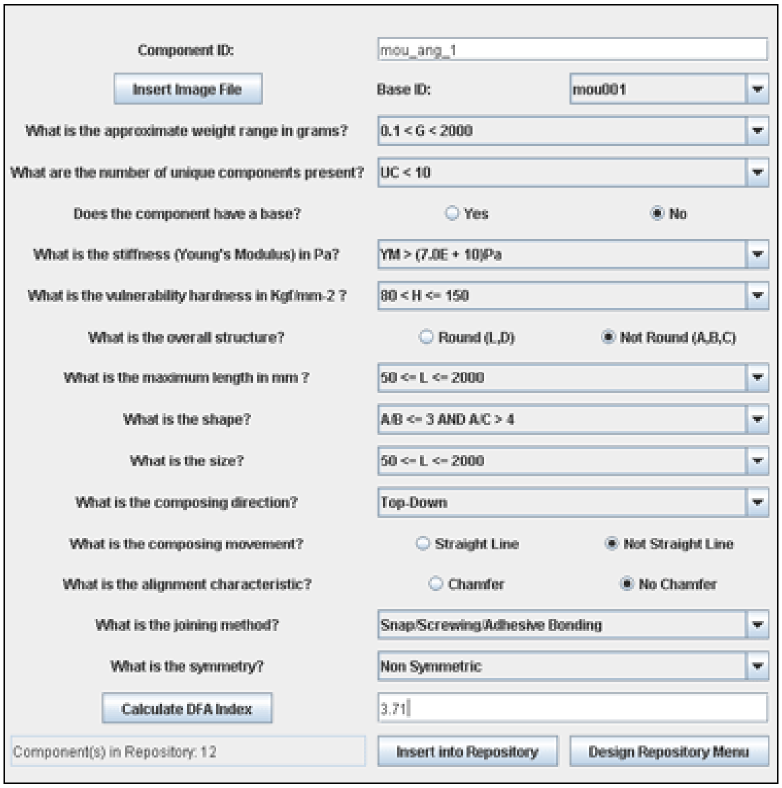
Figure 4-27: Components for “attaching sign to base frame” function



Sliding direction
Folding Easel
vertical support
Rotating
Base
spherical joint
(Tripod)
Horizontal base
Inverse tripod
Sliding direction
vertical support
Base
vertical support
screws
Base
Support hooks
Extending Hooks
vertical supports
Base
Sliding direction

Component	Name and ID	DFA Index
	Sliding Frame #1 *Fra_sli_1*	**3.71**
	Folding Easel *Fra_eas_1*	**5.14**
	Rotating Frame *Fra_rot_1*	**2.0**
	Tripod *Fra_tri_1*	**3.42**
	Inverse Tripod *Fra_inv_1*	**2.85**
	Sliding Frame #2 *Fra_sli_2*	**3.42**
	Fixed Frame *Fra_fix_1*	**2.57**
	Extending Hooks *Fra_hoo_1*	**3.42**
	Sliding Frame #3 *Fra_sli_3*	**2.85**

Figure 4-28: DFA indexes for “holding the sign” function



Component ID: mou_ang_1

Insert Image File — Base ID: mou001

What is the approximate weight range in grams? 0.1 < G < 2000

What are the number of unique components present? UC < 10

Does the component have a base? ○ Yes ● No

What is the stiffness (Young's Modulus) in Pa? YM > (7.0E + 10)Pa

What is the vulnerability hardness in Kgf/mm-2 ? 80 < H <= 150

What is the overall structure? ○ Round (L,D) ● Not Round (A,B,C)

What is the maximum length in mm ? 50 <= L <= 2000

What is the shape? A/B <= 3 AND A/C > 4

What is the size? 50 <= L <= 2000

What is the composing direction? Top-Down

What is the composing movement? ○ Straight Line ● Not Straight Line

What is the alignment characteristic? ○ Chamfer ● No Chamfer

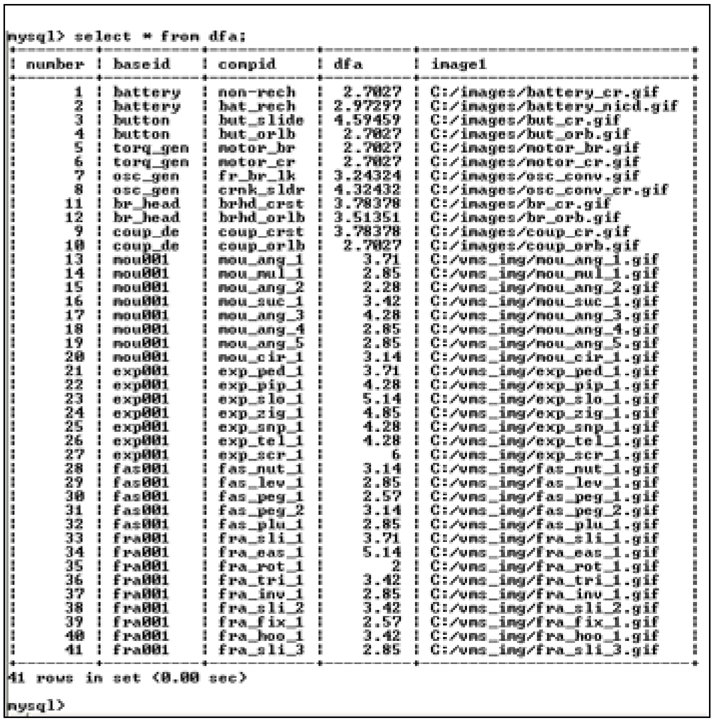
What is the joining method? Snap/Screwing/Adhesive Bonding

What is the symmetry? Non Symmetric

Calculate DFA Index — 3.71

Component(s) in Repository: 12

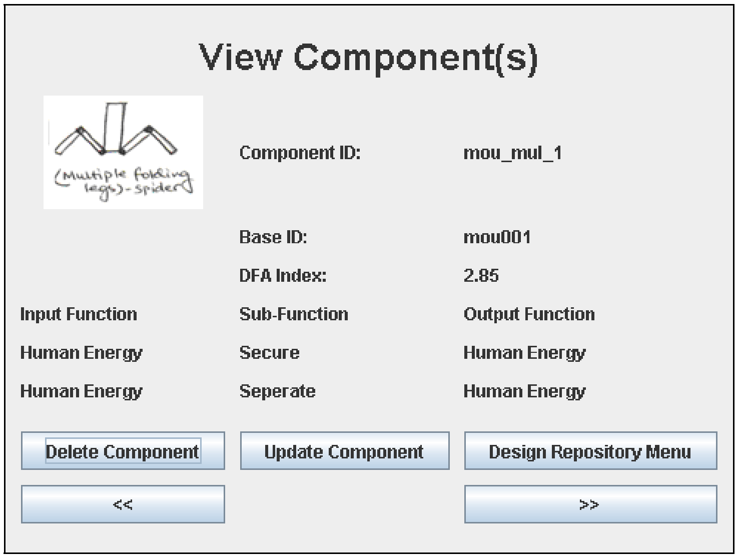
Figure 4-29: GUI for inputting the components in the design repository



```
mysql> select * from dfa;
+--------+----------+-----------+---------+----------------------------+
¦ number ¦ baseid   ¦ compid    ¦ dfa     ¦ image1                     ¦
+--------+----------+-----------+---------+----------------------------+
¦      1 ¦ battery  ¦ non-rech  ¦  2.7027 ¦ C:/images/battery_cr.gif   ¦
¦      2 ¦ battery  ¦ bat_rech  ¦ 2.97297 ¦ C:/images/battery_nicd.gif ¦
¦      3 ¦ button   ¦ but_slide ¦ 4.59459 ¦ C:/images/but_cr.gif       ¦
¦      4 ¦ button   ¦ but_orlb  ¦  2.7027 ¦ C:/images/but_orb.gif      ¦
¦      5 ¦ torq_gen ¦ motor_br  ¦  2.7027 ¦ C:/images/motor_br.gif     ¦
¦      6 ¦ torq_gen ¦ motor_cr  ¦  2.7027 ¦ C:/images/motor_cr.gif     ¦
¦      7 ¦ osc_gen  ¦ fr_br_lk  ¦ 3.24324 ¦ C:/images/osc_conv.gif     ¦
¦      8 ¦ osc_gen  ¦ crnk_sldr ¦ 4.32432 ¦ C:/images/osc_conv_cr.gif  ¦
¦     11 ¦ br_head  ¦ brhd_crst ¦ 3.78378 ¦ C:/images/br_cr.gif        ¦
¦     12 ¦ br_head  ¦ brhd_orlb ¦ 3.51351 ¦ C:/images/br_orb.gif       ¦
¦      9 ¦ coup_de  ¦ coup_crst ¦ 3.78378 ¦ C:/images/coup_cr.gif      ¦
¦     10 ¦ coup_de  ¦ coup_orlb ¦  2.7027 ¦ C:/images/coup_orb.gif     ¦
¦     13 ¦ mou001   ¦ mou_ang_1 ¦    3.71 ¦ C:/vms_img/mou_ang_1.gif   ¦
¦     14 ¦ mou001   ¦ mou_mul_1 ¦    2.85 ¦ C:/vms_img/mou_mul_1.gif   ¦
¦     15 ¦ mou001   ¦ mou_ang_2 ¦    2.28 ¦ C:/vms_img/mou_ang_2.gif   ¦
¦     16 ¦ mou001   ¦ mou_suc_1 ¦    3.42 ¦ C:/vms_img/mou_suc_1.gif   ¦
¦     17 ¦ mou001   ¦ mou_ang_3 ¦    4.28 ¦ C:/vms_img/mou_ang_3.gif   ¦
¦     18 ¦ mou001   ¦ mou_ang_4 ¦    2.85 ¦ C:/vms_img/mou_ang_4.gif   ¦
¦     19 ¦ mou001   ¦ mou_ang_5 ¦    2.85 ¦ C:/vms_img/mou_ang_5.gif   ¦
¦     20 ¦ mou001   ¦ mou_cir_1 ¦    3.14 ¦ C:/vms_img/mou_cir_1.gif   ¦
¦     21 ¦ exp001   ¦ exp_ped_1 ¦    3.71 ¦ C:/vms_img/exp_ped_1.gif   ¦
¦     22 ¦ exp001   ¦ exp_pip_1 ¦    4.28 ¦ C:/vms_img/exp_pip_1.gif   ¦
¦     23 ¦ exp001   ¦ exp_slo_1 ¦    5.14 ¦ C:/vms_img/exp_slo_1.gif   ¦
¦     24 ¦ exp001   ¦ exp_zig_1 ¦    4.85 ¦ C:/vms_img/exp_zig_1.gif   ¦
¦     25 ¦ exp001   ¦ exp_snp_1 ¦    4.28 ¦ C:/vms_img/exp_snp_1.gif   ¦
¦     26 ¦ exp001   ¦ exp_tel_1 ¦    4.28 ¦ C:/vms_img/exp_tel_1.gif   ¦
¦     27 ¦ exp001   ¦ exp_scr_1 ¦       6 ¦ C:/vms_img/exp_scr_1.gif   ¦
¦     28 ¦ fas001   ¦ fas_nut_1 ¦    3.14 ¦ C:/vms_img/fas_nut_1.gif   ¦
¦     29 ¦ fas001   ¦ fas_lev_1 ¦    2.85 ¦ C:/vms_img/fas_lev_1.gif   ¦
¦     30 ¦ fas001   ¦ fas_peg_1 ¦    2.57 ¦ C:/vms_img/fas_peg_1.gif   ¦
¦     31 ¦ fas001   ¦ fas_peg_2 ¦    3.14 ¦ C:/vms_img/fas_peg_2.gif   ¦
¦     32 ¦ fas001   ¦ fas_plu_1 ¦    2.85 ¦ C:/vms_img/fas_plu_1.gif   ¦
¦     33 ¦ fra001   ¦ fra_sli_1 ¦    3.71 ¦ C:/vms_img/fra_sli_1.gif   ¦
¦     34 ¦ fra001   ¦ fra_eas_1 ¦    5.14 ¦ C:/vms_img/fra_eas_1.gif   ¦
¦     35 ¦ fra001   ¦ fra_rot_1 ¦       2 ¦ C:/vms_img/fra_rot_1.gif   ¦
¦     36 ¦ fra001   ¦ fra_tri_1 ¦    3.42 ¦ C:/vms_img/fra_tri_1.gif   ¦
¦     37 ¦ fra001   ¦ fra_inv_1 ¦    2.85 ¦ C:/vms_img/fra_inv_1.gif   ¦
¦     38 ¦ fra001   ¦ fra_sli_2 ¦    3.42 ¦ C:/vms_img/fra_sli_2.gif   ¦
¦     39 ¦ fra001   ¦ fra_fix_1 ¦    2.57 ¦ C:/vms_img/fra_fix_1.gif   ¦
¦     40 ¦ fra001   ¦ fra_hoo_1 ¦    3.42 ¦ C:/vms_img/fra_hoo_1.gif   ¦
¦     41 ¦ fra001   ¦ fra_sli_3 ¦    2.85 ¦ C:/vms_img/fra_sli_3.gif   ¦
+--------+----------+-----------+---------+----------------------------+
41 rows in set (0.00 sec)

mysql>
```

Figure 4-30: mySQL table showing all components in the design repository



View Component(s)

Component ID: mou_mul_1

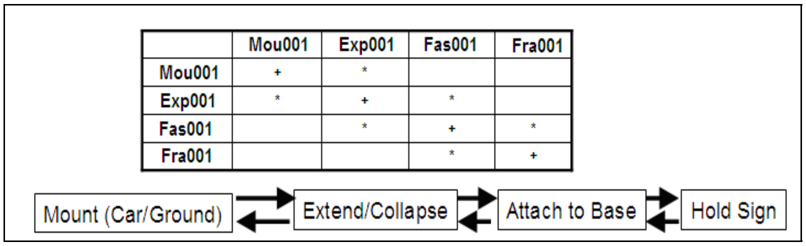
Base ID: mou001

DFA Index: 2.85

Input Function	Sub-Function	Output Function
Human Energy	Secure	Human Energy
Human Energy	Seperate	Human Energy

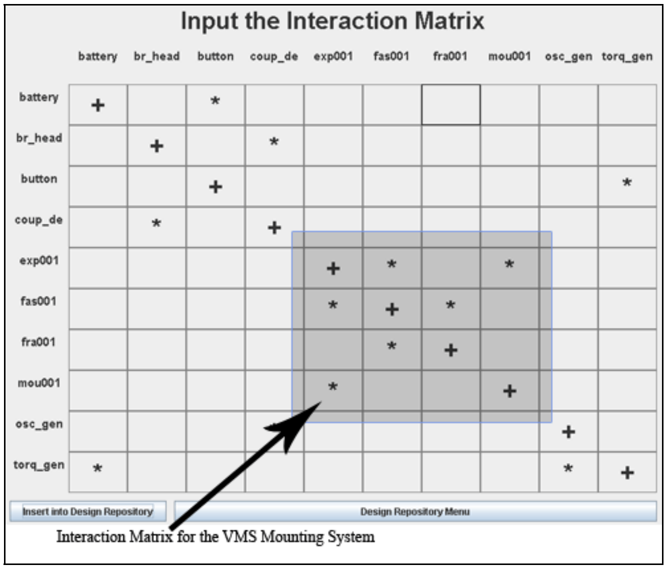
Figure 4-31: GUI for viewing the VMS components stored in the design repository

As observed from Figure 4-30, the mySQL table contains the previous 12 components which were obtained for the electronic toothbrush design example. Therefore, in total, the ‘DFA’ table contains 41 components. The design team also analyzed the interactions between the four rules in order to populate the ‘interactions’ table as shown in Figure 4-31. Figure 4-32 shows the GUI for inputting the interactions into the design repository. The GUI already contains the interactions between the six rules that were discussed for the electronic toothbrush in Section 4.1. The design repository is now assumed to be populated with rules, interactions and component specifications.



	Mou001	Exp001	Fas001	Fra001
Mou001	+	*		
Exp001	*	+	*	
Fas001		*	+	*
Fra001			*	+

Figure 4-32: Interaction between the four graph grammar rules



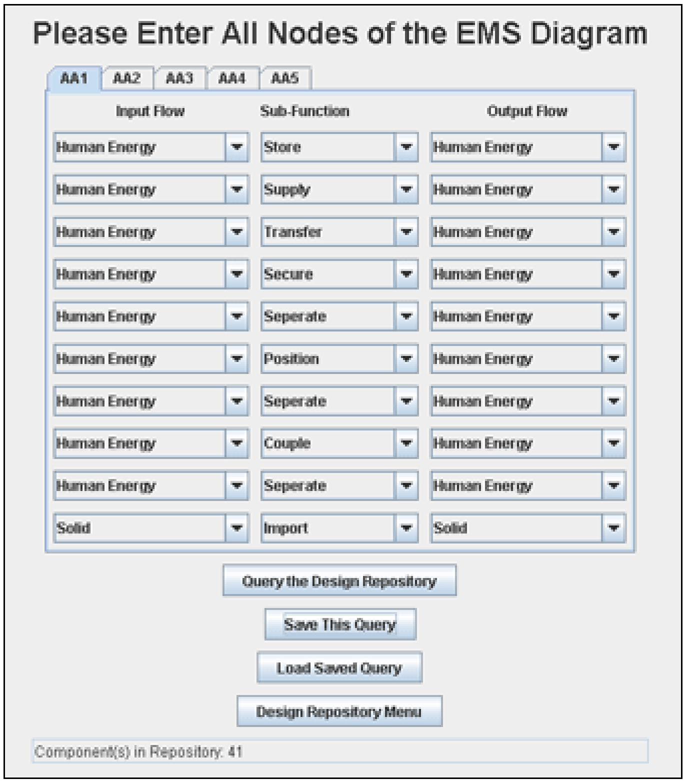
Input the Interaction Matrix

	battery	br_head	button	coup_de	exp001	fas001	fra001	mou001	osc_gen	torq_gen
battery	+		*							
br_head		+		*						
button			+							*
coup_de		*		+						
exp001					+	*		*		
fas001					*	+	*			
fra001						*	+			
mou001					*			+		
osc_gen									+	
torq_gen	*								*	+

Insert into Design Repository
Design Repository Menu
Interaction Matrix for the VMS Mounting System

Figure 4-33: GUI for inputting the interaction matrix into the design repository

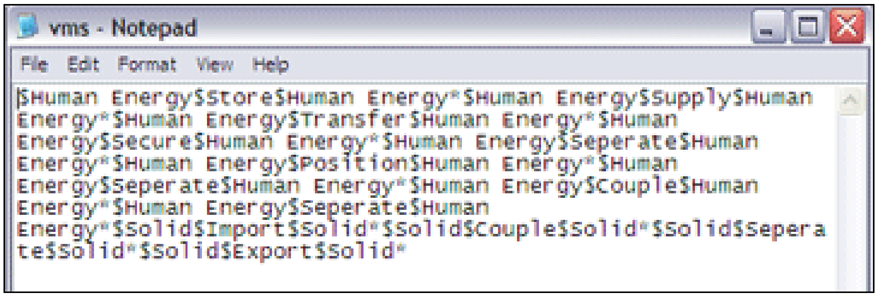
The next step involves the comparison of the EMS model with the rules stored in design repository. Figure 4-34 shows the GUI for inputting the EMS diagram (Figure 4-22) for comparison with the rules table, while Figure 4-35 shows the saved query.


Please Enter All Nodes of the EMS Diagram
AA1
AA2
AA3
AA4
AA5
Query the Design Repository
Save This Query
Load Saved Query
Design Repository Menu

Input Flow	Sub-Function	Output Flow
Human Energy	Store	Human Energy
Human Energy	Supply	Human Energy
Human Energy	Transfer	Human Energy
Human Energy	Secure	Human Energy
Human Energy	Seperate	Human Energy
Human Energy	Position	Human Energy
Human Energy	Seperate	Human Energy
Human Energy	Couple	Human Energy
Human Energy	Seperate	Human Energy
Solid	Import	Solid

Component(s) in Repository: 41

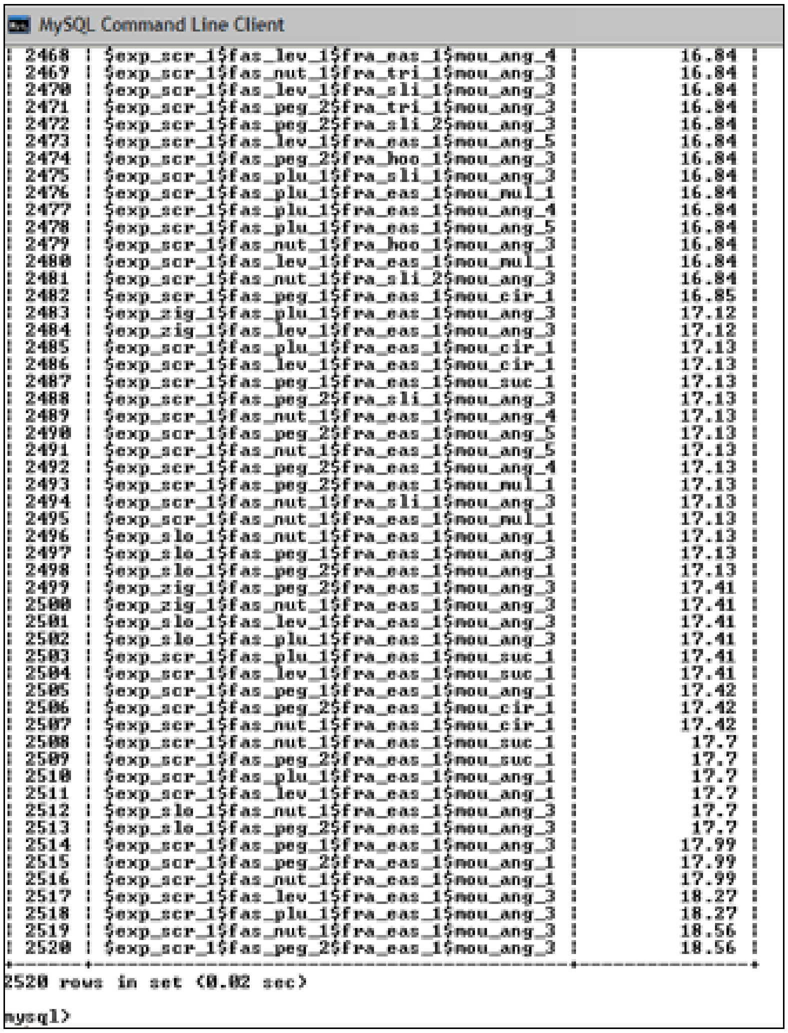
Figure 4-34: GUI for inputting nodes of the EMS diagram



```
$Human Energy$Store$Human Energy*$Human Energy$Supply$Human
Energy*$Human Energy$Transfer$Human Energy*$Human
Energy$Secure$Human Energy*$Human Energy$Seperate$Human
Energy*$Human Energy$Position$Human Energy*$Human
Energy$Seperate$Human Energy*$Human Energy$Couple$Human
Energy*$Human Energy$Seperate$Human
Energy*$Solid$Import$Solid*$Solid$Couple$Solid*$Solid$Sepera
te$Solid*$Solid$Export$Solid*
```

Figure 4-35: Notepad file showing the saved query for the VMS mounting system

In the design repository, 9 components are associated with ‘holding the sign’ rule, 5 components with ‘attaching the sign to the base frame’ rule, 7 components with ‘extending or collapsing’ rule, and 8 components with ‘mount to ground/vehicle’ rule. In total, this allocates for 2520 conceptual design combinations satisfying overall function of “securing VMS”. Figure 4-36 shows the temporary mySQL table: ‘final_info’, which highlights the 2520 design combinations generated automatically as a result of querying the design repository. It was also observed that the aggregated DFA index value ranges from 10.56 to 18.56.

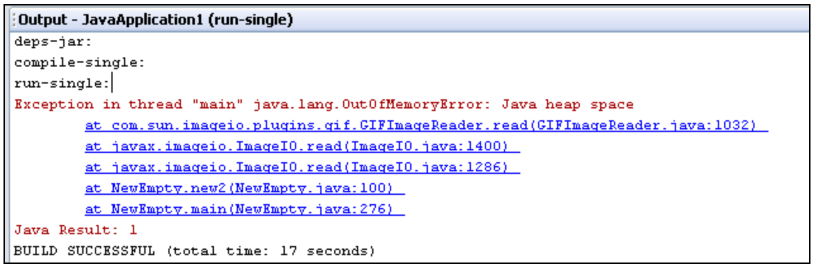


```
| 2468 | $exp_scr_1$fas_lev_1$fra_eas_1$mou_ang_4 |          16.84 |
| 2469 | $exp_scr_1$fas_nut_1$fra_tri_1$mou_ang_3 |          16.84 |
| 2470 | $exp_scr_1$fas_lev_1$fra_sli_1$mou_ang_3 |          16.84 |
| 2471 | $exp_scr_1$fas_peg_2$fra_tri_1$mou_ang_3 |          16.84 |
| 2472 | $exp_scr_1$fas_peg_2$fra_sli_2$mou_ang_3 |          16.84 |
| 2473 | $exp_scr_1$fas_lev_1$fra_eas_1$mou_ang_5 |          16.84 |
| 2474 | $exp_scr_1$fas_peg_2$fra_hoo_1$mou_ang_3 |          16.84 |
| 2475 | $exp_scr_1$fas_plu_1$fra_sli_1$mou_ang_3 |          16.84 |
| 2476 | $exp_scr_1$fas_plu_1$fra_eas_1$mou_mul_1 |          16.84 |
| 2477 | $exp_scr_1$fas_plu_1$fra_eas_1$mou_ang_4 |          16.84 |
| 2478 | $exp_scr_1$fas_plu_1$fra_eas_1$mou_ang_5 |          16.84 |
| 2479 | $exp_scr_1$fas_nut_1$fra_hoo_1$mou_ang_3 |          16.84 |
| 2480 | $exp_scr_1$fas_lev_1$fra_eas_1$mou_mul_1 |          16.84 |
| 2481 | $exp_scr_1$fas_nut_1$fra_sli_2$mou_ang_3 |          16.84 |
| 2482 | $exp_scr_1$fas_peg_1$fra_eas_1$mou_cir_1 |          16.85 |
| 2483 | $exp_zig_1$fas_plu_1$fra_eas_1$mou_ang_3 |          17.12 |
| 2484 | $exp_zig_1$fas_lev_1$fra_eas_1$mou_ang_3 |          17.12 |
| 2485 | $exp_scr_1$fas_plu_1$fra_eas_1$mou_cir_1 |          17.13 |
| 2486 | $exp_scr_1$fas_lev_1$fra_eas_1$mou_cir_1 |          17.13 |
| 2487 | $exp_scr_1$fas_peg_1$fra_eas_1$mou_suc_1 |          17.13 |
| 2488 | $exp_scr_1$fas_peg_2$fra_sli_1$mou_ang_3 |          17.13 |
| 2489 | $exp_scr_1$fas_nut_1$fra_eas_1$mou_ang_4 |          17.13 |
| 2490 | $exp_scr_1$fas_peg_2$fra_eas_1$mou_ang_5 |          17.13 |
| 2491 | $exp_scr_1$fas_nut_1$fra_eas_1$mou_ang_5 |          17.13 |
| 2492 | $exp_scr_1$fas_peg_2$fra_eas_1$mou_ang_4 |          17.13 |
| 2493 | $exp_scr_1$fas_peg_2$fra_eas_1$mou_mul_1 |          17.13 |
| 2494 | $exp_scr_1$fas_nut_1$fra_sli_1$mou_ang_3 |          17.13 |
| 2495 | $exp_scr_1$fas_nut_1$fra_eas_1$mou_mul_1 |          17.13 |
| 2496 | $exp_slo_1$fas_nut_1$fra_eas_1$mou_ang_1 |          17.13 |
| 2497 | $exp_slo_1$fas_peg_1$fra_eas_1$mou_ang_3 |          17.13 |
| 2498 | $exp_slo_1$fas_peg_2$fra_eas_1$mou_ang_1 |          17.13 |
| 2499 | $exp_zig_1$fas_peg_2$fra_eas_1$mou_ang_3 |          17.41 |
| 2500 | $exp_zig_1$fas_nut_1$fra_eas_1$mou_ang_3 |          17.41 |
| 2501 | $exp_slo_1$fas_lev_1$fra_eas_1$mou_ang_3 |          17.41 |
| 2502 | $exp_slo_1$fas_plu_1$fra_eas_1$mou_ang_3 |          17.41 |
| 2503 | $exp_scr_1$fas_plu_1$fra_eas_1$mou_suc_1 |          17.41 |
| 2504 | $exp_scr_1$fas_lev_1$fra_eas_1$mou_suc_1 |          17.41 |
| 2505 | $exp_scr_1$fas_peg_1$fra_eas_1$mou_ang_1 |          17.42 |
| 2506 | $exp_scr_1$fas_peg_2$fra_eas_1$mou_cir_1 |          17.42 |
| 2507 | $exp_scr_1$fas_nut_1$fra_eas_1$mou_cir_1 |          17.42 |
| 2508 | $exp_scr_1$fas_nut_1$fra_eas_1$mou_suc_1 |           17.7 |
| 2509 | $exp_scr_1$fas_peg_2$fra_eas_1$mou_suc_1 |           17.7 |
| 2510 | $exp_scr_1$fas_plu_1$fra_eas_1$mou_ang_1 |           17.7 |
| 2511 | $exp_scr_1$fas_lev_1$fra_eas_1$mou_ang_1 |           17.7 |
| 2512 | $exp_slo_1$fas_nut_1$fra_eas_1$mou_ang_3 |           17.7 |
| 2513 | $exp_slo_1$fas_peg_2$fra_eas_1$mou_ang_3 |           17.7 |
| 2514 | $exp_scr_1$fas_peg_1$fra_eas_1$mou_ang_3 |          17.99 |
| 2515 | $exp_scr_1$fas_peg_2$fra_eas_1$mou_ang_1 |          17.99 |
| 2516 | $exp_scr_1$fas_nut_1$fra_eas_1$mou_ang_1 |          17.99 |
| 2517 | $exp_scr_1$fas_lev_1$fra_eas_1$mou_ang_3 |          18.27 |
| 2518 | $exp_scr_1$fas_plu_1$fra_eas_1$mou_ang_3 |          18.27 |
| 2519 | $exp_scr_1$fas_nut_1$fra_eas_1$mou_ang_3 |          18.56 |
| 2520 | $exp_scr_1$fas_peg_2$fra_eas_1$mou_ang_3 |          18.56 |
+------+------------------------------------------+----------------+
2520 rows in set (0.02 sec)

mysql>
```

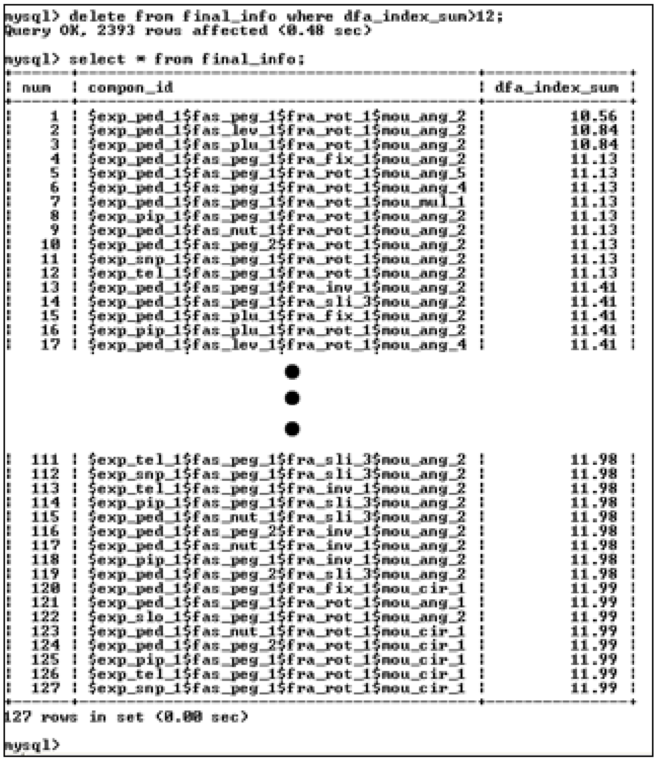
Figure 4-36: mySQL table final_info showing the 2520 possible combinations

The GUI which shows all the design combinations indicated an "out of memory" error (Figure 4-37) due to large memory needed to load image files corresponding to each conceptual combination. The current software is being tested on a personal computer, and therefore, due to the limited resources, it was decided to filter the final_info table and select only those combinations which had their aggregated DFA index less than 12 (Figure 4-38).



```
deps-jar:
compile-single:
run-single:
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at com.sun.imageio.plugins.gif.GIFImageReader.read(GIFImageReader.java:1032)
        at javax.imageio.ImageIO.read(ImageIO.java:1400)
        at javax.imageio.ImageIO.read(ImageIO.java:1286)
        at NewEmpty.new2(NewEmpty.java:100)
        at NewEmpty.main(NewEmpty.java:276)
Java Result: 1
BUILD SUCCESSFUL (total time: 17 seconds)
```

Figure 4-37: Error due to the large memory required in loading component images



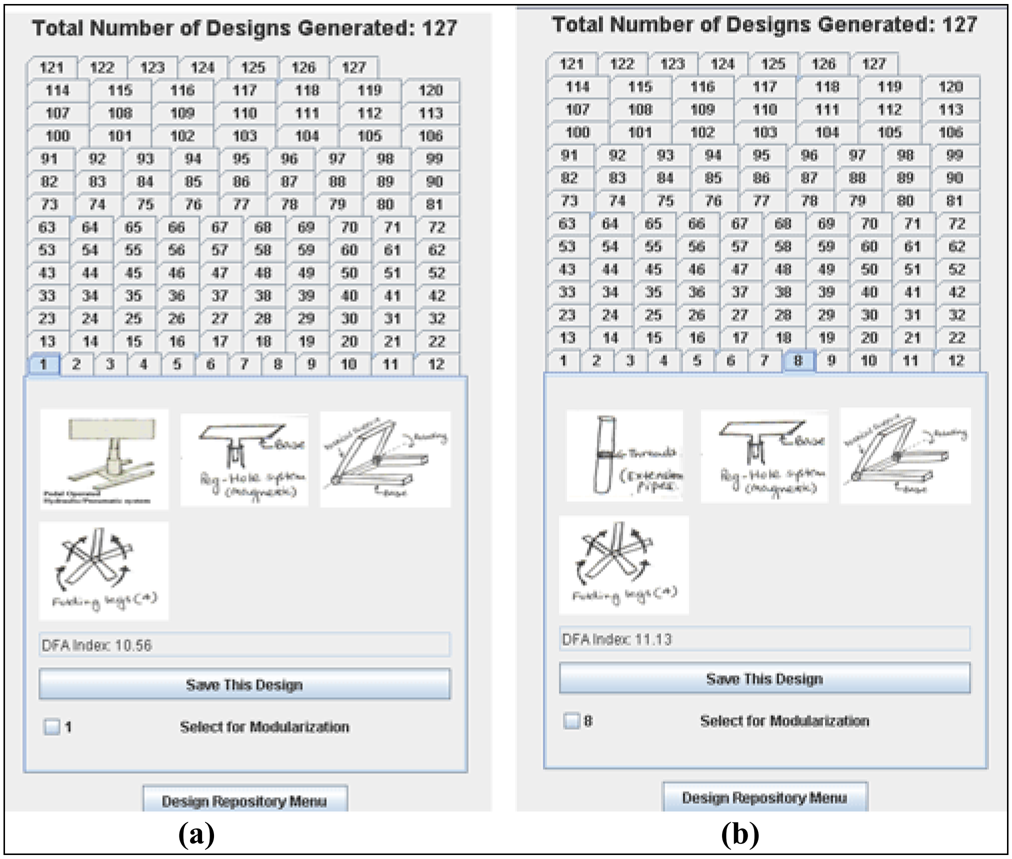
```
mysql> delete from final_info where dfa_index_sum>12;
Query OK, 2393 rows affected (0.48 sec)

mysql> select * from final_info;
+-------+--------------------------------------------+---------------+
| num   | compon_id                                  | dfa_index_sum |
+-------+--------------------------------------------+---------------+
|     1 | $exp_ped_1$fas_peg_1$fra_rot_1$mou_ang_2   |         10.56 |
|     2 | $exp_ped_1$fas_lev_1$fra_rot_1$mou_ang_2   |         10.84 |
|     3 | $exp_ped_1$fas_plu_1$fra_rot_1$mou_ang_2   |         10.84 |
|     4 | $exp_ped_1$fas_peg_1$fra_fix_1$mou_ang_2   |         11.13 |
|     5 | $exp_ped_1$fas_peg_1$fra_rot_1$mou_ang_5   |         11.13 |
|     6 | $exp_ped_1$fas_peg_1$fra_rot_1$mou_ang_4   |         11.13 |
|     7 | $exp_ped_1$fas_peg_1$fra_rot_1$mou_mul_1   |         11.13 |
|     8 | $exp_pip_1$fas_peg_1$fra_rot_1$mou_ang_2   |         11.13 |
|     9 | $exp_ped_1$fas_nut_1$fra_rot_1$mou_ang_2   |         11.13 |
|    10 | $exp_ped_1$fas_peg_2$fra_rot_1$mou_ang_2   |         11.13 |
|    11 | $exp_snp_1$fas_peg_1$fra_rot_1$mou_ang_2   |         11.13 |
|    12 | $exp_tel_1$fas_peg_1$fra_rot_1$mou_ang_2   |         11.13 |
|    13 | $exp_ped_1$fas_peg_1$fra_inv_1$mou_ang_2   |         11.41 |
|    14 | $exp_ped_1$fas_peg_1$fra_sli_3$mou_ang_2   |         11.41 |
|    15 | $exp_ped_1$fas_plu_1$fra_fix_1$mou_ang_2   |         11.41 |
|    16 | $exp_pip_1$fas_plu_1$fra_rot_1$mou_ang_2   |         11.41 |
|    17 | $exp_ped_1$fas_lev_1$fra_rot_1$mou_ang_4   |         11.41 |
                          ●
                          ●
                          ●
|   111 | $exp_tel_1$fas_peg_1$fra_sli_3$mou_ang_2   |         11.98 |
|   112 | $exp_snp_1$fas_peg_1$fra_sli_3$mou_ang_2   |         11.98 |
|   113 | $exp_tel_1$fas_peg_1$fra_inv_1$mou_ang_2   |         11.98 |
|   114 | $exp_pip_1$fas_peg_1$fra_sli_3$mou_ang_2   |         11.98 |
|   115 | $exp_ped_1$fas_nut_1$fra_sli_3$mou_ang_2   |         11.98 |
|   116 | $exp_ped_1$fas_peg_2$fra_inv_1$mou_ang_2   |         11.98 |
|   117 | $exp_ped_1$fas_nut_1$fra_inv_1$mou_ang_2   |         11.98 |
|   118 | $exp_pip_1$fas_peg_1$fra_inv_1$mou_ang_2   |         11.98 |
|   119 | $exp_ped_1$fas_peg_2$fra_sli_3$mou_ang_2   |         11.98 |
|   120 | $exp_ped_1$fas_peg_1$fra_fix_1$mou_cir_1   |         11.99 |
|   121 | $exp_ped_1$fas_peg_1$fra_rot_1$mou_ang_1   |         11.99 |
|   122 | $exp_slo_1$fas_peg_1$fra_rot_1$mou_ang_2   |         11.99 |
|   123 | $exp_ped_1$fas_nut_1$fra_rot_1$mou_cir_1   |         11.99 |
|   124 | $exp_ped_1$fas_peg_2$fra_rot_1$mou_cir_1   |         11.99 |
|   125 | $exp_pip_1$fas_peg_1$fra_rot_1$mou_cir_1   |         11.99 |
|   126 | $exp_tel_1$fas_peg_1$fra_rot_1$mou_cir_1   |         11.99 |
|   127 | $exp_snp_1$fas_peg_1$fra_rot_1$mou_cir_1   |         11.99 |
+-------+--------------------------------------------+---------------+
127 rows in set (0.00 sec)

mysql>
```

Figure 4-38: Filtered mySQL table showing combinations with aggregated DFA<12

It can be observed that by filtering the combinations, total number is reduced from 2,520 to 127.These combinations are ranked based on their total DFA index, and the design team selects two conceptual designs: #1 and #8 (Figure 4-39). In both the selected designs, it is assumed that 'Mou_ang_2' can be mounted on the ground as well as on the rear trunk of the stationary vehicle. The sign can be mounted on rear trunk of the vehicle through a pair of suction cups, which are fixed on the base of "Fra_rot_1".


Total Number of Designs Generated: 127

DFA Index: 10.56

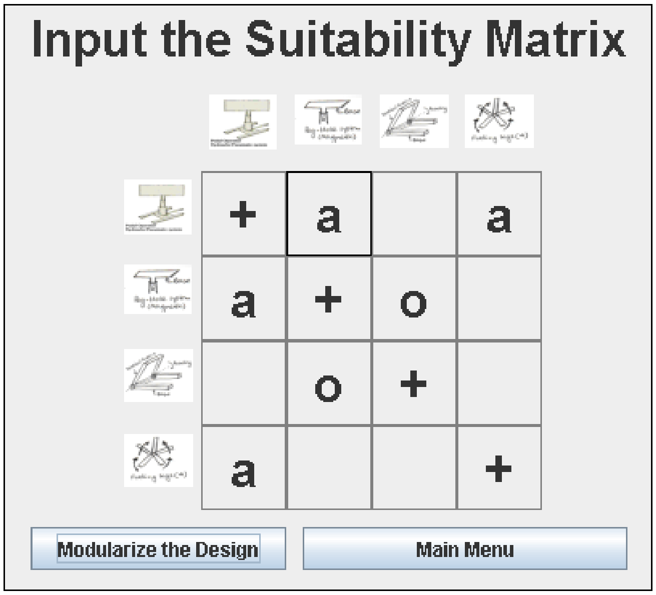
Total Number of Designs Generated: 127

DFA Index: 11.13

(a) **(b)**

Figure 4-39: Conceptual designs (a) #1 and (b) #8 selected after primary filtering

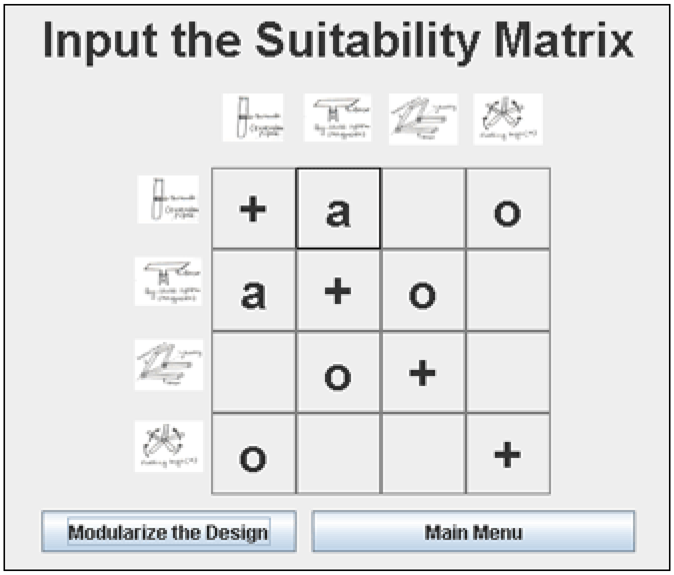
The design team then inputs the suitability matrix for both the selected concepts, which are highlighted in Figures 4-40 and 4-41. Modularization of both the selected designs is then carried out using the DA algorithm.



Input the Suitability Matrix

+	a		a
a	+	o	
	o	+	
a			+

Figure 4-40: GUI for inputting the suitability matrix for concept #1

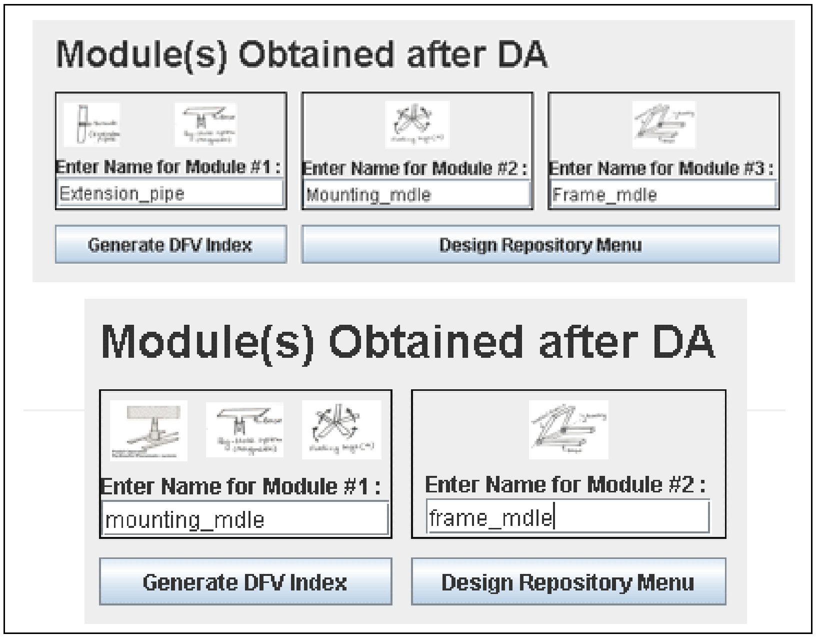


Input the Suitability Matrix

+	a		o
a	+	o	
	o	+	
o			+

Figure 4-41: GUI for inputting the suitability matrix for concept #8

Implementation of DA (Huang and Kusiak, 1998) resulted in the generation of two modules for concept #1 and three modules for concept #8. These are shown in Figure 4-42.



Module(s) Obtained after DA

Enter Name for Module #1 : Extension_pipe

Enter Name for Module #2 : Mounting_mdle

Enter Name for Module #3 : Frame_mdle

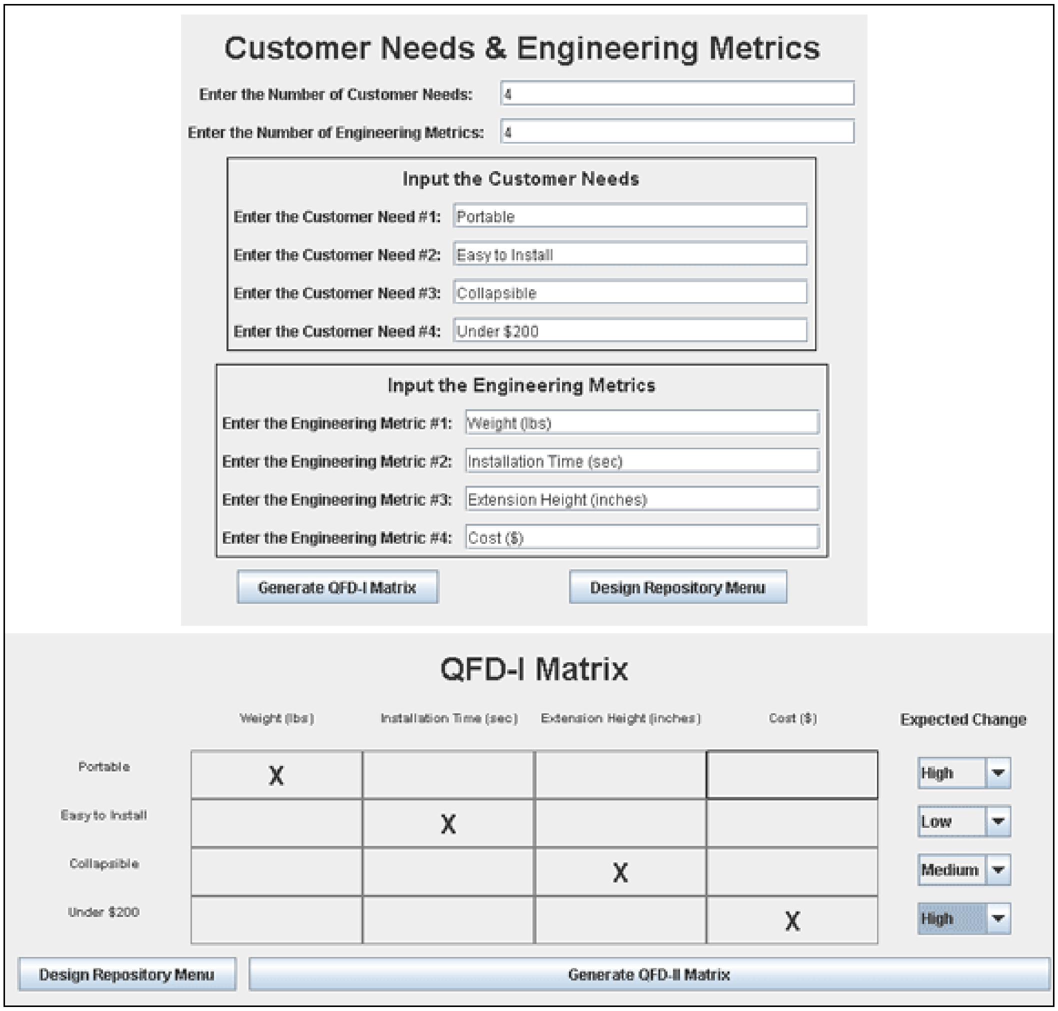
Module(s) Obtained after DA

Enter Name for Module #1 : mounting_mdle

Enter Name for Module #2 : frame_mdle

Figure 4-42: Modules obtained for the two selected designs (a) #8 and (b) #1

In order to obtain the best design, secondary filtering is done using the DFV index. Figure 4-43 shows the QFD phase-I matrix, while Figure 4-44 shows the QFD phase-II matrices for the two conceptual designs, namely conceptual design #1 and conceptual design #8. Next step involves the calculation of the DFV index for each conceptual design, and the GUI of the generated DFV matrix is shown in Figure 4-45. The value of the DFV index for conceptual design #1 is found to be 16, while for conceptual design #8, the value of the DFV index obtained is 23. Hence, conceptual design #1 is chosen as the best overall concept, since less redesign effort would be required to meet the future customer needs. Figure 4-46 shows the final design obtained from the proposed research framework.



Customer Needs & Engineering Metrics

Enter the Number of Customer Needs: 4

Enter the Number of Engineering Metrics: 4

Input the Customer Needs

Enter the Customer Need #1: Portable

Enter the Customer Need #2: Easy to Install

Enter the Customer Need #3: Collapsible

Enter the Customer Need #4: Under $200

Input the Engineering Metrics

Enter the Engineering Metric #1: Weight (lbs)

Enter the Engineering Metric #2: Installation Time (sec)

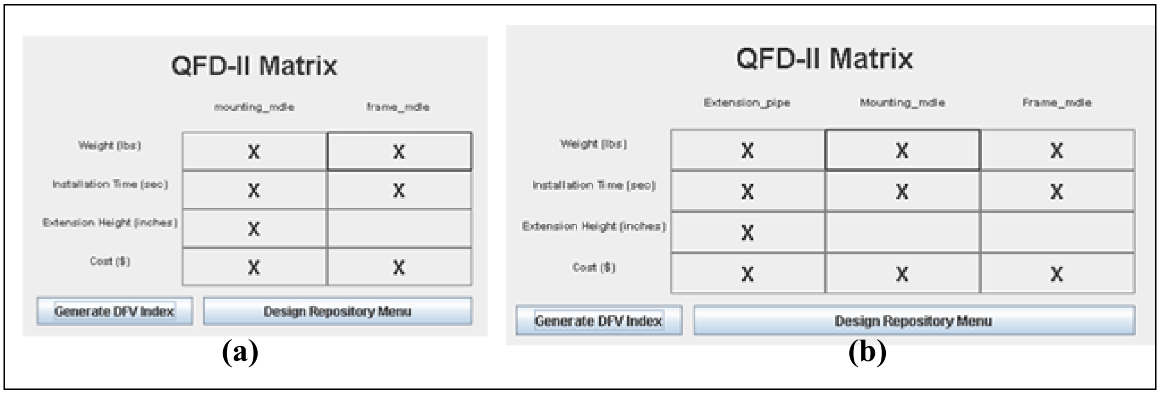
Enter the Engineering Metric #3: Extension Height (inches)

Enter the Engineering Metric #4: Cost ($)

QFD-I Matrix

	Weight (lbs)	Installation Time (sec)	Extension Height (inches)	Cost ($)	Expected Change
Portable	X				High
Easy to Install		X			Low
Collapsible			X		Medium
Under $200				X	High

Figure 4-43: GUI for inputting the QFD Phase I matrix for VMS mounting system



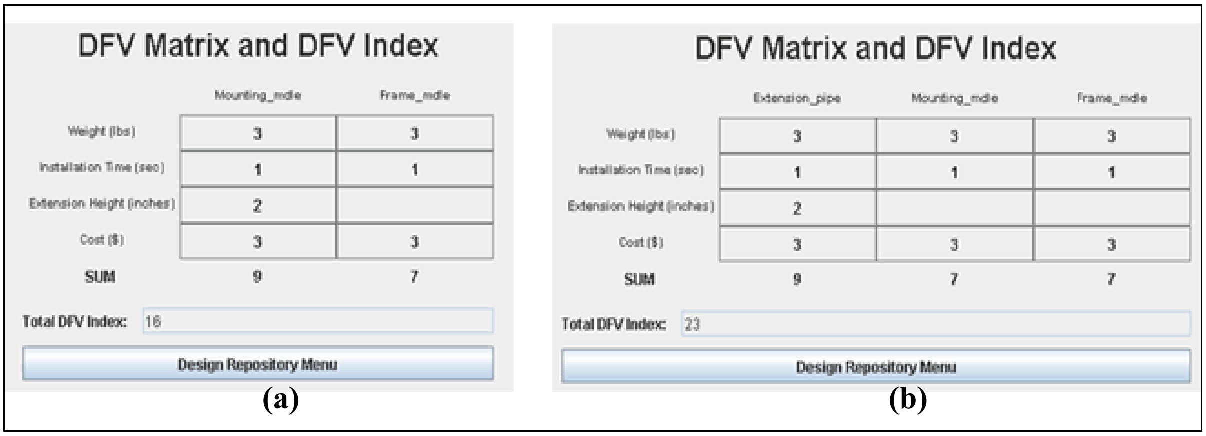
QFD-II Matrix

	mounting_mdle	frame_mdle
Weight (lbs)	X	X
Installation Time (sec)	X	X
Extension Height (inches)	X	
Cost ($)	X	X

QFD-II Matrix

	Extension_pipe	Mounting_mdle	Frame_mdle
Weight (lbs)	X	X	X
Installation Time (sec)	X	X	X
Extension Height (inches)	X		
Cost ($)	X	X	X

Figure 4-44: GUI for inputting QFD Phase II matrices for concepts (a) #1 and (b) #8



DFV Matrix and DFV Index

	Mounting_mdle	Frame_mdle
Weight (lbs)	3	3
Installation Time (sec)	1	1
Extension Height (inches)	2	
Cost ($)	3	3
SUM	9	7

Total DFV Index: 16

DFV Matrix and DFV Index

	Extension_pipe	Mounting_mdle	Frame_mdle
Weight (lbs)	3	3	3
Installation Time (sec)	1	1	1
Extension Height (inches)	2		
Cost ($)	3	3	3
SUM	9	7	7

Total DFV Index: 23

Figure 4-45: DFV matrices generated for the two selected concepts (a) #1 and (b) #8

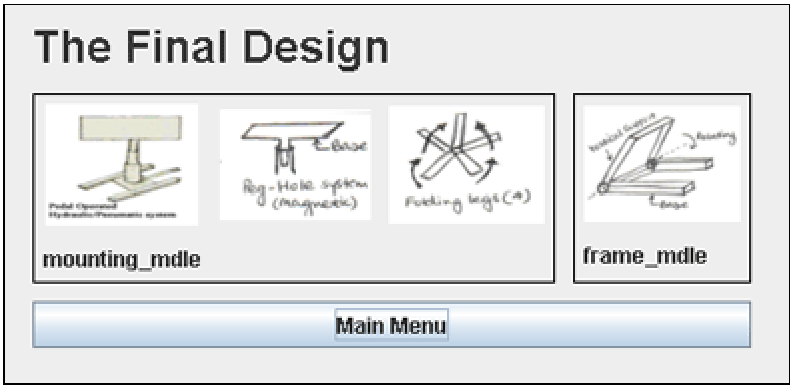


Figure 4-46: Final design of the VMS mounting system as shown by the software

Figure 4-46 shows the images of all the components that are to be used in the final conceptual design, thereby enabling the designer to easily visualize the conceptual design. Figure 4-47 shows the sketch of the final concept proposed by a designer after visualizing the conceptual design shown in Figure 4-46. On the right side of Figure 4-47, the two modules can be seen in their compact form, which allows them to be easily placed in a vehicle trunk. The designer proposed that the frame module as well as mounting module be fabricated using Aluminum, as this would make it portable and easy to be lifted by the end user. Additionally, the designer proposed to attach a pair of suction cups on the base of the frame module, thereby allowing the frame module to be directly mounted on any metallic surface (e.g., on the top or on the hood of a stationary vehicle). Therefore, it can be concluded that the proposed research framework not only helps the designers to be able to easily visualize the selected conceptual design, but also enables them to suggest further improvements to the selected conceptual design. The next chapter discusses the validation of the proposed research framework by using Data Envelopment Analysis (DEA).

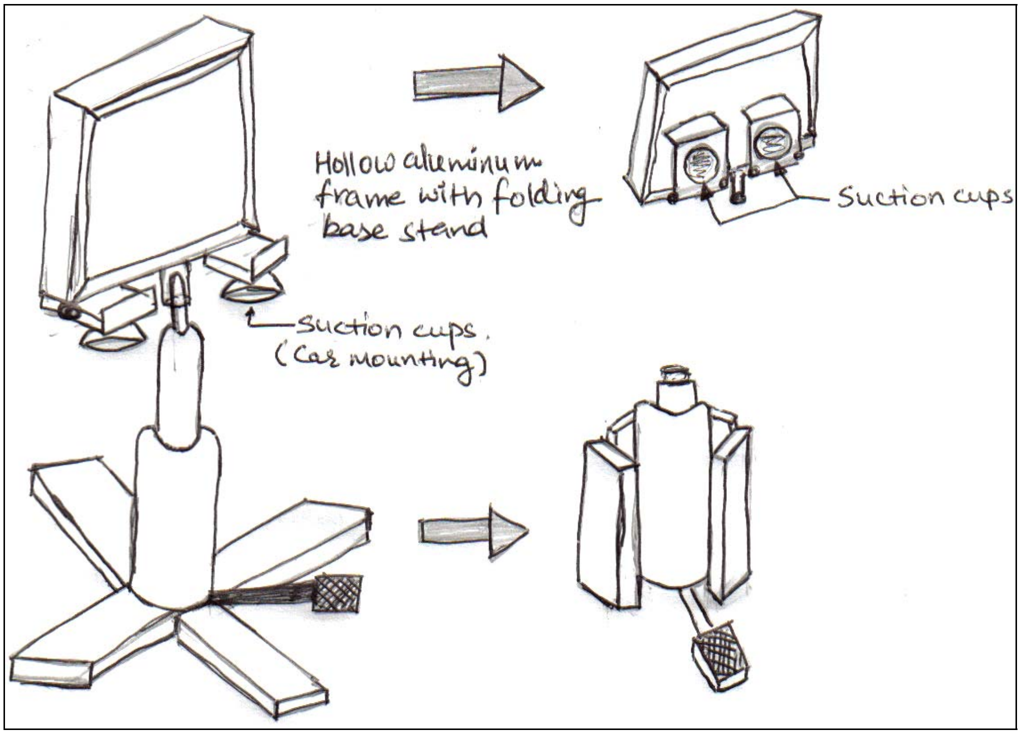


Figure 4-47: Final sketch of the best selected design concept

Chapter 5

Validating the Proposed Framework using Data Envelopment Analysis

In this chapter, the proposed research framework is validated using Data Envelopment Analysis (DEA). Section 5.1 provides an overview of DEA, while Section 5.2 discusses the DEA based concept selection tool developed by Lin et al. (2008). Section 5.3 implements the DEA based concept selection tool on the electronic toothbrush design example, and compares the result to that obtained from the proposed research framework.

5.1 Background on Data Envelopment Analysis

Data Envelopment Analysis (DEA) is a linear programming based technique, which was developed by Charnes et al. (1978). DEA is commonly used to measure the relative productive efficiency among a group of Decision Making Units (DMUs) by forming an efficient frontier. The efficiency value (θ) is measured by using the relative distance projection toward the frontier for a given a set of DMUs. The efficiency value (θ) is usually regarded as the efficiency or the productivity index, which ranges from 0 to 1. The most efficient DMUs are obtained on the frontier with $\theta = 1$, while the inefficient ones fall beyond the frontier with $\theta<1$.

There are a variety of models resulting from different ways in measuring the projection. For example, CCR (Charnes, Cooper, Rhodes) model (Charnes et al., 1978) and BCC (Banker, Charnes, Cooper) model (Banker et al., 1984) measure the projection to the frontier, while the additive model (ADD) measures the largest sum of the horizontal and vertical distances toward the frontier (Cooper et al., 2000). DEA has two major advantages when compared with other Multi-Criteria Decision Making (MCDM) methods. First of all, DEA easily solves multi-dimensional problems involving multiple input and multiple output indices. Secondly, DEA eliminates the problem of allocating unequal weights for the objective function as prevalent in most MCDM methods. DEA utilizes the weight for each input and output, which will let each DMU to reach its maximum possible efficiency value (Charnes et al., 1994). These two benefits of DEA allow it to be used in a large number of applications.

There are two possible orientations of DEA models: 1) input oriented model, and 2) output oriented model. For the input oriented model, the performance is improved by utilizing the inputs while output oriented model tries to improve the process by adjusting the outputs. Furthermore, DEA contains two types of Return to Scale (RTS) features: 1) Constant Return to Scale (CRS), and 2) Variable Return to Scale (VRS). These two RTS features mainly differ in the shape of the efficient frontier. In the CRS feature, it is assumed that the outputs always change in the same rate as the inputs, thus the frontier is shaped in a steady slope. On the other hand, for the VRS feature, the outputs might change in an increasing or decreasing manner as the inputs change. Hence, the frontier of VRS is a concave graph composed of several line segments, each with different slopes. The most commonly used DEA model is the CCR model (Charnes et al., 1978). The CCR model is based on the assumption of CRS feature. Each DMU in CCR model compares its performance to the most productive scale and receives an absolute efficiency value.

Sometimes a DEA result might comprise of multiple efficient DMUs, which all tie in the score of 1. During DEA analysis, one may also find the efficiency scores of all the DMUs to be quite close to each other, which may be due to the limited dispersion of each index. Accordingly, distinguishing DMUs in a detailed ranking is important for users in the decision making process. Cross efficiency ranking method is a widely used ranking method to rank DEA result that was first developed by Sexton et al. (1986). For each DMU, this ranking method calculates the efficiency scores as a product of n and the best weights of each DMU (Adler et al., 2002), and then forms a n × n matrix, called the cross-efficiency matrix. In the cross-efficiency method, each element's value ranges from 0 to 1, and the diagonal is the original DEA score. The cross-efficiency score of each DMU is obtained by averaging the sum of each row in the cross-efficiency matrix. Different from CCR scores, the most efficient DMU might score lower than 1 under the cross-efficiency method. If a DMU reaches a cross-efficiency score of 1, it indicates that this DMU dominates all the others in performance.

DEA has been applied extensively as a performance evaluation tool for a broad range of applications. In the engineering design area, Miyashita et al. (2002) constructed a supervisory system that used DEA to solve a collaborative design problem. The researchers adapted a CCR input oriented model with assurance region method to continuously search for a superior compromise solution between two conflicting design viewpoints. Paradi et al. (2002) used DEA to analyze the performance of engineering design teams at Bell Canada. The authors utilized an input-oriented CRS DEA model as well as a VRS DEA model in order to evaluate the performance of 39 engineering design teams and to discover the potential direction for improvement of inefficient teams. Farris et al. (2006) adopted DEA as a project evaluation tool to analyze projects from two different engineering design processes. According to the research by Farris et al. (2006), the DEA result indicated that adopting a new design process improved the overall performance. Section 5.2 discusses the DEA based concept selection tool developed by Lin et al. (2008).

5.2. DEA based Concept Selection Tool

In this section, DEA is utilized as a concept selection tool for the electronic toothbrush design example, which was presented in Section 4.1. The DEA-based concept selection tool comprises of five steps that are explained as follows:

1. **Data Collection:** This step involves collecting sufficient data for the concept-selection problem. It is imperative for the design decision-makers to have sufficient information about product specifications, product architecture (e.g., modules, components, etc.), possible

product concept alternatives etc., when dealing with the concept-selection phase. The sufficiency of the collected data determines the quality of the final decision.

2. **Acquiring Indices:** This step involves selecting the correct set of indices for the concept-selection process. In DEA analysis, using inappropriate indices may lead to meaningless results. Additionally, the number of different indices should be limited, and accordingly, only the main factors with potentially significant effects on the decision should be included in the set. Too many indices can cause the result to loose discriminatory power (Paradi et al., 2002). The recommended maximum number of input and output indices for DEA is equal to one-half the number of DMUs (Dyson et al., 2001).
3. **Model Selection:** This step involves selecting the most appropriate DEA model based on the property of the indices and the decision purpose. Model types that change based on the calculation of the projection (CCR, ADD), or problem/ variable characteristics (such as input-oriented or output oriented models) dictate the selection of the model. Steps 2 and 3 of the proposed framework should be treated very closely as the property of the indices would have strong influence on the model to be used.
4. **DEA Model Execution:** This step involves running the software package used to solve the DEA model selected in the previous step. A number of software packages have been developed to solve DEA problems, such as Frontier Analyst, DEA Frontier, etc. In addition, more generic softwares could also be programmed for the DEA application. For example, Excel VBA is utilized for the DEA-based concept selection tool.
5. **Result analysis:** This step involves ranking the results, since mostly a tie is present in multiple most efficient DMUs. Ranking of the results is completed with a specific DEA ranking method, such as cross-efficiency method, Andersen-Petersen method, assurance region method, etc. This ranking will break the ties. During the analysis of the results, special

attention should be directed to the meaning of the model parameters such as θ, η, μ etc., to obtain the most appropriate result.

The flow of the decision process is illustrated in Figure 5-1. Section 5.3 illustrates the implementation of the DEA based concept selection tool on the electric toothbrush example.

5.3 Implementation of DEA on the Electric Toothbrush Design Example

According to the first step of the DEA-based concept selection tool, the required data and information for the two electrical toothbrushes (i.e. Oral-B™ Vitality Series® Dual Clean, and Crest™ Spin Brush Pro) have been gathered by dissecting them and analyzing their components. These components are illustrated in Figures 4-2, 4-3 and 4-4 and are discussed in Section 4.1.

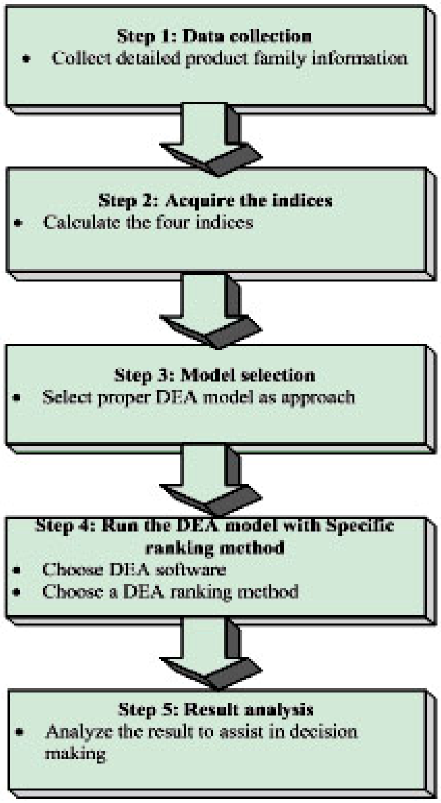

Step 1: Data collection
• Collect detailed product family information
Step 2: Acquire the indices
• Calculate the four indices
Step 3: Model selection
• Select proper DEA model as approach
Step 4: Run the DEA model with Specific ranking method
• Choose DEA software
• Choose a DEA ranking method
Step 5: Result analysis
• Analyze the result to assist in decision making

Figure 5-1: Process for product selection using DEA (Adopted from Lin et al., 2008)

In the second step of the DEA-based concept selection process, three indices are obtained. These three indices are explained as follows:

1. **DFA Index:** As mentioned previously in Section 2.6, DFA emerged as a tool to simplify the product structure, thereby reducing the total number of parts and the total cost of the parts. The calculation of DFA index used in the proposed research framework has been introduced in Section 3.2. As indicated before, DFA index is calculated based on the values obtained for 13 different criteria (Rampersad, 1994; Hsu et al., 1998). A lower value of DFA index represents a higher ease of assembly. For the electronic toothbrush design example, which was discussed in Section 4.1, it was observed that the value of the DFA index ranged from 17.5676 to 26.1621 for the two electronic toothbrush models. These DFA index values are highlighted in Table 5-1.
2. **Number of Modules:** It has been mentioned previously in Section 2.5.1 that by implementing modular product architecture, the manufacturing division can gain several benefits. As a result, a higher number of modules yields a better conceptual design for the manufacturing firm. The number of modules for the two electronic toothbrush models has been determined from the proposed research framework, which utilizes the Decomposition Approach (DA) by Huang and Kusiak (1998). The DA algorithm is briefly explained in Section 3.4 and the resultant modules are highlighted in Table 5-1.
3. **DFV Index:** As mentioned previously in Section 2.7, DFV emerged as a tool for the manufacturing firms to be able to optimize the costs of producing all the varieties of products together by incorporating the changes in the customer needs with time. The calculation of the DFV index for the proposed methodology has been described in Section 3.5, which has been adapted from the research by Martin and Ishii (2000). It is observed that the DFV index ranges from 19 to 31 as shown in Table 5-1, and is highly correlated to the number of modules. In other words, when the number of modules is 2, the DFV index is 19. However,

for 3 modules, the DFV index may be either 24 or 26, depending on whether or not the oscillation generator (i.e. crank-slider mechanism for Crest™ or the four-bar linkage for Oral-B™) is suitable to be included in the same module as the motor. The DFV index value for 4 modules is 31.

Table 5-1: Summary of indices for all conceptual designs of the electric toothbrush

No.	DFA	Modules	Compatibility	DFV
1	17.5676	4	1	31
2	17.5676	4	1	31
3	17.8378	4	0	31
4	17.8378	3	1	24
5	17.8378	3	1	24
6	17.8378	4	0	31
7	18.1081	3	0	24
8	18.1081	3	0	24
9	18.6486	3	1	26
10	18.6486	3	1	26
11	18.6486	4	0	31
12	18.6486	4	0	31
13	18.9189	2	1	19
14	18.9189	2	1	19
15	18.9189	3	0	26
16	18.9189	3	0	26
17	18.9189	4	1	31
18	18.9189	4	1	31
19	18.9189	3	0	24
20	18.9189	3	0	24
21	19.1892	3	1	24
22	19.1892	2	0	19
23	19.1892	3	1	24
24	19.1892	2	0	19
25	19.4594	4	1	31
26	19.4594	4	1	31
27	19.7297	3	1	24
28	19.7297	3	1	24
29	19.7297	4	0	31
30	19.7297	4	0	31
31	19.7297	3	0	26
32	19.7297	3	0	26
33	20	2	0	19
34	20	3	0	24
35	20	3	0	24
36	20	3	1	26
37	20	2	0	19
38	20	3	1	26
39	20.2703	2	1	19
40	20.2703	2	1	19
41	20.5405	4	0	31
42	20.5405	3	1	26
43	20.5405	3	1	26
44	20.5405	4	0	31
45	20.8108	2	1	19
46	20.8108	4	1	31
47	20.8108	3	0	24
48	20.8108	2	1	19
49	20.8108	4	1	31
50	20.8108	3	0	24
51	20.8108	3	0	26
52	20.8108	3	0	26
53	21.0811	3	1	24
54	21.0811	2	0	19
55	21.0811	3	1	24
56	21.0811	2	0	19
57	21.6216	3	0	26
58	21.6216	3	0	26
59	21.8919	3	1	26
60	21.8919	3	1	26
61	21.8919	2	0	19
62	21.8919	2	0	19
63	22.1621	2	1	19
64	22.1621	2	1	19

Table 5-1 summarizes the three different indices: 1) DFA index, 2) Number of modules, and 3) DFV index, for all the 64 concept alternatives generated by the proposed research framework. Additionally, the table also highlights the compatibility of the different components in each conceptual design alternative. The main reason for indicating compatibility is that those concepts possessing conflicting components in their design cannot be realized. For example, if the concept alternative comprises of a Crest™ coupler/de-coupler and an Oral-B™ brush head, the design is incompatible as these two components cannot mate with each other. All incompatible designs are denoted by 0 in the Compatibility column of Table 5-1. After eliminating the unworkable product concepts, the remaining 32 possible concepts are carried over to the third step in the DEA-based concept selection tool. The filtered conceptual design alternatives are shown in Table 5-2.

Table 5-2: Summary of the indices after filtering incompatible conceptual designs

No.	DFA	DFV	Modules
1	17.5676	31	4
2	17.5676	31	4
4	17.8378	24	3
5	17.8378	24	3
9	18.6486	26	3
10	18.6486	26	3
13	18.9189	19	2
14	18.9189	19	2
17	18.9189	31	4
18	18.9189	31	4
21	19.1892	24	3
23	19.1892	24	3
25	19.4594	31	4
26	19.4594	31	4
27	19.7297	24	3
28	19.7297	24	3
36	20	26	3
38	20	26	3
39	20.2703	19	2
40	20.2703	19	2
42	20.5405	26	3
43	20.5405	26	3
45	20.8108	19	2
46	20.8108	31	4
48	20.8108	19	2
49	20.8108	31	4
53	21.0811	24	3
55	21.0811	24	3
59	21.8919	26	3
60	21.8919	26	3
63	22.1621	19	2
64	22.1621	19	2

In the third step, the most appropriate DEA model for this electronic toothbrush design application needs to be determined. For manufacturers, reducing production cost can immediately ensure that they can keep pricing competitively and maintain profitability. DFA and DFV indices are both setup as input and therefore, minimizing these two inputs while keeping existing level of outputs is our target. Hence, in this case, input oriented DEA model is the proper method. Number of modules was chosen as the output index for the DEA model. However, it was found that the number of modules is very highly correlated with the DFV index. Furthermore, the range of the number of modules was very large, since the maximum module (i.e., 4) is twice of the minimum module (i.e., 2). Therefore, in order to obtain a stable DEA model, it was decided to take the square root of the number of modules as the output index. The overall objective of the DEA model is to choose the best conceptual design for the electronic toothbrush, and therefore, adopting the viewpoint of the most productive scale is imperative. For the above reason, the CRS DEA model meets our requirement and it is decided to use the CCR-input oriented model. The 32 compatible DMUs that were filtered from the previous step will be input into the CCR-input oriented model shown in Equation 1.

$$
\begin{aligned}
Max \quad & \theta_k = \frac{\sum_{r=1}^{s} u_r y_{rk}}{\sum_{i=1}^{m} v_i x_{ik}} \\
s.t. \quad & \frac{\sum_{r=1}^{s} u_r y_{rj}}{\sum_{i=1}^{m} v_i x_{ij}} \le 1, \quad j = 1, \cdots, n, \\
& v_i \ge \varepsilon > 0, \quad i = 1, \cdots, m, \\
& \mu_r \ge \varepsilon > 0, \quad r = 1, \cdots, s.
\end{aligned}
\qquad \text{.........} (1)
$$

From Equation 1, θ_k is the efficiency value of the k^{th} DMU, x_{ik} and y_{rk} represent the input and output indices of the k^{th} DMU. v_i and u_r are the weights, which are generated automatically during the computation process. The dual linear programming model is shown in Equation 2.

$$\begin{aligned} &Min\ \theta_k \\ &s.t \quad \theta x_k - \sum_{j=1}^{n} x_{ij}\lambda_j \geq 0 \quad (i = 1,2,\cdots m) \\ &\sum_{j=1}^{n} y_{rj}\lambda_j \geq y_{rk} \quad (r = 1,2,\cdots,s) \\ &\lambda_j \geq 0 \quad (j = 1,2,\cdots,n) \end{aligned} \qquad \text{.........} (2)$$

In Equation 2, θ_k ranges from 0 to 1 and DMUs with $\theta_k = 1$ are most efficient. These most efficient DMUs stand on the efficient frontier and are used as the reference set for all the other inefficient DMUs. DEA scores of the 32 conceptual design alternatives are shown in Table 5-3.

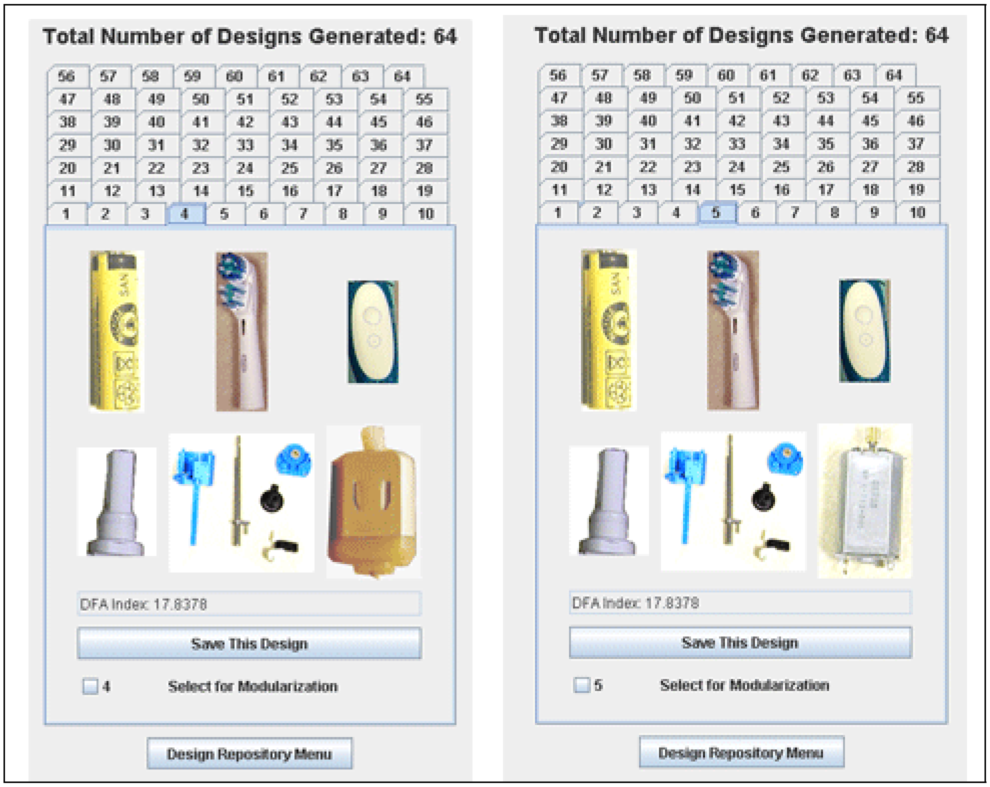
Table 5-3: Results generated from the DEA analysis

DMU	CCR Score (θ)	Ranking	DMU	CCR Score (θ)	Ranking
4	0.985104766	1	17	0.921840611	17
5	0.985104766	2	18	0.921840611	18
21	0.968088609	3	9	0.916992264	19
23	0.968088609	4	10	0.916992264	20
27	0.961623509	5	25	0.915938259	21
28	0.961623509	6	26	0.915938259	22
53	0.946220008	7	63	0.908201008	23
55	0.946220008	8	64	0.908201008	24
13	0.94589723	9	36	0.901931811	25
14	0.94589723	10	38	0.901931811	26
1	0.937398183	11	46	0.90189112	27
2	0.937398183	12	49	0.90189112	28
39	0.929435167	13	42	0.896197741	29
40	0.929435167	14	43	0.896197741	30
45	0.923165813	15	59	0.882510211	31
48	0.923165813	16	60	0.882510211	32

In Step 5, the result analysis is recommended. As seen in Table 5-3, DMU's 4 and 5 emerge to be the most efficient product concept in the CCR score. Analysis of these two DMU's reveals that they are the conceptual designs #4 and #5 from the output obtained from the proposed research software discussed in Section 4.1. The result obtained from the proposed framework was conceptual design #4, which is exactly the same as DMU 4. Furthermore, it can be observed from Table 5-4 that DMU 5 differs from DMU 4 only by the replacement of the Crest motor with the Oral-B motor. Figure 5-2 shows the two conceptual designs (i.e., conceptual designs #4 and #5) highlighting the similarity between the two designs.

Table 5-4: Components present in the two DMUs chosen by the DEA tool

DMU	Components Present in the Model	DFA	Modules	DFV
4	Rechargeable Battery, Oral-B Brush Head, Oral-B Push Button, Oral-B Coupler/De-coupler, Four-bar-link, Crest motor	17.8378	3	24
5	Rechargeable Battery, Oral-B Brush Head, Oral-B Push Button, Oral-B Coupler/De-coupler, Four-bar-link, Oral-B motor	17.8378	3	24



Total Number of Designs Generated: 64

DFA Index: 17.8378

□ 4

Total Number of Designs Generated: 64

DFA Index: 17.8378

□ 5

Figure 5-2: Two conceptual designs selected by the DEA conceptual design tool

It can be summarized that the DEA based concept selection tool provided similar results as obtained from the proposed research framework. As a result, the proposed research framework is found to be highly robust and positively beneficial to manufacturing firms as it would enable them to obtain new and innovative product designs that 1) are modular, 2) offer ease of assembly, and 3) can be easily updated to accommodate changes in future customer needs. Moreover, the proposed conceptual framework involves the designer to interact with the decision making process (i.e. filtering out incompatible designs), which further enhances the conceptual designs based on the designer's knowledge and expertise. The next chapter provides conclusions of the proposed research framework, and offers recommendations for future research.

Chapter 6

Conclusion

A review of the existing literature on the computational tools for electro-mechanical product design generation reveals that these tools either primarily focus only on the initial design phase, or only in the later steps of design. As a result, the existence of few computational tools at the conceptual design stage have caused the product designers to be limited to few options like drawing based on experience or utilizing patent searches and reverse-engineering techniques to generate successful product designs. Keeping this objective in mind, a new conceptual design generation tool based on the amalgamation of Modularity, DFA and DFV, has been the matter of discussion in this thesis. It is estimated that the effect of integrating Modularity, DFA and DFV along with the computer aided conceptual design tool would be tremendous, as it would benefit manufacturing firms by offering automated conceptual product designs.

The two electro-mechanical product design examples that were discussed in this thesis enunciate the application of the conceptual design software framework to real life industrial problems. The primary step of obtaining and assessing the customer needs is imperative since it enables the design team to focus the conceptual design more on the specific needs and tastes of the consumers. The development of the EMS functional model helps the design team to decompose the overall product function represented in the black box model, into simpler sub-functions and flows that are easily input in the software framework. The rule-based searching technique is then executed, which seeks to map all possible combinations of components present in the design repository, in order to satisfy the overall product function. The primary filtering of these conceptual design variants is based on the total DFA index value of the components present in that design variant. Modularization of two user-selected designs is performed, followed by a secondary filtering using a DFV index. The overall sequential approach of the software framework, allows the design team to be an integral part of the product design process, allowing them to utilize their knowledge and expertise to interactively filter out non-feasible conceptual design variants throughout the intermediary steps of the product design process. This software framework has been formally validated by utilizing a DEA based concept selection tool. The strength of the validation tool has been its ability to validate the realistic electro-mechanical product design problem definition as well as the conceptual design generation and selection capabilities of the software framework.

Overall, we believe that this conceptual design generation software will be an asset to the product designers, and should be an important part of their computational software toolkit. Apart from being utilized by manufacturing companies to generate new product designs, this software tool also finds an application in engineering education. As the proposed software framework is able to provide a virtual dissection environment, it can be integrated with the Cyber-Infrastructure-Based Engineering Repositories for Undergraduates (CIBER-U) framework proposed by Devendorf et al. (2007) to enable a better undergraduate learning experience.

Future improvements to the proposed software framework include the incorporation of a multi-criteria decision making capability as the current framework mainly ranks the conceptual designs only based on assembly time. Future research will involve incorporating additional filtering criteria in order to improve the efficiency of the concept selection process as well as evaluating the effectiveness of the proposed framework by comparing the conceptual designs generated from the software framework to the paper-pencil designs generated by designers at the conceptual stage. In the future, a major addition to the current framework would be the replacement of component images in the design repository with actual 3D CAD models. As a result, once the designer selects a best conceptual design, it is envisioned that individual CAD files of each component associated with that design should automatically open up within the 3D CAD-assembly environment. We believe that this feature would enable the designer to effectively utilize the 3-D features of CAD (i.e., scaling, rotating, assembling, etc.) at the conceptual design stage itself. Future plans also include making this software available as an online concept generator tool so that designers from various parts of the world can log-on to the system and populate the design repository.

Bibliography

N. Adler, L. Friedman and Z. Sinuany-Stern (2002), Review of Ranking Methods in the Data Envelopment Analysis Context, European Journal of Operation Research, vol. 140, pages 249 - 265.

Y. Akao (1990), Quality Functional Deployment, Productivity Press, Cambridge MA.

K.R. Allen and S. Carlson – Skalak (1998), Defining Product Architecture during Conceptual Design, Proceedings of the DETC Conference, Paper No. DETC98 / DTM – 5650

K.W.L. Antonio, R.C.M. Yam and E. Tang (2004), The impacts of product modularity on competitive capabilities and performance: An empirical study, International Journal of Production Economics, vol. 105, issue 1, pages 1- 20.

K. Aoki (1980), High speed and Flexible automated assembly line—why has automation successfully advanced in Japan?, Proceedings of the 4th International Conference on Production Engineering, Japan Society of Precision Engineering, Tokyo, pages 1–6.

C.Y. Baldwin and K.B. Clark, (1997), Managing in an age of modularity, Harvard Business Review, vol. 75, pages 84–93.

R.D. Banker, A. Charnes and W.W. Cooper (1984), Some Models for Estimating Technical and Scale Inefficiencies in Data Envelopment Analysis, Management Science, vol. 30, issue 9, pages 1078 – 1092.

Becker Associates Inc. (2000), http://www.becker-associates.com/thehouse.htm and http://www.becker-associates.com/qfdwhatis.htm.

O. Benami and Y. Jin (2000), An E-Documenting Approach to Conceptual Design, Proceedings of the ASME Design Engineering Technical Conference, vol. 1, pages 193-198.

M.R. Bohm, R.B. Stone, T.W. Simpson, E.D. Steva (2006), Introduction to Data Schema: The Inner Workings of a Design Repository, Proceedings of the ASME IDETC Conference, Paper No. DETC2006/CIE-99518.

M.R. Bohm, J.P. Vucovich and R.B. Stone (2005), Capturing Creativity using a Design Repository to Drive Concept Innovation, Proceedings of the ASME IDETC and CIE Conference, Paper No. DETC2005 – 85105

M. Bohm and R. Stone (2004), Representing Functionality to Support Reuse: Conceptual and Supporting Functions, Proceedings of the ASME DETC Conference, Paper No. DETC2004 – 57693

M. Bohm and R. Stone (2003), Refining Design Repositories, Creating Usable Framework with XML Data Representation, Proceedings of the 2003 NSF Grantees Conference, Birmingham, AL.

G. Boothroyd (1994), Product Design for Manufacture and Assembly, Computer Aided Design, vol. 26, issue 7, pages 505 – 520.

G. Boothroyd and P. Dewhurst (1989), Product Design for Assembly, Boothroyd Dewhurst Inc., Wakefield, RI.

J.C. Borda (1782), "Mémoir sur les Élections Au Scrutin" Histoire de Académie Royale des Sciences, Paris.

R. Bremner (1999), Cutting edge platforms, Financial Times Automotive World, pages 30–38.

C.R. Bryant, R.B. Stone, D.A. McAdams, T. Kurtoglu and M.I. Campbell (2005a), Concept Generation from the Functional Basis of Design, International Conference on Engineering Design, Melbourne, August 15-18.

C.R. Bryant, R.B. Stone, D.A. McAdams, T. Kurtoglu and M.I. Campbell (2005b), A Computational Technique for Concept Generation, Proceedings of the ASME IDETC and CIE Conference, Paper No. DETC2005-85323

J. Cagan (2001), Engineering Shape Grammars, Formal Engineering Design Synthesis, E.K. Antonsson and J. Cagan eds. Cambridge University Press.

B. Chandrasekaran (1994), Functional Representation: A Brief Historical Perspective, Applied Artificial Intelligence, vol. 8, pages 163 – 197.

A. Charnes, W.W. Cooper, and E. Rhodes (1978), Measuring the Efficiency of Decision Making Units, European Journal of Operation Research, vol. 2, issue 6, pages 429 - 444.

A. Charnes, W.W. Cooper, A.Y. Lewin and L.M. Seiford (1994), Data Envelopment Analysis: A Comprehensive Text with Models, Applications, References and DEA-Solver Software, Kluwer Academic, Boston, United States of America

W.W. Cooper, L.M. Seiford and K. Tone (2000), Data Envelopment Analysis: Theory, Methodology and Application. Kluwer Academic, Boston, United States of America

S.L. Coulter, B. Brass, M.W. McIntosh and D.W. Rosen (1998), Identification of Limiting Factors for Improving Design Modularity, Proceedings of the ASME DETC Conference, Paper No. DETC98 / DFM – 5659

Y-M Deng, S.B. Tor and G.A. Britton (2000), Abstracting and Exploring Functional Design Information for Conceptual Mechanical Product Design, Engineering with Computers, vol.16, pages 36 – 52.

Y-M Deng, S.B. Tor and G.A. Britton (1999), A Computerized Design Environment for Functional Modeling of Mechanical Products, Proceedings of the fifth ACM Symposium on Solid Modeling and Applications, pages 1-12.

M. Devendorf, K. Lewis, T.W. Simpson, R.B. Stone and W.C. Regli (2007), Evaluating the use of Cyberinfrastructure in the Classroom to Enhance Product Dissection, Proceedings of the ASME International Design Engineering Technical Conferences, paper no. DETC2007-35549.

S. Dowlatshahi (1992), Product Design in a Concurrent Engineering Environment, an Optimization Approach, Journal of Production Research, vol 30, issue 8, pages 1803 – 1818.

R. Dumitrescu and T. Szecsi (2002), Implementing Design for Manufacturing Rules, Technical Paper - Society of Manufacturing Engineers, issue 2, pages 1-11.

R.G. Dyson, R. Allen, A. Camanho, V.V. Podinovski, C.S. Sarrico and E.A. Shale (2001), Pitfalls and Protocols in DEA, European Journal of Operation Research, vol. 132, issue 2, pages 245 - 259.

O. Eggen (2003), Product Design 9, Modular Product Architectures, Nov. 2003.

A. Ericsson and G. Erixon (1999), Controlling design variants: Modular product platforms, ASME Press. New York, NY.

F. Fabricius (1994), A Seven Step Procedure for Design for Manufacture, World Class Design to Manufacture, vol. 1, issue 2, pages 23 – 30.

S.R. Farrel and T.W. Simpson (2001), Improving Commonality in Custom Products Using Product Platform, Proceedings of ASME DETC Conference, Paper No. DETC2001 / DAC – 21125

J.A. Farris, R.L. Groesbeck, E.M. Van Aken and G. Letens (2006), Evaluating the Relative Performance of Engineering Design Project: A Case Study Using Data Envelopment Analysis, IEEE Transactions on Engineering Management, vol. 53, issue 3, pages 471 - 482.

E. Feitzinger, H.L. Lee (1997), Mass customization at Hewlett-Packard: The power of postponement, Harvard Business Review, January–February, pages 116–121.

K. Fujita, H. Sakaguchi and S. Akagi (1999), Product variety deployment and its optimization under modular architecture and module communalization, Proceedings of ASME DETC Conference, vol. 4, pages 337 - 348.

J.K. Gershenson, G.J. Prasad and Y. Zhang (2003), Product Modularity: Definitions and Benefits, Journal of Engineering Design, vol. 14, issue 3, pages 295 – 313.

J.P. Gonzales–Zugasti and K.N. Otto (2000), Modular Platform-Based Product Family Design, ASME Automation Conference, vol. 2, pages 677-687.

F. Guo and J.K. Gershenson (2004), A Comparison of Modular Product Design Methods based on Improvement and Iteration, Proceedings of ASME DETC Conferences, vol. 3, pages 261 – 269.

S. Gupta and G.E. Okudan (2008a), Analyzing Three Modularizing Methodologies from Assembly and Variety Viewpoints, Proceedings of the Industrial Engineering Research Conference, Vancouver, BC, pages 1660 – 1665.

S. Gupta and G.E. Okudan (2008b), Computer-Aided Generation of Modularized Conceptual Designs with Assembly and Variety Considerations, Journal of Engineering Design (Accepted).

S. Gupta and G.E. Okudan (2008c), Generation of Modularized Conceptual Designs with Assembly and Variety Considerations, Proceedings of the ASME International Design Engineering Technical Conferences, New York, Paper No. DETC2008 – 50049

S. Gupta and G.E. Okudan (2008d), Assembly and Variety Considerations During Conceptual Design, Proceedings of the ASME International Design Engineering Technical Conferences, New York, Paper No. DETC2008 – 50050

S. Gupta, J. Baird and G.E. Okudan (2008), A Tool for Computer-Aided Generation of Modularized Conceptual Designs, Proceedings of the Industrial Engineering Research Conference, Vancouver, BC, pages 1666 – 1671.

S. Gupta and G.E. Okudan (2007), Modular Design: A Review of Research and Industrial Applications, Proceedings of the Industrial Engineering Research Conference, Nashville, TN, pages 1563 – 1568.

J.R. Hauser and D. Clausing (1988), The House of Quality, The Harvard Business Review, May-June, issue 3, pages 63-73.

J.W. Herrmann, J. Cooper, S.K. Gupta, C.C. Hayes, K. Ishii, D. Kazmer, P.A. Sandborn and W.H. Wood (2004), New Directions in Design for Manufacturing, Proceedings of the DETC and CIE Conference, Paper No. DETC2004 – 57770

J. Hirtz, R. Stone, D. McAdams, S. Szykman and K. Wood (2002), A Functional basis for Engineering Design: Reconciling and Evolving Previous Efforts, Research in Engineering Design, vol. 13, issue 2, pages 65 – 82.

V. Hoek (1998), Letters to the Editor, Harvard Business Review.

K. Hölttä, E. Suh, and O. de Weck (2005), Trade-off between modularity and performance for engineered systems and products, In Proc of International Conference on Engineering Design, pages 1-16.

K. Hölttä and M. Salonen (2003), Comparing three modularity methods, Proceedings of ASME Design Engineering Technical Conferences, Paper No. DETC2003 / DTM-48649

S-W Hsiao and E. Liu (2005), A structural component-based approach for designing product family, Computers in Industry, vol. 56, pages 13–28.

W. Hsu, J.Y.H. Fuh and Y. Zhang (1998), Synthesis of Design Concepts from a Design for Assembly Perspective, Computer Integrated Manufacturing Systems, vol. II, issue 1-2, pages 1-13.

G.Q. Huang, X.Y. Zhang and L. Liang (2005), Towards Integrated Optimal Configuration of Platform Products, Manufacturing Processes and Supply Chains, Journal of Operations Management, vol. 23, Issues 3-4, pages 267 – 290.

C-C Huang and A. Kusiak (1998), Modularity in Design of Product and Systems, IEEE Transactions on Systems, Man and Cybernetics, vol. 28, issue 1, pages 66 – 78.

Y. Iwasaki, A. Farquhar, R. Fikes and J. Rice (1997), A Web-Based Compositional Modeling System for Sharing of Physical Knowledge, Proceedings of the 15th International Conference on Artificial Intelligence, AAAI Press/MIT Press, August.

Y. Iwasaki, M. Vescovi, R. Fikes and B. Chandrasekaran (1995), Causal Functional Representation Language with Behavior – based Semantics, Applied Artificial Intelligence, vol. 9, issue 1, pages 5 – 31.

J. Jiao and M.M. Tseng (1999), A methodology for developing Product Family Architecture for mass customization, Journal of Intelligent Manufacturing, vol. 10, pages 3 - 20.

A. Jose and M. Tollenaere (2005), Modular and Platform Methods for product family design: literature analysis, journal of intelligent manufacturing, vol. 16, pages 371 – 390.

Y. Kitamura and R. Mizoguchi (1999), Metafunctions of Artifacts, Proceedings of the Thirteenth International Workshop on Qualitative Reasoning, pages 136-145, Loch Awe, Scotland.

Y. Kitamura and R. Mizoguchi (1998), Functional Ontology for Functional Understanding, Twelfth International Workshop on Qualitative Reasoning, AAAI Press, pages 77 – 87, Cape Cod, Massachusetts.

V.B. Kreng and T-P Lee (2004), Modular Product Design with grouping genetic algorithm – a case study, Computers and Industrial Engineering, vol. 46, pages 443 – 460.

H. Krikke, I.I. Blanc and S. van de Velde (2004), Product modularity and the design of closed-loop supply chains, California Management Review , vol. 46, issue 2, pages 23–39.

S. Kumara, I. Ham, M. Al-Hamando and K. Goodnow (1989), Causal Reasoning and Data Abstraction in Component Design, Annals of the CIRP, vol. 38, pages 145 – 148.

S. Kumara and I. Ham (1990), Use of Associative Memory and Self-Organization in Conceptual Design, Annals of CIRP, vol. 39, pages 117-120.

T. Kurtoglu, M.I. Campbell, J. Gonzalez, C.R. Bryant, R.B. Stone and D.A. McAdams (2005), Capturing Empirically Derived Design Knowledge for Creating Conceptual Design

Configurations, Proceedings of the ASME IDETC and CIE Conference, Paper No. DETC2005 – 84405

T. Kurtoglu, M.I. Campbell, C.R. Bryant, R.B. Stone, D.A. McAdams (2005), Deriving a Component Basis for Computational Functional Synthesis, International Conference on Engineering Design, Melbourne, August 15-18.

A. Kusiak (2002), Integrated Product and Process Design: A modularity perspective, Journal of Engineering Design, vol. 13, issue 3, pages 223 – 231.

A. Kusiak, T. N. Larson and J. Wang (1994), Reengineering of Design and Manufacturing Process, Computers and Industrial Engineering, vol. 26, issue 3, pages 521 – 536.

C-Y Lin, S. Gupta and G.E. Okudan (2008), An Improved Concept Selection approach for Design Decision Making, Proceedings of the Industrial Engineering Research Conference, Vancouver, BC, pages 1706 – 1711

Z-H Lin and A. Che (1998), Intelligent Quality Function Deployment System in Concurrent Engineering Environment, Proceedings of SPIE vol. 3517, pages 191 – 198.

A. Little, K. Wood and D. McAdams (1997), Functional Analysis: A fundamental empirical study for Reverse Engineering, Benchmarking and Redesign', Proceedings of the 1997 DETC Conference, Paper No. 97-DETC/DTM 3879

Y. Liu, A. Chakrabarti and T. Bligh (1999), Transforming Functional Solutions to Physical Solutions, Proceedings of the ASME Design Theory and Methodology Conference, Paper No. DETC99/DTM-8768

M.V. Martin and K. Ishii (2002), Design for Variety: Developing Standardized and Modularized Product Platform Architectures, Research in Engineering Design, vol. 13, pages 213 – 235.

M.V. Martin and K. Ishii (2000), Design for Variety: A Methodology for Developing Product Platform Architectures, Proceedings of the ASME DETC Conference, Paper No. DETC2000 / DFM – 14021

M.V. Martin and K. Ishii (1997), Design for Variety: Development of Complexity Indices and Design Charts, Proceedings of the ASME DETC Conference, Paper No. DETC97 / DFM – 4359

M.V. Martin and K. Ishii (1996), Design for Variety: A Methodology for Understanding the Costs of Product Proliferation, Proceedings of the ASME DETC and CIE Conference, Paper No. 96-DETC / DTM-1610.

C.A. Mattson and S.P. Magleby (2001), The influence of product modularity during concept selection of consumer products, Proceedings of the ASME DETC Conference, Paper No. DETC2001 / DTM-21712

J.H. Mikkola and O. Gassmann (2003), Managing Modularity of Product Architectures: Towards an Integrated Theory, IEEE Transactions on Engineering Management, vol. 50, issue 2, pages 205 – 218.

T. Miyashita and H. Yamakawa (2002), A Study on the Collaborative Design Using Supervisor System, JSME International Journal, vol. 45, issue 1, pages 333 – 341.

O.S. Muogboh (2003), Supporting Functionality Based Design in Computer Aided Design Systems, Phd. Dissertation, University of Pittsburgh.

J.W. Murdock, S. Szykman and R.D. Sriram (1997), An Information Modeling Framework to Support Database and Repositories, Proceedings of the ASME DETC Conference, Paper No. DETC97/DFM-4373

B. Nepal, L. Monplaisir and N. Singh (2005), Integrated fuzzy logic-based model for product modularization during concept development phase, International Journal of Production Economics, vol. 96, pp. 157 – 174.

G. Pahl, and W. Beitz (1998), Engineering Design A systematic approach, Springer-Verlag.

G. Pahl and W. Beitz (1984), Developing size ranges and modular products. In K. Wallace (ed.) Engineering Design (London: Design Council), pages 315 – 361.

J.C. Paradi, S. Smith and C. Schaffnit-Chatterjee (2002), Knowledge Worker Performance Analysis Using DEA: An Application to Engineering Design Team at Bell Canada, IEEE Transactions on Engineering Management, vol. 49, issue 1, pages 161 - 172.

T.U. Pimmler and S.D. Eppinger (1994), Integration Analysis of Product Decompositions, ASME Design Theory and Methodology conference, pp. 343 – 351.

S.F. Qin, R. Harrison, A.A. West, I.N. Jordnov and D.K. Wright (2003), A Framework of Web-based Conceptual Design, Computers in Industry, vol. 50, pages 153 – 164.

R. Rai and V. Allada (2003), Modular product family design: agent-based Pareto-optimization and quality loss function-based post-optimal analysis, International Journal of Production, vol. 41, issue 17, pages 4075 – 4098.

H.K. Rampersad (1995), Integrated and Simultaneous Design for Robotic Assembly, Wiley, London.

W. Rodenacker (1971), Methodishes Konstruieren, Berlin: Springer.

S.R. Rosenthal (1992), Effective Product Design and Development, How to Cut Lead Time and Increase Customer Satisfaction, Business One Irwin, Homewood, Illinois, 60430

M. Sasajima, Y. Kitamura, M. Ikeda and M. Mizoguchi (1996), Representation Language for Behavior and Function: FBRL, Experts Systems with Applications, vol. 10, issue 3, pages 471 – 479.

S.M. Salhieh and A.K. Kamrani (1999), Macro-Level Product Development using design for modularity, Robotics and Computer Integrated Manufacturing, vol. 15, pages 319-329.

R. Sanchez (1996), Strategic product creation: managing new interactions of technology, markets, and organizations, European Management Journal, vol. 14, issue 2, pages 121–138.

K-K Seo, J-H Park, D-S Jang and D. Wallace (2002), Prediction of the Life Cycle Cost Using Statistical and Artificial Neural Network Methods in Conceptual Product Design, International Journal of Computer Integrated Manufacturing, vol. 15, issue 6, pages 541 – 554.

T.R. Sexton, R.H. Silkman and A.J. Hogan (1986), Data envelopment analysis: Critique and extensions, In Measuring Efficiency: An Assessment of Data Envelopment Analysis (Ed.: Silkman, R.H.),Jossey-Bass, San Francisco, pages 73 - 105.

T.W. Simpson, M.D. Bauer, J.K. Allen and F. Mistree (1995), Implementation of DFA in Conceptual and Embodiment Design Using Decision Support Problems, ASME DETC Conference, vol. 1, pages 119 – 126.

M.E. Sosa, S.D. Eppinger and C.M. Rowles (2003), Identifying Modular and Integrative Systems and Their Impact on the Design Team Interactions, Journal of Mechanical Design, Vol. 125, pages 240 – 251.

S. Sosale, M. Hashemian and P. Gu (1997), An Integrated Modular Design Methodology for Life- cycle Engineering, Annals of the CIRP, vol. 46.

G. Stiny (1980), Introduction to Shape and Shape Grammars, Environment and Planning B: Planning and Design, vol. 7, pages 343 – 351.

R. Stone and K. Wood (2000), Development of a Functional Basis for Design, Journal of Mechanical Design, vol. 22, issue 4, pages 359 – 370.

R.B. Stone, K.L Wood and R.H. Crawford (2000), A heuristic method for identifying modules for product architectures, Design Studies, vol. 21, pages 5 - 31.

R. Stone (1997), Towards a Theory of Modular Design, Doctoral Thesis, The University of Texas at Austin.

Z. Strawbridge, D.A. McAdams and R.B. Stone (2002), A Computational Approach to Conceptual Design, Proceedings of the ASME DETC and CIE Conference, Paper No. DETC2002/DTM-34001.

S. Szykman (2002), Architecture and Implementation of a Design Repository System, Proceedings of the ASME DETC and CIE Conference, Paper No. DETC2002/CIE – 34463.

S. Szykman, J.W. Racz and R.D. Sriram (1999), The Representation of Function in Computer Based Design, Proceedings of the ASME DETC Conference, Paper No. DETC99/DTM-8742.

D. Tate, D. Lindholm and V. Harutunian, (1998), Dependencies in axiomatic design, Journal of Integrated Design and Process Technology, vol. 3, pages 159–166.

T. Tomiyama, Y. Umeda and H. Yoshikawa (1993), A CAD for Functional Design, Annals of the CIRP, vol. 42, issue 1, pages 143-146.

S.B. Tor, G.A. Britton, W.Y. Zhang and Y-M Deng (2002), Guiding Functional Design of Mechanical Products through Rule based Causal Behavioral Reasoning, International Journal of Production Research, vol. 40, issue 3, pages 667 – 682.

Y-T Tsai and K-S Wang (1999), The development of modular-based design considering technology complexity, European Journal of Operational Research vol. 119, pages 692-703.

D.G. Ullman (1997), The Mechanical Design Process, New York, McGraw Hill.

K. Ulrich (1995), The role of product architecture in the manufacturing firm, Research Policy, Vol. 24, pages 419–440.

K. Ulrich and S.D. Eppinger, (1995), Product Design and Development ,New York: McGraw-Hill.

K. Ulrich and K. Tung (1991), Fundamentals of product modularity, Proceedings of the ASME Design Engineering Technical Conferences—Conference on Design/Manufacture Integration, vol. 39, pages 73 - 79.

Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura and T. Tomiyama (1996), Supporting Conceptual Design Based on the Function-Behavior-State Modeler, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 10, pages 275-288.

A. Vivek, D.A. McAdams, R.B. Stone (2006), Visual Representations as an Aid to Concept Generation, Proceedings of the ASME DETC and CIE Conference, Paper No. DETC2006 – 99572.

Q. Wang, M. Rao and J. Zhou (1995), Intelligent Systems Approach to Conceptual Design, International Journal of Intelligent Systems, vol. 10, issue 3, pages 259-293.

D.E. Whitney (2004), “Physical limits to modularity”, Working paper, ESD-WP-2003-01.03-ESD, Massachusetts Institute of Technology, Engineering Systems Division.

H. Yoshikawa (1980), General Design Theory and a CAD System, Proceedings of the Working Conference on Man-Machine Communications in CAD/CAM, International Federation for Information Processing, pages 35–58.

H. Yoshikawa and K. Ando (1987), Intelligent CAD in Manufacturing, Annals of the CIRP, vol. 36, pages 77 – 80.

W.Y. Zhang, S.Y. Tor and G.A. Britton (2006), Managing Product Modularity in Product Family Design with Functional Modeling, International Journal of Advanced Manufacturing Technologies, vol. 30, pages 579 – 588.

W.Y. Zhang, S.B. Tor and G.A. Britton (2003), FuncDesigner – a functional design software system, International Journal of Advanced Manufacturing Technology, vol. 22, pages 295 – 305.

Y. Zhang and J.K. Gershenson (2003), An Initial Study of Direct Relationships between Life-cycle Modularity and Life-cycle Cost, vol. 11, issue 2, pages 121 – 128.

W.Y. Zhang, S.B. Tor and G.A. Britton (2002), A Heuristic State - Space Approach to the Functional Design of Mechanical Systems, International Journal of Advanced Manufacturing Technologies, vol.19, pages 235 – 244.

Appendix A

Java Code of the Research Software

1) authen1.java

```
//Authenticates the user to the menu system
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.sql.*;
import java.net.*;
import java.awt.event.*;

public class authen1 extends JFrame implements ActionListener
{
    public javax.swing.JButton jButton1,exitbutton;
    JButton designrep,genconp,genrule,genDFA,back2;
    public javax.swing.JLabel jLabel1,jLabel0,jLabe20,jLabe30;
    public javax.swing.JLabel jLabel2,jLabel4;
    public javax.swing.JPasswordField jPasswordField1;
    public javax.swing.JTextField jTextField1;
    String url,driver;
    JPanel p;
    Connection conn;
    Statement stmt;
    ResultSet rs;
    ImageIcon image1;
    public authen1()
    {
        p = new JPanel();
        this.setSize(700,700);
        this.setTitle ("Concept Generator Tool");
        p.setLayout(new GridBagLayout());
        p.setBackground(Color.WHITE);
    }
    public void authen2 ()
    {
        p.removeAll ();
        p = new JPanel();
        setSize(700,700);
        setResizable(false);
        p.setLayout(new GridBagLayout());
        setForeground(Color.WHITE);
        jLabel0 = new JLabel("Conceptual Generator Tool");
        jLabel0.setFont(new Font("System",Font.BOLD,25));
        jLabel4 = new JLabel("Developed by Saraj Gupta");
        jLabel4.setFont(new Font("System",Font.ITALIC,12));
        jTextField1 = new javax.swing.JTextField(20);
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jPasswordField1 = new javax.swing.JPasswordField(20);
        jButton1 = new javax.swing.JButton();
        exitbutton = new JButton("Exit");
        exitbutton.addActionListener (this);
        jLabel1.setText("User Name:");
        jLabel2.setText("Password:");
        jButton1.setText("Submit");
        jButton1.addActionListener (this);
        image1 = new ImageIcon("C:/jdk/jdk/bin/abb.gif");
        GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
        gridBagConstraintsx00.gridx = 0;
        gridBagConstraintsx00.gridy = 0;
        gridBagConstraintsx00.gridwidth = 3;
        gridBagConstraintsx00.insets = new Insets(5,5,5,5);
        p.add(jLabel0,gridBagConstraintsx00);
        GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
        gridBagConstraintsx01.gridx = 0;
        gridBagConstraintsx01.gridy = 1;
        gridBagConstraintsx01.gridwidth = 2;
        gridBagConstraintsx01.insets = new Insets(5,5,5,5);
        p.add(jLabel1, gridBagConstraintsx01);
        GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
        gridBagConstraintsx02.gridx = 2;
```

```
               gridBagConstraintsx02.gridy = 1;
               gridBagConstraintsx02.insets = new Insets(5,5,5,5);
               gridBagConstraintsx02.gridwidth = 1;
               gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
               p.add(jTextField1, gridBagConstraintsx02);
               GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
               gridBagConstraintsx03.gridx = 0;
               gridBagConstraintsx03.gridy = 2;
               gridBagConstraintsx03.insets = new Insets(5,5,5,5);
               gridBagConstraintsx03.gridwidth = 2;
               gridBagConstraintsx03.fill = GridBagConstraints.BOTH;
               p.add(jLabel2, gridBagConstraintsx03);
               GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
               gridBagConstraintsx04.gridx = 2;
               gridBagConstraintsx04.gridy = 2;
               gridBagConstraintsx04.insets = new Insets(5,5,5,5);
               gridBagConstraintsx04.gridwidth = 1;
               gridBagConstraintsx04.fill = GridBagConstraints.BOTH;
               p.add (jPasswordField1,gridBagConstraintsx04);
               GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
               gridBagConstraintsx05.gridx = 0;
               gridBagConstraintsx05.gridy = 3;
               gridBagConstraintsx05.insets = new Insets(5,5,5,5);
               gridBagConstraintsx05.gridwidth = 3;
               gridBagConstraintsx05.fill = GridBagConstraints.BOTH;
               p.add (jButton1,gridBagConstraintsx05);
               GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
               gridBagConstraintsx06.gridx = 0;
               gridBagConstraintsx06.gridy = 5;
               gridBagConstraintsx06.insets = new Insets(5,5,5,5);
               gridBagConstraintsx06.gridwidth = 1;
               gridBagConstraintsx06.fill = GridBagConstraints.BOTH;
               p.add (new JLabel(image1),gridBagConstraintsx06);
               GridBagConstraints gridBagConstraintsx07 = new GridBagConstraints();
               gridBagConstraintsx07.gridx = 1;
               gridBagConstraintsx07.gridy = 5;
               gridBagConstraintsx07.insets = new Insets(5,5,5,5);
               gridBagConstraintsx07.gridwidth = 2;
               gridBagConstraintsx07.fill = GridBagConstraints.BOTH;
               p.add (jLabel4,gridBagConstraintsx07);
               GridBagConstraints gridBagConstraintsx08 = new GridBagConstraints();
               gridBagConstraintsx08.gridx = 0;
               gridBagConstraintsx08.gridy = 4;
               gridBagConstraintsx08.insets = new Insets(5,5,5,5);
               gridBagConstraintsx08.gridwidth = 3;
               gridBagConstraintsx08.fill = GridBagConstraints.BOTH;
               p.add (exitbutton,gridBagConstraintsx08);
               this.getContentPane().add(p);
               setVisible(true);
           }
            public static void main (String[] args)
           {
                  authen1 aa = new authen1();
                  aa.authen2 ();
           }
           public void actionPerformed(ActionEvent ae)
           {
                      String str = ae.getActionCommand();
                      if(str.equals("Submit"))
                      {
                           try
                           {
                             driver = "com.mysql.jdbc.Driver";
                             url = "jdbc:mysql://localhost:3306/test";
                             Class.forName(driver);
                             conn =DriverManager.getConnection(url,"root","pennstate");
                             stmt = conn.createStatement();
                               rs = stmt.executeQuery("select * from authen;");
                               String ab1 = jTextField1.getText();
                               char ab2[] = jPasswordField1.getPassword();
                               String ab3 = new String(ab2);
                               while (rs.next())
                               {
                                   if((ab1.equals(rs.getString ("name")))&&(ab3.equals
(rs.getString ("password"))))
                                   {
```

```
                                p.removeAll ();
                                this.dispose();
                                main1 m = new main1();
                                m.Gen1();
                            }
                        }
                    }
                    catch(Exception ex)
                    {
                      System.out.println(ex);
                      return;
                    }
                }
                else
                System.exit (0);
        }


}
```

2) bestd.java

```
// Displays the final design
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.sql.*;
import java.net.*;
import javax.imageio.*;
import javax.swing.border.*;
import java.io.*;
import java.awt.event.*;
public class bestd extends JFrame implements ActionListener
{
    JPanel p;
    JTextField[] tf = new JTextField[100];
    String driver,url;
    Connection conn;
    int l=0,ddx1,ab1,num1,ddy1;
    Statement stmt,stmt1;
    ResultSet rs, rse;
    BufferedImage imageo;
    ImageIcon image1;
    StringTokenizer st;
    JButton b1,main1,dfv1;
    JLabel[] img = new JLabel[100];
    JPanel[] p1 = new JPanel[100];
    Border[] linec = new Border[100];
    JLabel[] lbl = new JLabel[100];
    JLabel md;
    GridBagConstraints[][] gridBagConstraints = new GridBagConstraints[10][100];
    GridBagConstraints[] gridBagConstraints1 = new GridBagConstraints[100];
    GridBagConstraints[] gridBagConstraints2 = new GridBagConstraints[100];
    GridBagConstraints[] gridBagConstraints3 = new GridBagConstraints[100];
    String[] mod = new String[20];
    int[][] dd1 = new int[10][100];
    public void init1(int num, int[][] dd, int ddx, int ddy, String[] mod_t)
    {
        ddx1 = ddx;
        num1 = num;
        ddy1 = ddy;
        dd1=dd;
        mod = mod_t;
        p = new JPanel(new GridBagLayout());
        this.setSize(700,700);
        this.setTitle("Final Design");
        md = new JLabel("The Final Design");
        main1 = new JButton("Main Menu");
        main1.addActionListener(this);
        md.setFont(new Font("System",Font.BOLD,25));
        try
        {
            driver = "com.mysql.jdbc.Driver";
            url = "jdbc:mysql://localhost:3306/test";
            Class.forName(driver);
```

```
                    conn =DriverManager.getConnection(url,"root","pennstate");
                    stmt = conn.createStatement();
                    stmt1 = conn.createStatement ();
                    rs = stmt.executeQuery("Select * from final_info where num = " + num + "
order by dfa_index_sum ASC;");
                    while(rs.next())
                    {
                        st = new StringTokenizer(rs.getString ("compon_id"),"$");
                        while(st.hasMoreTokens ())
                        {
                            rse = stmt1.executeQuery("select * from DFA where compid = \'" +
st.nextToken () + "\';");
                            while(rse.next())
                            {
                                l++;
                                imageo = ImageIO.read(new File(rse.getString ("image1")));
                                int w = (int)(imageo.getWidth()*0.8);
                                int h = (int)(imageo.getHeight ()*0.8);
                                Image imagei = imageo.getScaledInstance(w, h,
Image.SCALE_AREA_AVERAGING);
                                BufferedImage image00 = new BufferedImage(w, h,
BufferedImage.TYPE_INT_RGB);
                                Graphics2D g = image00.createGraphics();
                                g.drawImage(imagei, 0, 0, null);
                                g.dispose();
                                image1 = new ImageIcon(image00);
                                img[l] = new JLabel(image1);
                            }
                        }
                    }
                    GridBagConstraints gridBagConstraintsx011a = new GridBagConstraints();
                    gridBagConstraintsx011a.gridx = 0;
                    gridBagConstraintsx011a.gridy = 0;
                    gridBagConstraintsx011a.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx011a.gridwidth = 4;
                    gridBagConstraintsx011a.fill = GridBagConstraints.BOTH;
                    p.add(md, gridBagConstraintsx011a);
                    for(int u=1; u<=ddx; u++)
                    {
                        p1[u] = new JPanel(new GridBagLayout());
                        linec[u] = new LineBorder(Color.BLACK);
                        p1[u].setBorder(linec[u]);
                        lbl[u] = new JLabel(mod[u]);
                        for(int v=1; v<=ddy; v++)
                        {
                            if(dd[u][v]!=0)
                            {
                                gridBagConstraints[u][v] = new GridBagConstraints();
                                gridBagConstraints[u][v].gridx = v;
                                gridBagConstraints[u][v].gridy = 0;
                                gridBagConstraints[u][v].gridwidth = 1;
                                gridBagConstraints[u][v].insets = new Insets(5,5,5,5);
                                p1[u].add(img[dd[u][v]],gridBagConstraints[u][v]);
                            }
                        }
                        gridBagConstraints1[u] = new GridBagConstraints();
                        gridBagConstraints1[u].gridx = 0;
                        gridBagConstraints1[u].gridy = 1;
                        gridBagConstraints1[u].insets = new Insets(5,5,5,5);
                        gridBagConstraints1[u].gridwidth = 2;
                        gridBagConstraints1[u].fill = GridBagConstraints.BOTH;
                        p1[u].add(lbl[u], gridBagConstraints1[u]);
                        gridBagConstraints2[u] = new GridBagConstraints();
                        gridBagConstraints2[u].gridx = u-1;
                        gridBagConstraints2[u].gridy = 1;
                        gridBagConstraints2[u].insets = new Insets(5,5,5,5);
                        gridBagConstraints2[u].gridwidth = 1;
                        gridBagConstraints2[u].fill = GridBagConstraints.BOTH;
                        p.add(p1[u], gridBagConstraints2[u]);
                    }
                    GridBagConstraints gridBagConstraintsx015 = new GridBagConstraints();
                    gridBagConstraintsx015.gridx = 0;
                    gridBagConstraintsx015.gridy = 2;
                    gridBagConstraintsx015.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx015.gridwidth = 4;
                    gridBagConstraintsx015.fill = GridBagConstraints.BOTH;
                    p.add(main1, gridBagConstraintsx015);
                    this.getContentPane().add(p);
                    setVisible(true);
                }
                catch (Exception ex)
                {
                    System.out.println(ex);
                }
            }
            public void actionPerformed(ActionEvent ae)
            {
                String str = ae.getActionCommand();
                if(str.equals("Main Menu"))
                {
                    this.dispose();
                    main1 mm = new main1();
                    mm.Gen1 ();
                }
            }
           /* public static void main(String[] args)
            {
                mod mm = new mod();
                mm.init1();
            }*/
        }
```

3) DFA.java

```
        //Inputs all the DFA indices
        import java.awt.*;
        import java.applet.*;
        import java.awt.event.*;
        import java.sql.*;
        import javax.swing.*;
        import java.util.*;

        public class DFA extends JFrame implements ItemListener,ActionListener
        {
               JTextField compid,dfa,rec;
               JComboBox wt, uc,stiff,hardness,len,shape,size,compos,sym,join,baseid;
               JLabel
wt1,uc1,stiff1,hard1,len1,shape1,size1,compos1,bas1,mov1,align1,join1,sym1,round1,compid1,baseid1
;
               JButton calcdfa,submit,dfa1,main1,loadim;
               JRadioButton yes,no,round,notroun,st,notst,chamf,notchamf;
               ButtonGroup roun,cham,bas,mov;
               int rou=0,record=0,a2;
               double DFA = 0.0;
               Connection conn;
               Statement stmt;
               ResultSet rset;
               String driver,url,abc,abc1;
               FileDialog dialog1;
                ItemSelectable a;
                JPanel p;
               public void init1()
               {
                       try
                       {
                              driver = "com.mysql.jdbc.Driver";
                              url = "jdbc:mysql://localhost:3306/test";
                              Class.forName(driver);
                              conn =DriverManager.getConnection(url,"root","pennstate");
                              stmt = conn.createStatement();
                              rset = stmt.executeQuery("Select * from DFA");
                              while (rset.next())
                              record++;
                       }
                       catch(Exception ex)
                       {
                              System.out.println(ex);
                              return;
                       }
                        p = new JPanel();
                        this.setSize(700,700);
                        this.setResizable(false);
```

```
 this.setTitle ("Inserting Component(s) Menu");
 p.setLayout(new GridBagLayout());
 this.setBackground(Color.WHITE);
submit = new JButton("Insert into Repository");
dfa1 = new JButton("Calculate DFA Index");
dfa1.addActionListener(this);
submit.addActionListener(this);
main1 = new JButton("Design Repository Menu");
main1.addActionListener(this);
loadim = new JButton("Insert Image File");
loadim.addActionListener(this);
wt1 = new JLabel("What is the approximate weight range in grams?");
bas1 = new JLabel("Does the component have a base?");
uc1 = new JLabel("What are the number of unique components present?");
stiff1 = new JLabel("What is the stiffness (Young's Modulus) in Pa?");
hard1 = new JLabel("What is the vulnerability hardness in Kgf/mm-2 ?");
round1 = new JLabel("What is the overall structure?");
len1 = new JLabel("What is the maximum length in mm ?");
shape1 = new JLabel("What is the shape?");
size1 = new JLabel("What is the size?");
compos1 = new JLabel("What is the composing direction?");
mov1 = new JLabel("What is the composing movement?");
align1 =  new JLabel("What is the alignment characteristic?");
join1 = new JLabel("What is the joining method?");
sym1 = new JLabel("What is the symmetry?");
compid1 = new JLabel("Component ID: ");
baseid1 = new JLabel("Base ID: ");
compid = new JTextField(6);
dfa = new JTextField(6);
//dfa.setEditable(false);
rec = new JTextField(15);
rec.setText("Component(s) in Repository: " + Integer.toString(record));
rec.setEditable(false);
roun = new ButtonGroup();
cham = new ButtonGroup();
bas = new ButtonGroup();
mov = new ButtonGroup();
yes = new JRadioButton("Yes",true);
 yes.setActionCommand ("yes1");
 no = new JRadioButton("No",false);
 no.setActionCommand ("no1");
 round = new JRadioButton("Round (L,D)",true);
 round.setActionCommand ("rou");
 notroun = new JRadioButton("Not Round (A,B,C)",false);
round.addItemListener(this);
notroun.addItemListener(this);
st = new JRadioButton("Straight Line",true);
notst = new JRadioButton("Not Straight Line",false);
chamf = new JRadioButton("Chamfer",true);
notchamf = new JRadioButton("No Chamfer",false);
notroun.setActionCommand ("notrou");
 st.setActionCommand ("Stline");
 notst.setActionCommand ("NotSt");
 chamf.setActionCommand ("chamf");
 notchamf.setActionCommand ("notchamf");
 cham.add(chamf);
 cham.add (notchamf);
 bas.add(yes);
 bas.add (no);
 mov.add (st);
 mov.add (notst);
 roun.add (round);
 roun.add (notroun);
 baseid = new JComboBox();
try
{
        String b="",c;
        rset = stmt.executeQuery("Select * from rules");
        while (rset.next())
        {
                c = rset.getString("base_id");
                if(!c.equals(b))
                baseid.addItem(c);
                b=c;
        }
}
```

```
catch(Exception ex)
{
        System.out.println(ex);
        return;
}
wt = new JComboBox();
wt.addItem("0.1 < G < 2000");
wt.addItem("0.01 <= G <= 0.1");
wt.addItem("2000 <= G <= 6000");
wt.addItem("G < 0.01");
wt.addItem("G > 6000");

join = new JComboBox();
join.addItem("Snap/Screwing/Adhesive Bonding");
join.addItem("Press Fitting/Welding/Riveting/Soldering");

uc = new JComboBox();
uc.addItem("UC < 10");
uc.addItem("UC >= 10");

stiff = new JComboBox();
stiff.addItem("YM > (7.0E + 10)Pa");
stiff.addItem("YM < (7.0E + 10)Pa");

hardness = new JComboBox();
hardness.addItem("H <= 80");
hardness.addItem("80 < H <= 150");
hardness.addItem("H > 150");

len = new JComboBox();
len.addItem("5 < L <= 50");
len.addItem("2 <= L <= 5");
len.addItem("50 <= L <= 2000");
len.addItem("L < 2");
len.addItem("L > 2000");

shape = new JComboBox();
shape.addItem("L/D < 0.8");
shape.addItem("0.8 <= L/D <= 1.5");
shape.addItem("L/D > 1.5");

size = new JComboBox();
size.addItem("0.25 < t <= 50");
size.addItem("t <= 0.25");
size.addItem("t > 50");

compos = new JComboBox();
compos.addItem("Top-Down");
compos.addItem("Side-In");
compos.addItem("Bottom-Up");
compos.addItem("Other");

sym = new JComboBox();
sym.addItem("alpha AND beta symmetric");
sym.addItem("alpha symmetric AND beta asymmetric");
sym.addItem("beta symmetric AND alpha symmetric");
sym.addItem("apha AND beta asymmetric");

GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
gridBagConstraintsx01.gridx = 0;
gridBagConstraintsx01.gridy = 0;
gridBagConstraintsx01.insets = new Insets(5,5,5,5);
p.add(compid1, gridBagConstraintsx01);
GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
gridBagConstraintsx02.gridx = 1;
gridBagConstraintsx02.gridy = 0;
gridBagConstraintsx02.insets = new Insets(5,5,5,5);
gridBagConstraintsx02.gridwidth = 2;
gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
p.add(compid, gridBagConstraintsx02);

GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
gridBagConstraintsx03.gridx = 0;
gridBagConstraintsx03.gridy = 1;
gridBagConstraintsx03.insets = new Insets(5,5,5,5);
p.add(loadim, gridBagConstraintsx03);
```

```
GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
gridBagConstraintsx04.gridx = 1;
gridBagConstraintsx04.gridy = 1;
gridBagConstraintsx04.insets = new Insets(5,5,5,5);
gridBagConstraintsx04.gridwidth = 1;
gridBagConstraintsx04.fill = GridBagConstraints.BOTH;
p.add(baseid1, gridBagConstraintsx04);
GridBagConstraints gridBagConstraintsx04b = new GridBagConstraints();
gridBagConstraintsx04b.gridx = 2;
gridBagConstraintsx04b.gridy = 1;
gridBagConstraintsx04b.insets = new Insets(5,5,5,5);
gridBagConstraintsx04b.gridwidth = 1;
gridBagConstraintsx04b.fill = GridBagConstraints.BOTH;
p.add(baseid, gridBagConstraintsx04b);

GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
gridBagConstraintsx05.gridx = 0;
gridBagConstraintsx05.gridy = 2;
gridBagConstraintsx05.insets = new Insets(5,5,5,5);
p.add(wt1, gridBagConstraintsx05);
GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
gridBagConstraintsx06.gridx = 1;
gridBagConstraintsx06.gridy = 2;
gridBagConstraintsx06.insets = new Insets(5,5,5,5);
gridBagConstraintsx06.gridwidth = 2;
gridBagConstraintsx06.fill = GridBagConstraints.BOTH;
p.add(wt, gridBagConstraintsx06);

GridBagConstraints gridBagConstraintsx07 = new GridBagConstraints();
gridBagConstraintsx07.gridx = 0;
gridBagConstraintsx07.gridy = 3;
gridBagConstraintsx07.insets = new Insets(5,5,5,5);
p.add(uc1, gridBagConstraintsx07);
GridBagConstraints gridBagConstraintsx08 = new GridBagConstraints();
gridBagConstraintsx08.gridx = 1;
gridBagConstraintsx08.gridy = 3;
gridBagConstraintsx08.insets = new Insets(5,5,5,5);
gridBagConstraintsx08.gridwidth = 2;
gridBagConstraintsx08.fill = GridBagConstraints.BOTH;
p.add(uc, gridBagConstraintsx08);

GridBagConstraints gridBagConstraintsx09 = new GridBagConstraints();
gridBagConstraintsx09.gridx = 0;
gridBagConstraintsx09.gridy = 4;
gridBagConstraintsx09.insets = new Insets(5,5,5,5);
p.add(bas1, gridBagConstraintsx09);
GridBagConstraints gridBagConstraintsx010 = new GridBagConstraints();
gridBagConstraintsx010.gridx = 1;
gridBagConstraintsx010.gridy = 4;
gridBagConstraintsx010.insets = new Insets(5,5,5,5);
p.add(yes, gridBagConstraintsx010);
GridBagConstraints gridBagConstraintsx011 = new GridBagConstraints();
gridBagConstraintsx011.gridx = 2;
gridBagConstraintsx011.gridy = 4;
gridBagConstraintsx011.insets = new Insets(5,5,5,5);
p.add(no, gridBagConstraintsx011);

GridBagConstraints gridBagConstraintsx012 = new GridBagConstraints();
gridBagConstraintsx012.gridx = 0;
gridBagConstraintsx012.gridy = 5;
gridBagConstraintsx012.insets = new Insets(5,5,5,5);
p.add(stiff1, gridBagConstraintsx012);
GridBagConstraints gridBagConstraintsx013 = new GridBagConstraints();
gridBagConstraintsx013.gridx = 1;
gridBagConstraintsx013.gridy = 5;
gridBagConstraintsx013.insets = new Insets(5,5,5,5);
gridBagConstraintsx013.gridwidth = 2;
gridBagConstraintsx013.fill = GridBagConstraints.BOTH;
p.add(stiff, gridBagConstraintsx013);

GridBagConstraints gridBagConstraintsx014 = new GridBagConstraints();
gridBagConstraintsx014.gridx = 0;
gridBagConstraintsx014.gridy = 6;
gridBagConstraintsx014.insets = new Insets(5,5,5,5);
p.add(hard1, gridBagConstraintsx014);
GridBagConstraints gridBagConstraintsx015 = new GridBagConstraints();
```

```
gridBagConstraintsx015.gridx = 1;
gridBagConstraintsx015.gridy = 6;
gridBagConstraintsx015.insets = new Insets(5,5,5,5);
gridBagConstraintsx015.gridwidth = 2;
gridBagConstraintsx015.fill = GridBagConstraints.BOTH;
p.add(hardness, gridBagConstraintsx015);

GridBagConstraints gridBagConstraintsx016 = new GridBagConstraints();
gridBagConstraintsx016.gridx = 0;
gridBagConstraintsx016.gridy = 7;
gridBagConstraintsx016.insets = new Insets(5,5,5,5);
p.add(round1, gridBagConstraintsx016);
GridBagConstraints gridBagConstraintsx017 = new GridBagConstraints();
gridBagConstraintsx017.gridx = 1;
gridBagConstraintsx017.gridy = 7;
gridBagConstraintsx017.insets = new Insets(5,5,5,5);
p.add(round, gridBagConstraintsx017);
GridBagConstraints gridBagConstraintsx018 = new GridBagConstraints();
gridBagConstraintsx018.gridx = 2;
gridBagConstraintsx018.gridy = 7;
gridBagConstraintsx018.insets = new Insets(5,5,5,5);
p.add(notroun, gridBagConstraintsx018);

GridBagConstraints gridBagConstraintsx019 = new GridBagConstraints();
gridBagConstraintsx019.gridx = 0;
gridBagConstraintsx019.gridy = 8;
gridBagConstraintsx019.insets = new Insets(5,5,5,5);
p.add(len1, gridBagConstraintsx019);
GridBagConstraints gridBagConstraintsx020 = new GridBagConstraints();
gridBagConstraintsx020.gridx = 1;
gridBagConstraintsx020.gridy = 8;
gridBagConstraintsx020.insets = new Insets(5,5,5,5);
gridBagConstraintsx020.gridwidth = 2;
gridBagConstraintsx020.fill = GridBagConstraints.BOTH;
p.add(len, gridBagConstraintsx020);

GridBagConstraints gridBagConstraintsx021 = new GridBagConstraints();
gridBagConstraintsx021.gridx = 0;
gridBagConstraintsx021.gridy = 9;
gridBagConstraintsx021.insets = new Insets(5,5,5,5);
p.add(shape1, gridBagConstraintsx021);
GridBagConstraints gridBagConstraintsx022 = new GridBagConstraints();
gridBagConstraintsx022.gridx = 1;
gridBagConstraintsx022.gridy = 9;
gridBagConstraintsx022.insets = new Insets(5,5,5,5);
gridBagConstraintsx022.gridwidth = 2;
gridBagConstraintsx022.fill = GridBagConstraints.BOTH;
p.add(shape, gridBagConstraintsx022);

GridBagConstraints gridBagConstraintsx023 = new GridBagConstraints();
gridBagConstraintsx023.gridx = 0;
gridBagConstraintsx023.gridy = 10;
gridBagConstraintsx023.insets = new Insets(5,5,5,5);
p.add(size1, gridBagConstraintsx023);
GridBagConstraints gridBagConstraintsx024 = new GridBagConstraints();
gridBagConstraintsx024.gridx = 1;
gridBagConstraintsx024.gridy = 10;
gridBagConstraintsx024.insets = new Insets(5,5,5,5);
gridBagConstraintsx024.gridwidth = 2;
gridBagConstraintsx024.fill = GridBagConstraints.BOTH;
p.add(size, gridBagConstraintsx024);

GridBagConstraints gridBagConstraintsx025 = new GridBagConstraints();
gridBagConstraintsx025.gridx = 0;
gridBagConstraintsx025.gridy = 11;
gridBagConstraintsx025.insets = new Insets(5,5,5,5);
p.add(compos1, gridBagConstraintsx025);
GridBagConstraints gridBagConstraintsx026 = new GridBagConstraints();
gridBagConstraintsx026.gridx = 1;
gridBagConstraintsx026.gridy = 11;
gridBagConstraintsx026.insets = new Insets(5,5,5,5);
gridBagConstraintsx026.gridwidth = 2;
gridBagConstraintsx026.fill = GridBagConstraints.BOTH;
p.add(compos, gridBagConstraintsx026);

GridBagConstraints gridBagConstraintsx027 = new GridBagConstraints();
```

```
gridBagConstraintsx027.gridx = 0;
gridBagConstraintsx027.gridy = 12;
gridBagConstraintsx027.insets = new Insets(5,5,5,5);
p.add(mov1, gridBagConstraintsx027);
GridBagConstraints gridBagConstraintsx028 = new GridBagConstraints();
gridBagConstraintsx028.gridx = 1;
gridBagConstraintsx028.gridy = 12;
gridBagConstraintsx028.insets = new Insets(5,5,5,5);
p.add(st, gridBagConstraintsx028);
GridBagConstraints gridBagConstraintsx029 = new GridBagConstraints();
gridBagConstraintsx029.gridx = 2;
gridBagConstraintsx029.gridy = 12;
gridBagConstraintsx029.insets = new Insets(5,5,5,5);
p.add(notst, gridBagConstraintsx029);

GridBagConstraints gridBagConstraintsx030 = new GridBagConstraints();
gridBagConstraintsx030.gridx = 0;
gridBagConstraintsx030.gridy = 13;
gridBagConstraintsx030.insets = new Insets(5,5,5,5);
p.add(align1, gridBagConstraintsx030);
GridBagConstraints gridBagConstraintsx031 = new GridBagConstraints();
gridBagConstraintsx031.gridx = 1;
gridBagConstraintsx031.gridy = 13;
gridBagConstraintsx031.insets = new Insets(5,5,5,5);
p.add(chamf, gridBagConstraintsx031);
GridBagConstraints gridBagConstraintsx032 = new GridBagConstraints();
gridBagConstraintsx032.gridx = 2;
gridBagConstraintsx032.gridy = 13;
gridBagConstraintsx032.insets = new Insets(5,5,5,5);
p.add(notchamf, gridBagConstraintsx032);

GridBagConstraints gridBagConstraintsx033 = new GridBagConstraints();
gridBagConstraintsx033.gridx = 0;
gridBagConstraintsx033.gridy = 14;
gridBagConstraintsx033.insets = new Insets(5,5,5,5);
p.add(join1, gridBagConstraintsx033);
GridBagConstraints gridBagConstraintsx034 = new GridBagConstraints();
gridBagConstraintsx034.gridx = 1;
gridBagConstraintsx034.gridy = 14;
gridBagConstraintsx034.insets = new Insets(5,5,5,5);
gridBagConstraintsx034.gridwidth = 2;
gridBagConstraintsx034.fill = GridBagConstraints.BOTH;
p.add(join, gridBagConstraintsx034);

GridBagConstraints gridBagConstraintsx035 = new GridBagConstraints();
gridBagConstraintsx035.gridx = 0;
gridBagConstraintsx035.gridy = 15;
gridBagConstraintsx035.insets = new Insets(5,5,5,5);
p.add(sym1, gridBagConstraintsx035);
GridBagConstraints gridBagConstraintsx036 = new GridBagConstraints();
gridBagConstraintsx036.gridx = 1;
gridBagConstraintsx036.gridy = 15;
gridBagConstraintsx036.insets = new Insets(5,5,5,5);
gridBagConstraintsx036.gridwidth = 2;
gridBagConstraintsx036.fill = GridBagConstraints.BOTH;
p.add(sym, gridBagConstraintsx036);

GridBagConstraints gridBagConstraintsx037 = new GridBagConstraints();
gridBagConstraintsx037.gridx = 0;
gridBagConstraintsx037.gridy = 16;
gridBagConstraintsx037.insets = new Insets(5,5,5,5);
p.add(dfa1, gridBagConstraintsx037);
GridBagConstraints gridBagConstraintsx038 = new GridBagConstraints();
gridBagConstraintsx038.gridx = 1;
gridBagConstraintsx038.gridy = 16;
gridBagConstraintsx038.insets = new Insets(5,5,5,5);
gridBagConstraintsx038.gridwidth = 2;
gridBagConstraintsx038.fill = GridBagConstraints.BOTH;
p.add(dfa, gridBagConstraintsx038);

GridBagConstraints gridBagConstraintsx039 = new GridBagConstraints();
gridBagConstraintsx039.gridx = 0;
gridBagConstraintsx039.gridy = 17;
gridBagConstraintsx039.insets = new Insets(5,5,5,5);
gridBagConstraintsx039.fill = GridBagConstraints.BOTH;
p.add(rec, gridBagConstraintsx039);
```

```
                GridBagConstraints gridBagConstraintsx040 = new GridBagConstraints();
                gridBagConstraintsx040.gridx = 1;
                gridBagConstraintsx040.gridy = 17;
                gridBagConstraintsx040.insets = new Insets(5,5,5,5);
                gridBagConstraintsx040.gridwidth = 1;
                gridBagConstraintsx040.fill = GridBagConstraints.BOTH;
                p.add(submit, gridBagConstraintsx040);
                GridBagConstraints gridBagConstraintsx041 = new GridBagConstraints();
                gridBagConstraintsx041.gridx = 2;
                gridBagConstraintsx041.gridy = 17;
                gridBagConstraintsx041.insets = new Insets(5,5,5,5);
                gridBagConstraintsx041.gridwidth = 1;
                gridBagConstraintsx041.fill = GridBagConstraints.BOTH;
                p.add(main1, gridBagConstraintsx041);
                 this.getContentPane().add(p);
                 this.setVisible(true);
        }
        public void itemStateChanged(ItemEvent ie)
        {
                a = ie.getItemSelectable ();
                if(a == round)
                {
                        shape.removeAllItems();
                        shape.addItem("L/D < 0.8");
                        shape.addItem("0.8 <= L/D <= 1.5");
                        shape.addItem("L/D > 1.5");

                        size.removeAllItems();
                        size.addItem("0.25 < t <= 50");
                        size.addItem("t <= 0.25");
                        size.addItem("t > 50");

                        sym.removeAllItems();
                        sym.addItem("alpha AND beta symmetric");
                        sym.addItem("alpha symmetric AND beta asymmetric");
                        sym.addItem("beta symmetric AND alpha symmetric");
                        sym.addItem("apha AND beta asymmetric");
                }
                else
                {
                        shape.removeAllItems();
                        shape.addItem("A/B <= 3 AND A/C > 4");
                        shape.addItem("A/B > 3");
                        shape.addItem("A/B <= 3 AND A/C <= 4");

                        size.removeAllItems();
                        size.addItem("5 < L <= 50");
                        size.addItem("2 <= L <= 5");
                        size.addItem("L < 2");
                        size.addItem("50 <= L <= 2000");
                        size.addItem("L > 2000");


                        sym.removeAllItems();
                        sym.addItem("180 deg symmetric about many axis");
                        sym.addItem("180 deg symmetric about one axis");
                        sym.addItem("Non Symmetric");
                }
        }
        public void actionPerformed(ActionEvent ae)
        {
                String str = ae.getActionCommand();
                if(str.equals("Calculate DFA Index"))
                {
                        DFA = 0.0;
                        String wei = (String) wt.getSelectedItem();
                        if(wei.equals("0.1 < G < 2000")) DFA=1;
                        else if(wei.equals("0.01 <= G <= 0.1")) DFA=2;
                        else if(wei.equals("2000 <= G <= 6000")) DFA=2;
                        else if(wei.equals("G < 0.01")) DFA=4;
                        else DFA=4;

                        String uco = (String) uc.getSelectedItem();
                        if(uco.equals("UC < 10")) DFA+=1;
                        else DFA+=4;
```

```
                              String a1 = (String) bas.getSelection().getActionCommand();
                              if(a1.equals("yes1")) DFA+=1;
                              else DFA+=4;

                              String sti = (String) stiff.getSelectedItem();
                              if(sti.equals("YM > (7.0E + 10)Pa")) DFA+=1;
                              else DFA+=4;

                              String har = (String) hardness.getSelectedItem();
                              if(har.equals("H <= 80")) DFA+=1;
                              else if(har.equals("80 < H <= 150")) DFA+=2;
                              else DFA+=4;

                              String leng = (String) len.getSelectedItem();
                              if(leng.equals("5 < L <= 50")) DFA+=1;
                              else if(leng.equals("2 <= L <= 5")) DFA+=2;
                              else if(leng.equals("50 <= L <= 2000")) DFA+=2;

                              else if(leng.equals("L < 2")) DFA+=4;

                              else DFA+=4;

                              String shap = (String) shape.getSelectedItem();
                              if(shap.equals("L/D < 0.8")) DFA+=1;
                              else if(shap.equals("0.8 <= L/D <= 1.5")) DFA+=2;
                              else if(shap.equals("L/D > 1.5")) DFA+=4;
                              else if(shap.equals("A/B <= 3 AND A/C > 4")) DFA+=1;
                              else if(shap.equals("A/B > 3")) DFA+=1;
                              else DFA+=2;

                              String siz = (String) size.getSelectedItem();
                              if(siz.equals("0.25 < t <= 50")) DFA+=1;
                              else if(siz.equals("t <= 0.25")) DFA+=4;
                              else if(siz.equals("t > 50")) DFA+=4;
                              else if(siz.equals("5 < L <= 50")) DFA+=1;
                              else if(siz.equals("2 <= L <= 5")) DFA+=2;
                              else if(siz.equals("L < 2")) DFA+=4;
                              else if(siz.equals("50 <= L <= 2000")) DFA+=2;
                              else DFA+=4;

                              String com = (String) compos.getSelectedItem();
                              if(com.equals("Top-Down")) DFA+=1;
                              else if(com.equals("Side-In")) DFA+=2;
                              else if(com.equals("Bottom-Up")) DFA+=4;
                              else DFA+=6;

                              String movem = mov.getSelection().getActionCommand();
                              if(movem.equals("Stline")) DFA+=1;
                              else DFA+=2;

                              String cha = cham.getSelection().getActionCommand();
                              if(cha.equals("chamf")) DFA+=1;
                              else DFA+=4;

                              String joi = (String) join.getSelectedItem();
                              if(joi.equals("Snap/Screwing/Adhesive Bonding")) DFA+=1;
                              else if(joi.equals("Press Fitting/Welding/Riveting/Soldering"))
DFA+=2;

                              String symm = (String) sym.getSelectedItem();
                              if(symm.equals("alpha AND beta symmetric")) DFA+=1;
                              else if(symm.equals("alpha symmetric AND beta asymmetric")) DFA+=2;
                              else if(symm.equals("beta symmetric AND alpha symmetric")) DFA+=2;
                              else if(symm.equals("apha AND beta asymmetric")) DFA+=4;

                              else if(symm.equals("180 deg symmetric about many axis")) DFA+=1;
                              else if(symm.equals("180 deg symmetric about one axis")) DFA+=2;
                              else if(symm.equals("Non Symmetric")) DFA+=4;
                              DFA = (10*(DFA-13.0))/37.0;
                              dfa.setText(Double.toString(DFA));
                       }
                       else if(str.equals("Insert into Repository"))
                       {
                            int response = JOptionPane.showConfirmDialog (this,"Are you sure you
want to insert into design repository?","Confirm Insert
Dialog",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
```

```
                           if(response == JOptionPane.YES_OPTION)
                           {
                                String comid = compid.getText();
                              String basid = (String) baseid.getSelectedItem();
                              baseid.setSelectedIndex(0);
                              compid.setText("");
                              sym.setSelectedIndex(0);
                              join.setSelectedIndex(0);
                              wt.setSelectedIndex(0);
                              uc.setSelectedIndex(0);
                              stiff.setSelectedIndex(0);
                              hardness.setSelectedIndex(0);
                              len.setSelectedIndex(0);
                              shape.setSelectedIndex(0);
                              size.setSelectedIndex(0);
                              compos.setSelectedIndex(0);
                                ButtonModel model1 = yes.getModel();
                              bas.setSelected(model1,true);
                                ButtonModel model2 = st.getModel();
                              mov.setSelected (model2,true);
                                ButtonModel model3 = chamf.getModel();
                              cham.setSelected(model3, true);
                                DFA = Double.parseDouble (dfa.getText());
                              dfa.setText("");
                              record = 0;
                              try
                              {
                                     int a2 = 0;
                                       rset = stmt.executeQuery("Select * from DFA");
                                     while (rset.next())
                                     record++;
                                       a2 = record + 1;
                                       stmt.executeUpdate("insert into DFA values("+ a2 + ",\'"
+ basid +"\',\'" + comid +"\'," + DFA +",\'"+ abc1 +"\');");
                                     rec.setText("Component(s) in Repository: " +
Integer.toString(a2));
                              }
                              catch(Exception ex)
                              {
                                     System.out.println(ex);
                                     return;
                              }
                                this.dispose();
                                main2 m = new main2();
                                m.Gen2();
                           }
                      }
                      else if(str.equals("Insert Image File"))
                      {
                              Frame myFrame = getFrame(loadim);
                                dialog1 = new FileDialog(myFrame, "Open File", FileDialog.LOAD);
                              dialog1.setVisible(true);
                              abc = dialog1.getDirectory()+dialog1.getFile();
                              StringTokenizer st1 = new StringTokenizer(abc,"\\");
                              abc1 = st1.nextToken();
                              while(st1.hasMoreTokens())
                              abc1 += ("/" + st1.nextToken());
                              System.out.println(abc1);
                      }
                      else
                      {
                              try
                              {
                                     this.dispose();
                                       main2 m = new main2();
                                       m.Gen2();
                                     stmt.close();
                                     conn.close();
                              }
                              catch(Exception ex)
                              {
                                     System.out.println(ex);
                                     return;
                              }
                      }
               }
```

```
        static Frame getFrame (Component c)
        {
                Frame frame1 = null;
                 while ((c= c.getParent())!= null)
               {
                       if (c instanceof Frame)
                       frame1 = (Frame) c;
                }
                return frame1;
        }
}
```

4) DFA2.java

```
// To view all the components stored in Design Repository
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.sql.*;
import java.net.*;
import javax.imageio.*;
import java.io.*;
import java.awt.event.*;

public class DFA2 extends JFrame implements ActionListener
{
    JPanel p;
    ImageIcon image1;
    JLabel
jLabe20,compid,baseid,compid1,baseid1,input1,input2,input3,output1,output2,output3,sub1,sub2,sub3
,dfa2;
    JButton back,front,main1,delet,updat;
    String driver,url,aa,comp,bas,inp1,su1,oup1,inp2,su2,oup2,inp3,su3,oup3,dfa1;
    Connection conn;
    Statement stmt,stmt2;
    ResultSet rs;
    BufferedImage imageo;
    FileInputStream fis;
    FileOutputStream fos;
    File jsv;
    int key=1,record = 0;
    public DFA2()
    {
        p = new JPanel(new GridBagLayout());
        this.setSize(700,700);
        this.setResizable(false);
        this.setTitle("View Components Menu");
    }
    public void Gen1()
    {
        jLabe20 = new JLabel("View Component(s)");
        jLabe20.setFont(new Font("System",Font.BOLD,25));
        dfa2 = new JLabel("DFA Index:");
        compid = new JLabel("Component ID:");
        baseid = new JLabel("Base ID:");
        delet = new JButton("Delete Component");
        delet.addActionListener (this);
        updat = new JButton("Update Component");
        updat.addActionListener (this);
        JButton back = new JButton("<<");
        back.addActionListener (this);
        if(key <= 1) back.setEnabled (false);
        JButton front = new JButton(">>");
        front.addActionListener (this);
        JButton main1 = new JButton("Design Repository Menu");
        main1.addActionListener (this);
        try
        {
            driver = "com.mysql.jdbc.Driver";
            url = "jdbc:mysql://localhost:3306/test";
            Class.forName(driver);
            conn =DriverManager.getConnection(url,"root","pennstate");
            stmt = conn.createStatement();
            rs = stmt.executeQuery("select * from DFA");
            while(rs.next ())
```

```
                    {
                        record++;
                    }
                    if(key >= record) front.setEnabled (false);
                    rs = stmt.executeQuery("select * from DFA,rules where number = " + key + "
and base_id = (select baseid from DFA where number = " + key + ") order by number ASC;");
                    while(rs.next ())
                    {
                        aa = rs.getString("image1");
                        dfa1 = rs.getString ("dfa");
                        comp = rs.getString ("compid");
                        bas = rs.getString ("base_id");
                        inp1 = rs.getString("input1");
                        oup1 = rs.getString ("output1");
                        su1 = rs.getString ("subfunction");
                        inp2 = rs.getString("input2");
                        oup2 = rs.getString ("output2");
                        su2 = rs.getString ("subfunction2");
                        inp3 = rs.getString("input3");
                        oup3 = rs.getString ("output3");
                        su3 = rs.getString ("subfunction3");
                        baseid1 = new JLabel(bas);
                        compid1 = new JLabel(comp);
                        input1 = new JLabel(inp1);
                        sub1 = new JLabel(su1);
                        output1 = new JLabel(oup1);
                        input2 = new JLabel(inp2);
                        sub2 = new JLabel(su2);
                        output2 = new JLabel(oup2);
                        input3 = new JLabel(inp3);
                        sub3 = new JLabel(su3);
                        output3 = new JLabel(oup3);
                        imageo = ImageIO.read(new File(aa));
                        imageo.getScaledInstance(200, 200, Image.SCALE_SMOOTH);
                        break;
                    }

                }
                catch(Exception ex)
                {
                    System.out.println(ex);
                    return;
                }
                image1 = new ImageIcon(imageo);
                GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
                gridBagConstraintsx00.gridx = 0;
                gridBagConstraintsx00.gridy = 0;
                gridBagConstraintsx00.gridwidth = 3;
                gridBagConstraintsx00.insets = new Insets(5,5,5,5);
                p.add(jLabe20,gridBagConstraintsx00);
                GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
                gridBagConstraintsx01.gridx = 0;
                gridBagConstraintsx01.gridy = 1;
                gridBagConstraintsx01.gridwidth = 1;
                gridBagConstraintsx01.insets = new Insets(5,5,5,5);
                p.add(new JLabel(image1), gridBagConstraintsx01);
                GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
                gridBagConstraintsx02.gridx = 1;
                gridBagConstraintsx02.gridy = 1;
                gridBagConstraintsx02.insets = new Insets(5,5,5,5);
                gridBagConstraintsx02.gridwidth = 1;
                gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
                p.add(compid, gridBagConstraintsx02);
                GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
                gridBagConstraintsx03.gridx = 2;
                gridBagConstraintsx03.gridy = 1;
                gridBagConstraintsx03.insets = new Insets(5,5,5,5);
                gridBagConstraintsx03.gridwidth = 1;
                gridBagConstraintsx03.fill = GridBagConstraints.BOTH;
                p.add(compid1, gridBagConstraintsx03);
                GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
                gridBagConstraintsx04.gridx = 1;
                gridBagConstraintsx04.gridy = 2;
                gridBagConstraintsx04.insets = new Insets(5,5,5,5);
                gridBagConstraintsx04.gridwidth = 1;
                gridBagConstraintsx04.fill = GridBagConstraints.BOTH;
```

```
p.add (baseid,gridBagConstraintsx04);
GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
gridBagConstraintsx05.gridx = 2;
gridBagConstraintsx05.gridy = 2;
gridBagConstraintsx05.insets = new Insets(5,5,5,5);
gridBagConstraintsx05.gridwidth = 1;
gridBagConstraintsx05.fill = GridBagConstraints.BOTH;
p.add (baseid1,gridBagConstraintsx05);
GridBagConstraints gridBagConstraintsx05a = new GridBagConstraints();
gridBagConstraintsx05a.gridx = 1;
gridBagConstraintsx05a.gridy = 3;
gridBagConstraintsx05a.insets = new Insets(5,5,5,5);
gridBagConstraintsx05a.gridwidth = 1;
gridBagConstraintsx05a.fill = GridBagConstraints.BOTH;
p.add (dfa2,gridBagConstraintsx05a);
GridBagConstraints gridBagConstraintsx05b = new GridBagConstraints();
gridBagConstraintsx05b.gridx = 2;
gridBagConstraintsx05b.gridy = 3;
gridBagConstraintsx05b.insets = new Insets(5,5,5,5);
gridBagConstraintsx05b.gridwidth = 1;
gridBagConstraintsx05b.fill = GridBagConstraints.BOTH;
p.add (new JLabel(dfa1),gridBagConstraintsx05b);
GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
gridBagConstraintsx06.gridx = 0;
gridBagConstraintsx06.gridy = 4;
gridBagConstraintsx06.insets = new Insets(5,5,5,5);
gridBagConstraintsx06.gridwidth = 1;
gridBagConstraintsx06.fill = GridBagConstraints.BOTH;
p.add (new JLabel("Input Function"),gridBagConstraintsx06);
GridBagConstraints gridBagConstraintsx07 = new GridBagConstraints();
gridBagConstraintsx07.gridx = 1;
gridBagConstraintsx07.gridy = 4;
gridBagConstraintsx07.insets = new Insets(5,5,5,5);
gridBagConstraintsx07.gridwidth = 1;
gridBagConstraintsx07.fill = GridBagConstraints.BOTH;
p.add (new JLabel("Sub-Function"),gridBagConstraintsx07);
GridBagConstraints gridBagConstraintsx08 = new GridBagConstraints();
gridBagConstraintsx08.gridx = 2;
gridBagConstraintsx08.gridy = 4;
gridBagConstraintsx08.insets = new Insets(5,5,5,5);
gridBagConstraintsx08.gridwidth = 1;
gridBagConstraintsx08.fill = GridBagConstraints.BOTH;
p.add (new JLabel("Output Function"),gridBagConstraintsx08);
GridBagConstraints gridBagConstraintsx09 = new GridBagConstraints();
gridBagConstraintsx09.gridx = 0;
gridBagConstraintsx09.gridy = 5;
gridBagConstraintsx09.insets = new Insets(5,5,5,5);
gridBagConstraintsx09.gridwidth = 1;
gridBagConstraintsx09.fill = GridBagConstraints.BOTH;
p.add (input1,gridBagConstraintsx09);
GridBagConstraints gridBagConstraintsx010 = new GridBagConstraints();
gridBagConstraintsx010.gridx = 1;
gridBagConstraintsx010.gridy = 5;
gridBagConstraintsx010.insets = new Insets(5,5,5,5);
gridBagConstraintsx010.gridwidth = 1;
gridBagConstraintsx010.fill = GridBagConstraints.BOTH;
p.add (sub1,gridBagConstraintsx010);
GridBagConstraints gridBagConstraintsx011 = new GridBagConstraints();
gridBagConstraintsx011.gridx = 2;
gridBagConstraintsx011.gridy = 5;
gridBagConstraintsx011.insets = new Insets(5,5,5,5);
gridBagConstraintsx011.gridwidth = 1;
gridBagConstraintsx011.fill = GridBagConstraints.BOTH;
p.add (output1,gridBagConstraintsx011);
GridBagConstraints gridBagConstraintsx012 = new GridBagConstraints();
gridBagConstraintsx012.gridx = 0;
gridBagConstraintsx012.gridy = 6;
gridBagConstraintsx012.insets = new Insets(5,5,5,5);
gridBagConstraintsx012.gridwidth = 1;
gridBagConstraintsx012.fill = GridBagConstraints.BOTH;
p.add (input2,gridBagConstraintsx012);
GridBagConstraints gridBagConstraintsx013 = new GridBagConstraints();
gridBagConstraintsx013.gridx = 1;
gridBagConstraintsx013.gridy = 6;
gridBagConstraintsx013.insets = new Insets(5,5,5,5);
gridBagConstraintsx013.gridwidth = 1;
```

```
        gridBagConstraintsx013.fill = GridBagConstraints.BOTH;
        p.add (sub2,gridBagConstraintsx013);
        GridBagConstraints gridBagConstraintsx014 = new GridBagConstraints();
        gridBagConstraintsx014.gridx = 2;
        gridBagConstraintsx014.gridy = 6;
        gridBagConstraintsx014.insets = new Insets(5,5,5,5);
        gridBagConstraintsx014.gridwidth = 1;
        gridBagConstraintsx014.fill = GridBagConstraints.BOTH;
        p.add (output2,gridBagConstraintsx014);
        GridBagConstraints gridBagConstraintsx015 = new GridBagConstraints();
        gridBagConstraintsx015.gridx = 0;
        gridBagConstraintsx015.gridy = 7;
        gridBagConstraintsx015.insets = new Insets(5,5,5,5);
        gridBagConstraintsx015.gridwidth = 1;
        gridBagConstraintsx015.fill = GridBagConstraints.BOTH;
        p.add (input3,gridBagConstraintsx015);
        GridBagConstraints gridBagConstraintsx016 = new GridBagConstraints();
        gridBagConstraintsx016.gridx = 1;
        gridBagConstraintsx016.gridy = 7;
        gridBagConstraintsx016.insets = new Insets(5,5,5,5);
        gridBagConstraintsx016.gridwidth = 1;
        gridBagConstraintsx016.fill = GridBagConstraints.BOTH;
        p.add (sub3,gridBagConstraintsx016);
        GridBagConstraints gridBagConstraintsx017 = new GridBagConstraints();
        gridBagConstraintsx017.gridx = 2;
        gridBagConstraintsx017.gridy = 7;
        gridBagConstraintsx017.insets = new Insets(5,5,5,5);
        gridBagConstraintsx017.gridwidth = 1;
        gridBagConstraintsx017.fill = GridBagConstraints.BOTH;
        p.add (output3,gridBagConstraintsx017);
        GridBagConstraints gridBagConstraintsx018 = new GridBagConstraints();
        gridBagConstraintsx018.gridx = 0;
        gridBagConstraintsx018.gridy = 8;
        gridBagConstraintsx018.insets = new Insets(5,5,5,5);
        gridBagConstraintsx018.gridwidth = 1;
        gridBagConstraintsx018.fill = GridBagConstraints.BOTH;
        p.add (delet,gridBagConstraintsx018);
        GridBagConstraints gridBagConstraintsx019 = new GridBagConstraints();
        gridBagConstraintsx019.gridx = 1;
        gridBagConstraintsx019.gridy = 8;
        gridBagConstraintsx019.insets = new Insets(5,5,5,5);
        gridBagConstraintsx019.gridwidth = 1;
        gridBagConstraintsx019.fill = GridBagConstraints.BOTH;
        p.add (updat,gridBagConstraintsx019);
        GridBagConstraints gridBagConstraintsx021 = new GridBagConstraints();
        gridBagConstraintsx021.gridx = 2;
        gridBagConstraintsx021.gridy = 8;
        gridBagConstraintsx021.insets = new Insets(5,5,5,5);
        gridBagConstraintsx021.gridwidth = 1;
        gridBagConstraintsx021.fill = GridBagConstraints.BOTH;
        p.add (main1,gridBagConstraintsx021);
        GridBagConstraints gridBagConstraintsx020 = new GridBagConstraints();
        gridBagConstraintsx020.gridx = 0;
        gridBagConstraintsx020.gridy = 9;
        gridBagConstraintsx020.insets = new Insets(5,5,5,5);
        gridBagConstraintsx020.gridwidth = 1;
        gridBagConstraintsx020.fill = GridBagConstraints.BOTH;
        p.add (back,gridBagConstraintsx020);
        GridBagConstraints gridBagConstraintsx022 = new GridBagConstraints();
        gridBagConstraintsx022.gridx = 2;
        gridBagConstraintsx022.gridy = 9;
        gridBagConstraintsx022.insets = new Insets(5,5,5,5);
        gridBagConstraintsx022.gridwidth = 1;
        gridBagConstraintsx022.fill = GridBagConstraints.BOTH;
        p.add (front,gridBagConstraintsx022);
        this.getContentPane().add(p);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent ae)
    {
       String str = ae.getActionCommand();
       if(str.equals("Design Repository Menu"))
       {
            this.dispose();
            main2 mm = new main2();
            mm.Gen2 ();
```

```
                }
                if(str.equals("Update Component"))
                {
                    this.dispose();
                    updateDFA mm = new updateDFA();
                    mm.init1 (key);
                }
                if(str.equals(">>"))
                {
                    key = key + 1;
                    if(key > record)
                    {
                        key = record;
                    }
                    if((key >=1)&&(key <= record))
                    {
                        this.dispose();
                        p.removeAll();
                        Gen1();
                    }
                }
                else if(str.equals("<<"))
                {
                    if(key > 1)
                    {
                        key = key - 1;
                    }
                    if(key >= 1)
                    {
                       this.dispose();
                        p.removeAll();
                        Gen1();
                    }
                }
                if(str.equals("Delete Component"))
                {
                    int response = JOptionPane.showConfirmDialog (this,"Are you sure you want to
delete the component?","Confirm Delete
Dialog",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
                    if(response == JOptionPane.YES_OPTION)
                    {
                        try
                        {
                            stmt2 = conn.createStatement ();
                            stmt2.executeUpdate("delete from DFA where number = " + key + ";");
                            stmt2.executeUpdate("update DFA set number = number - 1 where number
> " + key + ";");

                            stmt2.close();
                            key = key + 1;
                            if((key >=1)&&(key < record))
                            {
                                this.dispose();
                                p.removeAll();
                                Gen1();
                            }
                            if(key >= record)
                            {
                                front.setEnabled (false);
                                key = record;
                            }
                        }
                        catch(Exception ex)
                        {
                            System.out.println(ex);
                        }
                    }
                }
            }
        }
```

5) DFV.java

```
        //Shows the DFV matrix along with the DFV index value for each of the two designs
        import javax.swing.*;
        import java.util.*;
        import java.awt.*;
        import java.awt.image.BufferedImage;
        import java.sql.*;
        import java.net.*;
        import javax.imageio.*;
        import javax.swing.border.*;
        import java.io.*;
        import java.awt.event.*;
        public class DFV extends JFrame implements ActionListener
        {
            String url,driver,moduls;
            Connection conn;
            Statement stmt,stmt1,stmt2;
            ResultSet rs;
            JPanel p,p1;
            JTextField tf;
            int l=0,n=4;
            BufferedImage imageo;
            ImageIcon image1;
            JLabel[] img = new JLabel[100];
            JLabel[] img2 = new JLabel[100];
            JLabel[][] lbl = new JLabel[15][15];
            JPanel[][] ai = new JPanel[15][15];
            JComboBox[] wt = new JComboBox[10];
            Border lineb,linec;
            JLabel newp,dfvlbl;
            JButton main1;
            JTextField tf1;
            String[] cn1 = new String[20];
            String[] em1 = new String[20];
            String[][] chan = new String[10][10];
            int[] change = new int[20];
            int[] sum = new int[10];
            int row, col, count,total=0,ab1,ddy,num,ddx,ddx2,ddy2,num2,dfv;
            String[] modul = new String[20];
            int[][] dd = new int[10][100];
            int[][] dd2 = new int[10][100];
            String mod_tok="";
            StringTokenizer st1,st2,st3;
            public void init1(String[] mod, int ddx1,int y, String[] em,int[] wt, String[][] abb,
int bb, int[][] dd1, int ddy1, int num1)
            {
                ab1 = bb;//Keeps a track of whether 1st or 2nd design alternative.
                ddy = ddy1;
                num = num1;
                dd = dd1;
                modul = mod;
                col = y;
                em1 = em;
                change = wt;
                chan = abb;
                ddx = ddx1;
                for(int i =1; i<=ddx; i++) //To find out number of modules;
                {
                   //System.out.println(modul[i]);
                     if(!modul[i].equals(""))
                     {
                         count++;
                         mod_tok = mod_tok + "$" + modul[i];
                     }
                }
                System.out.println ("Count" + count);
                p1 = new JPanel(new GridBagLayout());
                this.setSize(700,700);
                //setResizable(false);
                p = new JPanel();
                p.setLayout(new GridLayout(col+2,count+1));
                p.setForeground(Color.BLUE);
                this.setTitle("DFV Matrix and DFV Index");
                main1 = new JButton("Design Repository Menu");
                main1.addActionListener(this);
```

```
lineb = new LineBorder(Color.gray);
linec = new LineBorder(Color.BLACK);
dfvlbl=new JLabel("Total DFV Index: ");
tf1=new JTextField();
tf1.setEditable (false);
for(int i=1; i<=count; i++)
{
    img[i] = new JLabel(modul[i]);
    img[i].setFont(new Font("System",Font.BOLD,10));
}
for(int j=1; j<=col; j++)
{
    img2[j] = new JLabel(em[j]);
    img2[j].setFont(new Font("System",Font.BOLD,10));
}
try
{
    ai[1][1] = new JPanel();
    //ai[1][1].setBorder (lineb);
    p.add(ai[1][1]);
    for(int m=2;m<=count+1;m++)
    {
        ai[1][m] = new JPanel();
        //ai[1][m].setBorder (lineb);
        ai[1][m].add(img[m-1]);
        p.add (ai[1][m]);
    }
    for(int i = 2; i<=col+1; i++)
    {
        for(int j=1;j<=count+1;j++)
        {
            if(j==1)
            {
                ai[i][1] = new JPanel();
                //ai[i][1].setBorder (lineb);
                ai[i][1].add(img2[i-1]);
                p.add (ai[i][1]);
            }
            else
            {
                ai[i][j] = new JPanel();
                ai[i][j].setBorder(lineb);
                lbl[i][j] = new JLabel();
                if(!chan[i-1][j-1].equals ("0"))
                lbl[i][j].setText (chan[i-1][j-1]);
                else
                lbl[i][j].setText (" ");
                ai[i][j].add (lbl[i][j]);
                ai[i][j].setForeground(Color.WHITE);
                p.add(ai[i][j]);
            }
        }
    }
    ai[col+2][1] = new JPanel();
    //ai[col+2][1].setBorder (lineb);
    lbl[col+2][1] = new JLabel("SUM");
    ai[col+2][1].add (lbl[col+2][1]);
    p.add (ai[col+2][1]);
    for(int i = 1; i<=count; i++)
    {
        for(int j=1;j<=col;j++)
        {
                sum[i] = sum[i] + Integer.parseInt (chan[j][i]);
        }
        total = total+sum[i];
        ai[col+2][1+i] = new JPanel();
        //ai[col+2][1+i].setBorder (lineb);
        lbl[col+2][1+i] = new JLabel(Integer.toString (sum[i]));
        ai[col+2][1+i].add (lbl[col+2][1+i]);
        p.add (ai[col+2][1+i]);
    }
    tf1.setText (Integer.toString (total));
    newp = new JLabel("DFV Matrix and DFV Index");
    newp.setFont(new Font("System",Font.BOLD,25));
    GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
    gridBagConstraintsx00.gridx = 0;
```

```
            gridBagConstraintsx00.gridy = 0;
            gridBagConstraintsx00.gridwidth = 2;
            gridBagConstraintsx00.insets = new Insets(5,5,5,5);
            p1.add(newp,gridBagConstraintsx00);
            GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
            gridBagConstraintsx01.gridx = 0;
            gridBagConstraintsx01.gridy = 1;
            gridBagConstraintsx01.gridwidth = 2;
            gridBagConstraintsx01.insets = new Insets(5,5,5,5);
            p1.add(p, gridBagConstraintsx01);
            GridBagConstraints gridBagConstraintsx02a = new GridBagConstraints();
            gridBagConstraintsx02a.gridx = 0;
            gridBagConstraintsx02a.gridy = 2;
            gridBagConstraintsx02a.insets = new Insets(5,5,5,5);
            gridBagConstraintsx02a.gridwidth = 1;
            gridBagConstraintsx02a.fill = GridBagConstraints.BOTH;
            p1.add(dfvlbl, gridBagConstraintsx02a);
            GridBagConstraints gridBagConstraintsx02b = new GridBagConstraints();
            gridBagConstraintsx02b.gridx = 1;
            gridBagConstraintsx02b.gridy = 2;
            gridBagConstraintsx02b.insets = new Insets(5,5,5,5);
            gridBagConstraintsx02b.gridwidth = 1;
            gridBagConstraintsx02b.fill = GridBagConstraints.BOTH;
            p1.add(tf1, gridBagConstraintsx02b);
            GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
            gridBagConstraintsx02.gridx = 0;
            gridBagConstraintsx02.gridy = 3;
            gridBagConstraintsx02.insets = new Insets(5,5,5,5);
            gridBagConstraintsx02.gridwidth = 2;
            gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
            p1.add(main1, gridBagConstraintsx02);
            /*GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
            gridBagConstraintsx03.gridx = 0;
            gridBagConstraintsx03.gridy = 3;
            gridBagConstraintsx03.insets = new Insets(5,5,5,5);
            gridBagConstraintsx03.gridwidth = 1;
            gridBagConstraintsx03.fill = GridBagConstraints.BOTH;
            p1.add(new JButton("Main Menu"), gridBagConstraintsx03);*/
            this.getContentPane().add(p1);
            setVisible(true);
            System.out.println ("DFV: " + num + ",total: " + total + ", ab1: " + ab1);
            calcfunc();
        }
        catch (Exception ex)
        {
            System.out.println(ex);
        }
    }
    public void actionPerformed(ActionEvent ae)
    {
        String str = ae.getActionCommand();
        if(str.equals("Design Repository Menu"))
        {
            this.dispose();
            main1 mm = new main1();
            mm.Gen1 ();
        }
    }
    public void calcfunc()
    {
        String abc = "",abc2="",tem,tem1,abc3="";
        int u1=0,v1=0,w1=0;
        String[] modul_t = new String[20];
        try
        {
            for(int i=1;i<=ddx;i++)
            {
                abc = abc + "$";
                for(int j=1;j<=ddy;j++)
                {
                    if(dd[i][j]!=0)
                        abc = abc + "&" + dd[i][j];
                }
            }
            System.out.println(abc);
            driver = "com.mysql.jdbc.Driver";
```

```
                    url = "jdbc:mysql://localhost:3306/test";
                    Class.forName(driver);
                    conn =DriverManager.getConnection(url,"root","pennstate");
                    stmt = conn.createStatement();
                    System.out.println("DFV AB1: " + ab1);
                    if(ab1==1) //corresponds to the first design alternative.
                    {
                        stmt.executeUpdate ("drop table if exists fin_des;");
                        stmt.executeUpdate ("create table fin_des(ddx int,ddy int,dd 
varchar(50),modul varchar(100),num int,dfv int);");
                        stmt.executeUpdate("insert into fin_des(ddx,ddy,dd,modul,num,dfv) 
values(" + ddx + "," + ddy + ",\'" + abc + "\',\'" + mod_tok + "\'," + num + "," + total + ");");
                        NewEmpty aa = new NewEmpty();
                        aa.new2(2);
                    }
                    else if(ab1==2)
                    {
                        rs = stmt.executeQuery("Select * from fin_des;");
                        while(rs.next ())
                        {
                            abc2 = rs.getString ("dd");
                            ddx2 = rs.getInt ("ddx");
                            ddy2 = rs.getInt ("ddy");
                            num2 = rs.getInt ("num");
                            abc3 = rs.getString ("modul");
                            dfv = rs.getInt("dfv");
                        }
                        if(dfv < total)
                        {
                            st1 = new StringTokenizer(abc2,"$");
                            while(st1.hasMoreTokens())
                            {
                                u1++;
                                tem = st1.nextToken();
                                st2 = new StringTokenizer(tem,"&");
                                v1=0;
                                while(st2.hasMoreTokens ())
                                {
                                    v1++;
                                    tem1 = st2.nextToken();
                                    dd2[u1][v1] = Integer.parseInt (tem1);
                                }
                            }
                            st3 = new StringTokenizer(abc3,"$");
                            while(st3.hasMoreTokens ())
                            {
                                w1++;
                                modul_t[w1] = st3.nextToken ();
                            }
                            for(int i=1;i<=u1;i++)
                            {
                                for(int j=1;j<=v1;j++)
                                {
                                    System.out.print(dd2[i][j] + " ");
                                }
                                System.out.print("\n");
                            }
                            bestd bd = new bestd();
                            bd.init1 (num2, dd2, ddx2, ddy2, modul_t);
                        }
                        else if(dfv > total)
                        {
                            bestd bd = new bestd();
                            bd.init1 (num, dd, ddx, ddy, modul);
                        }
                    }
                }
                catch(Exception ex)
                {
                    System.out.println(ex.getMessage());
                }
            }
             /*public static void main(String[] args)
            {
                QFD2 qq = new QFD2();
                qq.init1();
```

```
    }*/
}
```

6) inprules.java

```
//Inputs the functions from the user and returns all the possible combinations
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.*;
import java.io.*;
import java.util.*;

public class inprules extends JFrame implements ActionListener
{
        boolean flag1 = true,fl=false;
         JTextField baseid,rec;
         JButton savrul,loadrul;
        JComboBox[] rulein = new JComboBox[1000];
         JComboBox[] rulesub = new JComboBox[1000];
         JComboBox[] ruleout = new JComboBox[1000];
         GridBagConstraints[] bag = new GridBagConstraints[1000];
         GridBagConstraints[] bag1 = new GridBagConstraints[1000];
         GridBagConstraints[] bag2 = new GridBagConstraints[1000];
        JLabel inpu,subf,outpu,baseid1,and1,and2,title;
         JPanel[] jp = new JPanel[15];
         JPanel p;
         JTabbedPane tp;
        JButton inser,main1;
         String
inp1,su1,oup1,inp2,su2,oup2,inp3,su3,oup3,tfin,tfsub,tfout,tfin2,tfsub2,tfout2,tfin3,tfsub3,tfout
3,tfcomp,dfa1;
        int rul=0,record=0,i=1,j;
         int[] count1 = new int[100];
         int jpo,yo=0;
         int[] el = new int[200];
         float[] aa1 = new float[300];
         float suma;
         int dummyi,dummyj,dummyk;
         boolean flag2 = true, flag3 = true;
         String a1 = "rule1",a2,aa2;
         String a1o = "rule1";
        Connection conn;
        Statement stmt,stmt1,stmt2,stmt3;
        ResultSet rset,rs,rse1;
        String a, driver,url;
         String[] u10 = new String[15];
        FileDialog dialog1;
         int i22=0,i33=0;
         public void init1()
        {
                p = new JPanel();
                 setSize(700,700);
                 p.setLayout(new GridBagLayout());
                 setResizable(false);
                 this.setTitle("Input EMS Diagram");
                 try
                {
                        driver = "com.mysql.jdbc.Driver";
                        url = "jdbc:mysql://localhost:3306/test";
                        Class.forName(driver);
                        conn =DriverManager.getConnection(url,"root","pennstate");
                        stmt = conn.createStatement();
                          stmt1 = conn.createStatement ();
                          stmt2 = conn.createStatement ();
                          stmt3 = conn.createStatement ();
                        rset = stmt.executeQuery("Select * from DFA");
                        while (rset.next())
                        {
                                record++;
                        }
                }
                catch(Exception ex)
                {
                        System.out.println(ex);
```

```
                               return;
                       }
                       title = new JLabel("Please Enter All Nodes of the EMS Diagram");
                        title.setFont(new Font("System",Font.BOLD,25));
                        inser = new JButton("Query the Design Repository");
                       inser.addActionListener(this);
                       main1 = new JButton("Design Repository Menu");
                       main1.addActionListener(this);
                        savrul = new JButton("Save This Query");
                        savrul.addActionListener (this);
                        loadrul = new JButton("Load Saved Query");
                        loadrul.addActionListener (this);
                        tp = new JTabbedPane();
                        for(j=1;j<=5;j++)
                        {
                            jp[j] = new JPanel();
                            inpu = new JLabel("Input Flow");
                            subf = new JLabel("Sub-Function");
                            outpu = new JLabel("Output Flow");
                            jp[j].setLayout(new GridBagLayout());
                            GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
                            gridBagConstraintsx00.gridx = 0;
                            gridBagConstraintsx00.gridy = 0;
                            gridBagConstraintsx00.insets = new Insets(5,5,5,5);
                            p.add(title, gridBagConstraintsx00);
                            GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
                            gridBagConstraintsx01.gridx = 0;
                            gridBagConstraintsx01.gridy = 0;
                            gridBagConstraintsx01.insets = new Insets(5,5,5,5);
                            jp[j].add(inpu, gridBagConstraintsx01);
                            GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
                            gridBagConstraintsx02.gridx = 1;
                            gridBagConstraintsx02.gridy = 0;
                            gridBagConstraintsx02.insets = new Insets(5,5,5,5);
                            gridBagConstraintsx02.gridwidth = 1;
                            gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
                            jp[j].add(subf, gridBagConstraintsx02);
                            GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
                            gridBagConstraintsx03.gridx = 2;
                            gridBagConstraintsx03.gridy = 0;
                            gridBagConstraintsx03.insets = new Insets(5,5,5,5);
                            jp[j].add(outpu, gridBagConstraintsx03);
                            rec = new JTextField(15);
                            rec.setText("Component(s) in Repository: " +
Integer.toString(record));
                            rec.setEditable(false);
                            for(int k=((10*j)-9);k<=(10*j);k++)
                            {
                                rulein[k] =  new JComboBox();
                                rulein[k].addItem("");
                                rulein[k].addItem("Acoustic Energy");
                                rulein[k].addItem("Analog Control");
                                rulein[k].addItem("Auditory");
                                rulein[k].addItem("Biological Energy");
                                rulein[k].addItem("Chemical Energy");
                                rulein[k].addItem("Colloidal");
                                rulein[k].addItem("Composite");
                                rulein[k].addItem("Discrete Control");
                                rulein[k].addItem("Electrical Energy");
                                rulein[k].addItem("Electromagnetic Energy");
                                rulein[k].addItem("Gas");
                                rulein[k].addItem("Gas-Gas Mixture");
                                rulein[k].addItem("Human Energy");
                                rulein[k].addItem("Human Material");
                                rulein[k].addItem("Hydraulic Energy");
                                rulein[k].addItem("Liquid");
                                rulein[k].addItem("Liquid-Gas");
                                rulein[k].addItem("Liquid-Liquid");
                                rulein[k].addItem("Magnetic Energy");
                                rulein[k].addItem("Mechanical Energy");
                                rulein[k].addItem("Object");
                                rulein[k].addItem("Olfactory");
                                rulein[k].addItem("Optical Energy");
                                rulein[k].addItem("Particulate");
                                rulein[k].addItem("Plasma");
                                rulein[k].addItem("Pneumatic Energy");
```

```
 rulein[k].addItem("Radioactive/Nuclear");
 rulein[k].addItem("Rotational Energy");
 rulein[k].addItem("Solar Energy");
 rulein[k].addItem("Solid");
 rulein[k].addItem("Solid-Gas");
 rulein[k].addItem("Solid-Liquid");
 rulein[k].addItem("Solid-Liquid-Gas");
 rulein[k].addItem("Solid-Solid");
 rulein[k].addItem("Thermal Energy");
 rulein[k].addItem("Translational Energy");
 rulein[k].addItem("Tactile");
 rulein[k].addItem("Taste");
 rulein[k].addItem("Visual");
 //rulein[k].addFocusListener (this);

 ruleout[k] = new JComboBox();
 ruleout[k].addItem("");
 ruleout[k].addItem("Acoustic Energy");
 ruleout[k].addItem("Analog Control");
 ruleout[k].addItem("Auditory");
 ruleout[k].addItem("Biological Energy");
 ruleout[k].addItem("Chemical Energy");
 ruleout[k].addItem("Colloidal");
 ruleout[k].addItem("Composite");
 ruleout[k].addItem("Discrete Control");
 ruleout[k].addItem("Electrical Energy");
 ruleout[k].addItem("Electromagnetic Energy");
 ruleout[k].addItem("Gas");
 ruleout[k].addItem("Gas-Gas Mixture");
 ruleout[k].addItem("Human Energy");
 ruleout[k].addItem("Human Material");
 ruleout[k].addItem("Hydraulic Energy");
 ruleout[k].addItem("Liquid");
 ruleout[k].addItem("Liquid-Gas");
 ruleout[k].addItem("Liquid-Liquid");
 ruleout[k].addItem("Magnetic Energy");
 ruleout[k].addItem("Mechanical Energy");
 ruleout[k].addItem("Object");
 ruleout[k].addItem("Olfactory");
 ruleout[k].addItem("Optical Energy");
 ruleout[k].addItem("Particulate");
 ruleout[k].addItem("Plasma");
 ruleout[k].addItem("Pneumatic Energy");
 ruleout[k].addItem("Radioactive/Nuclear");
 ruleout[k].addItem("Rotational Energy");
 ruleout[k].addItem("Solar Energy");
 ruleout[k].addItem("Solid");
 ruleout[k].addItem("Solid-Gas");
 ruleout[k].addItem("Solid-Liquid");
 ruleout[k].addItem("Solid-Liquid-Gas");
 ruleout[k].addItem("Solid-Solid");
 ruleout[k].addItem("Thermal Energy");
 ruleout[k].addItem("Translational Energy");
 ruleout[k].addItem("Tactile");
 ruleout[k].addItem("Taste");
 ruleout[k].addItem("Visual");
 ruleout[k].addActionListener (this);
 //rulein[k].addFocusListener (this);

rulesub[k] = new JComboBox();
 rulesub[k].addItem("");
 rulesub[k].addItem("Actuate");
 rulesub[k].addItem("Allow DOF");
 rulesub[k].addItem("Branch");
 rulesub[k].addItem("Change");
 rulesub[k].addItem("Channel");
 rulesub[k].addItem("Collect");
 rulesub[k].addItem("Condition");
 rulesub[k].addItem("Connect");
 rulesub[k].addItem("Contain");
 rulesub[k].addItem("Control Magnitude");
 rulesub[k].addItem("Convert");
 rulesub[k].addItem("Couple");
 rulesub[k].addItem("Decrease");
 rulesub[k].addItem("Decrement");
 rulesub[k].addItem("Detect");
```

```
        rulesub[k].addItem("Display");
        rulesub[k].addItem("Distribute");
        rulesub[k].addItem("Divide");
        rulesub[k].addItem("Export");
        rulesub[k].addItem("Extract");
        rulesub[k].addItem("Guide");
        rulesub[k].addItem("Import");
        rulesub[k].addItem("Increase");
        rulesub[k].addItem("Increment");
        rulesub[k].addItem("Indicate");
        rulesub[k].addItem("Inhibit");
        rulesub[k].addItem("Join");
        rulesub[k].addItem("Link");
        rulesub[k].addItem("Measure");
        rulesub[k].addItem("Mix");
        rulesub[k].addItem("Position");
        rulesub[k].addItem("Prevent");
        rulesub[k].addItem("Process");
        rulesub[k].addItem("Provision");
        rulesub[k].addItem("Regulate");
        rulesub[k].addItem("Remove");
        rulesub[k].addItem("Rotate");
        rulesub[k].addItem("Secure");
        rulesub[k].addItem("Sense");
        rulesub[k].addItem("Seperate");
        rulesub[k].addItem("Shape");
        rulesub[k].addItem("Signal");
        rulesub[k].addItem("Stabilize");
        rulesub[k].addItem("Stop");
        rulesub[k].addItem("Store");
        rulesub[k].addItem("Supply");
        rulesub[k].addItem("Support");
        rulesub[k].addItem("Track");
        rulesub[k].addItem("Transfer");
        rulesub[k].addItem("Translate");
        rulesub[k].addItem("Transmit");
        rulesub[k].addItem("Transport");
        rulesub[k].addActionListener(this);
        //rulein[k].addFocusListener (this);

      bag[k] = new GridBagConstraints();
      bag[k].gridx = 0;
      bag[k].gridy = i;
      bag[k].insets = new Insets(5,5,5,5);
      bag[k].gridwidth = 1;
      bag[k].fill = GridBagConstraints.BOTH;
      jp[j].add(rulein[k], bag[k]);
      bag1[k] = new GridBagConstraints();
      bag1[k].gridx = 1;
      bag1[k].gridy = i;
      bag1[k].insets = new Insets(5,5,5,5);
      bag1[k].gridwidth = 1;
      bag1[k].fill = GridBagConstraints.BOTH;
      jp[j].add(rulesub[k],bag1[k]);
      bag2[k] = new GridBagConstraints();
      bag2[k].gridx = 2;
      bag2[k].gridy = i;
      bag2[k].insets = new Insets(5,5,5,5);
      jp[j].add(ruleout[k], bag2[k]);
        i++;
        if(i==11) i=1;
    }
    tp.addTab("AA"+j,jp[j]);
}
GridBagConstraints gridBagConstraintsx04a = new GridBagConstraints();
gridBagConstraintsx04a.gridx = 0;
gridBagConstraintsx04a.gridy = 1;
gridBagConstraintsx04a.insets = new Insets(5,5,5,5);
p.add(tp, gridBagConstraintsx04a);
GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
gridBagConstraintsx04.gridx = 0;
gridBagConstraintsx04.gridy = 2;
gridBagConstraintsx04.insets = new Insets(5,5,5,5);
p.add(inser, gridBagConstraintsx04);
GridBagConstraints gridBagConstraintsx05a = new GridBagConstraints();
gridBagConstraintsx05a.gridx = 0;
```

```
                       gridBagConstraintsx05a.gridy = 3;
                       gridBagConstraintsx05a.insets = new Insets(5,5,5,5);
                       p.add(savrul, gridBagConstraintsx05a);
                       GridBagConstraints gridBagConstraintsx05b = new GridBagConstraints();
                       gridBagConstraintsx05b.gridx = 0;
                       gridBagConstraintsx05b.gridy = 4;
                       gridBagConstraintsx05b.insets = new Insets(5,5,5,5);
                       p.add(loadrul, gridBagConstraintsx05b);
                       GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
                       gridBagConstraintsx05.gridx = 0;
                       gridBagConstraintsx05.gridy = 5;
                       gridBagConstraintsx05.insets = new Insets(5,5,5,5);
                       p.add(main1, gridBagConstraintsx05);
                       GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
                       gridBagConstraintsx06.gridx = 0;
                       gridBagConstraintsx06.gridy = 6;
                       gridBagConstraintsx06.insets = new Insets(5,5,5,5);
                       gridBagConstraintsx06.gridwidth = 2;
                       gridBagConstraintsx06.fill = GridBagConstraints.BOTH;
                       p.add(rec, gridBagConstraintsx06);
                       this.getContentPane().add(p);
                       setVisible(true);
               }
              /* public void focusGained(FocusEvent fe)
               {
                   for(int jjj = 1;jjj<=50;jjj++)
                   {
                       if(rulesub[jjj] == fe.getComponent ())
                       {
                           System.out.println("Focus Gained: " + rulesub[jjj]);
                       }
                   }
               }
               public void focusLost(FocusEvent fe)
               {
                   //System.out.println("Focus Gained: " + fe.getComponent ());
               }*/
               public void actionPerformed(ActionEvent ae)
               {
                      boolean flg = false,flg2 = false;
                       FileDialog fld;
                       String bb = "",abc="";
                       int det = 0;
                       for(int i = 1;i<=50; i++)
                       {
                           if(ae.getSource() == rulesub[i])
                           {
                               det=i;
                               flg = true;
                               break;
                           }
                           else if(ae.getSource () == ruleout[i])
                           {
                               det = i;
                               fl=false;
                               flg = true;
                               if(fl == false)
                               flg2 = true;
                               break;
                           }
                       }
                       if(flg == true)
                       {
                           if(flg2 == true)
                           {
                               bb = (String)ruleout[det].getSelectedItem ();
                               rulein[det+1].setSelectedItem (bb);
                           }
                           else
                           {
                               bb = (String)rulesub[det].getSelectedItem ();
                               if((bb.equals ("Actuate"))||(bb.equals ("Store"))||(bb.equals
("Supply"))||(bb.equals ("Transfer"))||(bb.equals ("Import"))||(bb.equals ("Export"))||(bb.equals
("Couple"))||(bb.equals ("Seperate")))
                               {
                                   int u = rulein[det].getSelectedIndex();
```

```
                                    ruleout[det].setSelectedIndex(u);
                                }
                            }
                        }
                        else if(flg == false)
                        {
                            String str = ae.getActionCommand();
                            if(str.equals ("Save This Query"))
                            {
                                try
                                {
                                    fld = new FileDialog(this, "Save File As", FileDialog.SAVE);
                                    fld.setVisible(true);
                                    if(fld.getFile () != null)
                                    {
                                        abc = fld.getDirectory()+File.separator +
fld.getFile()+".txt";
                                    }
                                    else
                                        abc = fld.getDirectory() + File.separator + "aa.txt";
                                    BufferedWriter out = new BufferedWriter(new FileWriter(abc));
                                    for(int ui = 1;ui<=50;ui++)
                                    {
                                        if((!(rulein[ui].getSelectedItem
()).equals(""))||(!(rulesub[ui].getSelectedItem ()).equals(""))||(!(ruleout[ui].getSelectedItem
()).equals("")))
                                        {
                                            out.write ("$"+ rulein[ui].getSelectedItem() + "$" +
rulesub[ui].getSelectedItem () + "$" + ruleout[ui].getSelectedItem ()+"*");
                                        }
                                    }
                                    out.close ();
                                }
                                catch(IOException ex)
                                {
                                    System.out.println(ex);
                                }
                            }
                            if(str.equals ("Load Saved Query"))
                            {
                                String sou="";
                                int ju = 1,iu=1;
                                fl = true;
                                try
                                {
                                    fld = new FileDialog(this, "Open Query File",
FileDialog.LOAD);

                                    fld.setVisible(true);
                                    if(fld.getFile () != null)
                                    {
                                        abc = fld.getDirectory()+File.separator + fld.getFile();
                                    }
                                    //else
                                    //    abc = fld.getDirectory() + File.separator + "aa.txt";
                                    BufferedReader in = new BufferedReader(new FileReader(abc));
                                    //while(in.readLine () != null)
                                    //{
                                        sou = in.readLine ();
                                    //}
                                    StringTokenizer st1 = new StringTokenizer(sou,"*");
                                    while (st1.hasMoreTokens ())
                                    {
                                        String abc1 = st1.nextToken ();
                                        StringTokenizer st2 = new StringTokenizer(abc1,"$");
                                        ju=1;
                                        while(st2.hasMoreTokens())
                                        {
                                            String abc2 = st2.nextToken();
                                            if(ju==1)
                                            rulein[iu].setSelectedItem (abc2);
                                            if(ju==2)
                                            rulesub[iu].setSelectedItem (abc2);
                                            if(ju==3)
                                            {
                                                ruleout[iu].setSelectedItem (abc2);
                                                break;
```

```
                                           }
                                           ju++;
                                       }
                                       iu++;
                                   }
                               }
                               catch(IOException ex)
                               {
                                   System.out.println(ex);
                               }
                           }
                           if(str.equals("Query the Design Repository"))
                           {
                               try
                               {
                                   rset = stmt.executeQuery("Select * from rules;");
                                   while (rset.next())
                                   {
                                       i33 = 1;
                                       inp1 = rset.getString("input1");
                                       oup1 = rset.getString ("output1");
                                       su1 = rset.getString ("subfunction");
                                       inp2 = rset.getString("input2");
                                       oup2 = rset.getString ("output2");
                                       su2 = rset.getString ("subfunction2");
                                       inp3 = rset.getString("input3");
                                       oup3 = rset.getString ("output3");
                                       su3 = rset.getString ("subfunction3");
                                       tfcomp = rset.getString ("base_id");
                                       if(((!inp1.equals
(""))&&(!oup1.equals(""))&&(!su1.equals("")))&&((!inp2.equals
(""))&&(!oup2.equals(""))&&(!su2.equals("")))&&((!inp3.equals
(""))&&(!oup3.equals(""))&&(!su3.equals(""))))
                                       {
                                           for(int u1=1;u1<=48;u1++)
                                           {
                                               tfin = (String)rulein[u1].getSelectedItem();
                                               tfsub = (String)rulesub[u1].getSelectedItem();
                                               tfout = (String)ruleout[u1].getSelectedItem();
                                               tfin2 = (String)rulein[u1+1].getSelectedItem();
                                               tfsub2 = (String)rulesub[u1+1].getSelectedItem();
                                               tfout2 = (String)ruleout[u1+1].getSelectedItem();
                                               tfin3 = (String)rulein[u1+2].getSelectedItem();
                                               tfsub3 = (String)rulesub[u1+2].getSelectedItem();
                                               tfout3 = (String)ruleout[u1+2].getSelectedItem();
                                               if(((!tfin.equals(""))&&(!tfsub.equals
(""))&&(!tfout.equals ("")))&&((!tfin2.equals(""))&&(!tfsub2.equals (""))&&(!tfout2.equals
("")))&&((!tfin3.equals(""))&&(!tfsub3.equals (""))&&(!tfout3.equals (""))))
                                               {
                                                   if(((inp1.equals(tfin))&&(oup1.equals
(tfout))&&(su1.equals(tfsub)))&&((inp2.equals(tfin2))&&(oup2.equals
(tfout2))&&(su2.equals(tfsub2)))&&((inp3.equals(tfin3))&&(oup3.equals
(tfout3))&&(su3.equals(tfsub3))))
                                                   {
                                                       i22++;
                                                       if(flag1 == true)
                                                       {
                                                           flag1=false;
                                                           stmt1.executeUpdate ("drop table if
exists rule" + i22+";");
                                                           stmt1.executeUpdate ("create table
rule" + i22 + " (no int,compo_id varchar(50),dfa_index float);");
                                                       }
                                                       rs = stmt2.executeQuery ("select * from
DFA where baseid = \'" + tfcomp + "\';");
                                                       while(rs.next ())
                                                       {
                                                           i33++;
                                                           stmt1.executeUpdate("insert into
rule" + i22 + "(no,compo_id, dfa_index) values("+ (i33-1) + ",\'" + rs.getString ("compid") +
"\'," + Float.parseFloat(rs.getString ("dfa"))+");");
                                                       }
                                                       count1[i22] = i33-1;
                                                       rs.close();
                                                     }
                                                   }
```

```
                                                }
                                            }
                                            else if(((!inp1.equals
(""))&&(!oup1.equals(""))&&(!su1.equals("")))&&((!inp2.equals
(""))&&(!oup2.equals(""))&&(!su2.equals(""))))
                                            {
                                                for(int u1=1;u1<=49;u1++)
                                                {
                                                      tfin =
(String)rulein[u1].getSelectedItem();
                                                      tfsub =
(String)rulesub[u1].getSelectedItem();
                                                      tfout =
(String)ruleout[u1].getSelectedItem();
                                                      tfin2 =
(String)rulein[u1+1].getSelectedItem();
                                                      tfsub2 =
(String)rulesub[u1+1].getSelectedItem();
                                                      tfout2 =
(String)ruleout[u1+1].getSelectedItem();
                                                      if(((!tfin.equals(""))&&(!tfsub.equals
(""))&&(!tfout.equals ("")))&&((!tfin2.equals(""))&&(!tfsub2.equals (""))&&(!tfout2.equals
(""))))
                                                      {
                                                          if(((inp1.equals(tfin))&&(oup1.equals
(tfout))&&(su1.equals(tfsub)))&&((inp2.equals(tfin2))&&(oup2.equals
(tfout2))&&(su2.equals(tfsub2))))
                                                          {
                                                            i22++;
                                                              if(flag1 == true)
                                                              {
                                                                flag1=false;
                                                                stmt1.executeUpdate ("drop table
if exists rule" + i22+";");
                                                                stmt1.executeUpdate ("create
table rule" + i22 + " (no int,compo_id varchar(50),dfa_index float);");
                                                              }
                                                              rs = stmt2.executeQuery ("select *
from DFA where baseid = \'" + tfcomp + "\';");
                                                              while(rs.next ())
                                                              {
                                                                i33++;
                                                                stmt1.executeUpdate("insert into
rule" + i22 + "(no,compo_id, dfa_index) values("+ (i33-1) + ",\'" + rs.getString ("compid") +
"\'," + Float.parseFloat(rs.getString ("dfa"))+");");
                                                              }
                                                              count1[i22] = i33-1;
                                                              rs.close();
                                                            }
                                                        }
                                                    }
                                                }
                                             else if(((!inp1.equals
(""))&&(!oup1.equals(""))&&(!su1.equals(""))))
                                             {
                                                 for(int u1=1;u1<=50;u1++)
                                                  {
                                                      tfin =
(String)rulein[u1].getSelectedItem();
                                                      tfsub =
(String)rulesub[u1].getSelectedItem();
                                                      tfout =
(String)ruleout[u1].getSelectedItem();
                                                      if(((!tfin.equals(""))&&(!tfsub.equals
(""))&&(!tfout.equals (""))))
                                                      {
                                                          if(((inp1.equals(tfin))&&(oup1.equals
(tfout))&&(su1.equals(tfsub))))
                                                          {
                                                             i22++;
                                                              if(flag1 == true)
                                                              {
                                                                flag1=false;
                                                                stmt1.executeUpdate ("drop table
if exists rule" + i22+";");
```

```
                                                                stmt1.executeUpdate ("create
table rule" + i22 + " (no int,compo_id varchar(50),dfa_index float);");
                                                             }
                                                             rs = stmt2.executeQuery ("select *
from DFA where baseid = \'" + tfcomp + "\';");

                                                             while(rs.next ())
                                                             {
                                                                i33++;
                                                                 stmt1.executeUpdate("insert into
rule" + i22 + "(no,compo_id, dfa_index) values("+ (i33-1) + ",\'" + rs.getString ("compid") +
"\'," + Float.parseFloat(rs.getString ("dfa"))+");");
                                                             }
                                                             count1[i22] = i33-1;
                                                             rs.close();
                                                          }
                                                      }
                                                  }
                                              }
                                              flag1 = true;
                                            }
                                            stmt1.executeUpdate ("drop table if exists
final_info1;");

                                            stmt1.executeUpdate("create table final_info1 (num
int,compon_id varchar(350),dfa_index_sum float);");
                                            for(dummyi=2; dummyi<=i22; dummyi++)
                                            {
                                                a1 = a1 + ",rule" + dummyi;
                                            }
                                            for(int ijk =0;ijk<20; ijk++) el[ijk] = 1;
                                            int uk=1;
                                            while(true)
                                            {
                                                a2 = "Select * from " + a1 + " where ";
                                                for(dummyi = 1;dummyi <i22;dummyi++)
                                                {
                                                    a2 = a2 + " rule" + dummyi + ".no = " +
el[dummyi] + " and";
                                                }
                                                for (dummyj = 1; dummyj<=count1[i22];dummyj++)
                                                {
                                                    rse1 = stmt3.executeQuery(a2 + " rule" + i22
+ ".no = " + dummyj + ";");

                                                    while(rse1.next ())
                                                    {
                                                        suma = 0;
                                                        aa2 = "";
                                                        for(int jdk1 = 1; jdk1<= i22;jdk1++)
                                                        {
                                                            aa1[jdk1] = Float.parseFloat
(rse1.getString ("rule" + jdk1 + ".dfa_index"));

                                                            suma = suma + aa1[jdk1];
                                                            aa2 = aa2 + "$" + rse1.getString
("rule" + jdk1 + ".compo_id");

                                                        }
                                                    }
                                                    stmt1.executeUpdate("insert into
final_info1(num,compon_id, dfa_index_sum) values("+ uk + ",\'"+ aa2 + "\'," + suma + ");");
                                                    uk++;
                                                }
                                                flag2 = true;
                                                for(dummyk = 1; dummyk<i22;dummyk++)
                                                {
                                                    if((el[dummyk] ==
count1[dummyk])&&(flag2==true))

                                                    {
                                                        flag2 = true;
                                                    }
                                                    else
                                                        flag2 = false;
                                                }
                                                if(flag2 == true)
                                                {
                                                    break;
                                                }
                                                flag3 = true;
                                                for(jpo = i22-1;jpo>=1;jpo--)
                                                {
                                                    if(flag3 == true)
                                                    {
                                                        el[jpo] = el[jpo] + 1;
                                                        if(el[jpo]>count1[jpo])
                                                        {
                                                            el[jpo] = 1;
                                                            flag3 = true;
                                                        }
                                                        else flag3 = false;
                                                    }
                                                }
                                            }
                                            for(int jdk1 = 1;jdk1<= i22; jdk1++)
                                            {
                                                stmt1.executeUpdate ("drop table if exists rule"
+ jdk1 + ";");
                                            }
                                            int numb = 0;
                                            stmt1.executeUpdate ("drop table if exists
final_info");
                                            stmt1.executeUpdate ("create table final_info (num
int,compon_id varchar(350),dfa_index_sum float);");
                                            rs = stmt2.executeQuery ("select * from final_info1
order by dfa_index_sum ASC");
                                            while(rs.next ())
                                            {
                                                numb++;
                                                stmt1.executeUpdate("insert into
final_info(num,compon_id, dfa_index_sum) values("+ numb + ",\'"+ rs.getString ("compon_id") +
"\'," + rs.getFloat ("dfa_index_sum") + ");");
                                            }
                                            stmt.close();
                                            stmt1.close();
                                            stmt2.close();
                                            stmt3.close();
                                            conn.close();
                                            this.dispose();
                                            NewEmpty mm = new NewEmpty();
                                            mm.new2(1);
                                    }
                                    catch(Exception ex)
                                    {
                                        System.out.println(ex);
                                    }
                                }
                                else if(str.equals ("Design Repository Menu"))
                                {
                                        try
                                        {
                                            this.dispose();
                                            main1 mm = new main1();
                                            mm.Gen1 ();
                                            stmt.close();
                                            conn.close();
                                        }
                                        catch(Exception ex)
                                        {
                                            System.out.println (ex);
                                        }
                                }
                        }
                }
        }
```

7) interaction.java

```
//Inputs the Interaction Matrix to be stored in the Design Repository
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.sql.*;
import java.net.*;
import javax.imageio.*;
```

```
        import javax.swing.border.*;
        import java.io.*;
        import java.awt.event.*;
        public class interaction extends JFrame implements
MouseMotionListener,MouseListener,ActionListener
        {
            JPanel p,p1;
            JTextField tf;
            String driver,url;
            Connection conn;
            Statement stmt,stmt2;
            ResultSet rs,rs2,rs1;
            int l=0,n=4,rec=0,n1;
            BufferedImage imageo;
            ImageIcon image1;
            JButton inser,back;
            JLabel[] img = new JLabel[200];
            JLabel[] img2 = new JLabel[200];
            JLabel[][] lbl = new JLabel[20][20];
            JPanel[][] ai = new JPanel[20][20];
            int[][] a = new int[20][20];
            String[] anb = new String[200];
            Border lineb,linec;
            JLabel newp;
            DatabaseMetaData dm;
            boolean flag=false;
            public void init1()
            {
                try
                {
                    driver = "com.mysql.jdbc.Driver";
                    url = "jdbc:mysql://localhost:3306/test";
                    Class.forName(driver);
                    conn =DriverManager.getConnection(url,"root","pennstate");
                    stmt = conn.createStatement();
                    stmt2 = conn.createStatement ();
                    dm = conn.getMetaData ();
                    rs1 = dm.getTables (null,null,"interaction",null);
                    if(rs1.next ())
                        flag=true;
                    else //interaction table does not exist.
                        flag=false;
                    rs = stmt.executeQuery("Select * from rules;");
                    while(rs.next ())
                    {
                        rec++;
                        anb[rec] = rs.getString ("base_id");
                    }
                    n1 = rec + 1;
                    p1 = new JPanel(new GridBagLayout());
                    inser = new JButton("Insert into Design Repository");
                    inser.addActionListener (this);
                    back = new JButton("Design Repository Menu");
                    back.addActionListener (this);
                    setSize(700,700);
                    p = new JPanel();
                    p.setLayout(new GridLayout(n1,n1));
                    p.setForeground(Color.BLUE);
                    this.setTitle("Interaction Matrix");
                    lineb = new LineBorder(Color.gray);
                    linec = new LineBorder(Color.BLACK);
                    rs2 = stmt2.executeQuery("Select * from rules;");
                    while(rs2.next ())
                    {
                        l++;
                        img[l] = new JLabel(rs2.getString ("base_id"));
                        img[l].setFont(new Font("System",Font.BOLD,15));
                        img2[l] = new JLabel(rs2.getString ("base_id"));
                        img2[l].setFont(new Font("System",Font.BOLD,15));
                    }
                    ai[1][1] = new JPanel();
                    p.add(ai[1][1]);
                    for(int m=2;m<=n1;m++)
                    {
                        ai[1][m] = new JPanel();
                        ai[1][m].add(img[m-1]);
                        p.add (ai[1][m]);
                    }
                    for(int i = 2; i<=n1; i++)
                    {
                        for(int j=1;j<=n1;j++)
                        {
                            if(j==1)
                            {
                                ai[i][1] = new JPanel();
                                ai[i][1].add(img2[i-1]);
                                p.add (ai[i][1]);
                            }
                            else if(i==j)
                            {
                                ai[i][j] = new JPanel();
                                ai[i][j].setBorder (lineb);
                                JLabel pp = new JLabel("+");
                                pp.setFont(new Font("System",Font.BOLD,30));
                                ai[i][j].add(pp);
                                p.add (ai[i][j]);
                            }
                            else
                            {
                                ai[i][j] = new JPanel();
                                ai[i][j].setBorder (lineb);
                                lbl[i][j] = new JLabel();
                                if(flag==false)
                                lbl[i][j].setText (" ");
                                else
                                {
                                    rs = stmt.executeQuery("Select * from interaction where
baseid1 = \'" + anb[i-1] + "\' and baseid2 = \'" + anb[j-1] + "\';");
                                    while(rs.next ())
                                    {
                                        lbl[i][j] = new JLabel();
                                        if(rs.getInt ("interact")==0)
                                        {
                                                lbl[i][j].setText(" ");
                                                lbl[i][j].setFont(new
Font("System",Font.BOLD,30));
                                        }
                                        else
                                        {
                                            lbl[i][j].setText("*");
                                            lbl[i][j].setFont(new Font("System",Font.BOLD,30));
                                        }
                                    }
                                }
                                ai[i][j].add (lbl[i][j]);
                                ai[i][j].addMouseMotionListener(this);
                                ai[i][j].addMouseListener(this);
                                ai[i][j].setForeground(Color.WHITE);
                                p.add(ai[i][j]);
                            }
                        }
                    }
                    newp = new JLabel("Input the Interaction Matrix");
                    newp.setFont(new Font("System",Font.BOLD,30));
                    GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
                    gridBagConstraintsx00.gridx = 0;
                    gridBagConstraintsx00.gridy = 0;
                    gridBagConstraintsx00.gridwidth = 2;
                    gridBagConstraintsx00.insets = new Insets(5,5,5,5);
                    p1.add(newp,gridBagConstraintsx00);
                    GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
                    gridBagConstraintsx01.gridx = 0;
                    gridBagConstraintsx01.gridy = 1;
                    gridBagConstraintsx01.gridwidth = 2;
                    gridBagConstraintsx01.insets = new Insets(5,5,5,5);
                    p1.add(p, gridBagConstraintsx01);
                    GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
                    gridBagConstraintsx02.gridx = 0;
                    gridBagConstraintsx02.gridy = 2;
                    gridBagConstraintsx02.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx02.gridwidth = 1;
                    gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
```

```
        p1.add(inser, gridBagConstraintsx02);
        GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
        gridBagConstraintsx03.gridx = 1;
        gridBagConstraintsx03.gridy = 2;
        gridBagConstraintsx03.insets = new Insets(5,5,5,5);
        gridBagConstraintsx03.gridwidth = 1;
        gridBagConstraintsx03.fill = GridBagConstraints.BOTH;
        p1.add(back, gridBagConstraintsx03);
        this.getContentPane().add(p1);
        setVisible(true);
    }
    catch (Exception ex)
    {
        System.out.println(ex);
    }
}
public void mouseMoved(MouseEvent e)
{
   for(int i1=2;i1<=n1;i1++)
   {
     for(int j1=2;j1<=n1;j1++)
      {
            if(e.getComponent () == ai[i1][j1])
            {
                ai[i1][j1].setBorder(linec);
                p.revalidate();
            }
            else
            {
                ai[i1][j1].setBorder(lineb);
                p.revalidate();
            }
       }
   }
 }
public void mouseDragged(MouseEvent e)
{
   //System.out.println("Mouse dragged");
}
public void mousePressed(MouseEvent e)
{
   //System.out.println("Mouse Pressed");
}
public void mouseReleased(MouseEvent e)
{
   //System.out.println("Mouse Released");
}
public void mouseEntered(MouseEvent e)
{
   //System.out.println("Mouse Entered");
}
public void mouseExited(MouseEvent e)
{
   //System.out.println ("Mouse Exited");
}
public void mouseClicked(MouseEvent e)
{
   for(int i1=2;i1<=n1;i1++)
   {
     for(int j1=2;j1<=n1;j1++)
      {
            if(e.getComponent () == ai[i1][j1])
            {
                String abb = lbl[i1][j1].getText();
                if(abb.equals(" "))
                {
                    lbl[i1][j1].setText ("*");
                    lbl[i1][j1].setFont(new Font("System",Font.BOLD,30));
                }
                else
                    lbl[i1][j1].setText (" ");
                p.revalidate();
            }
       }
   }
}
```

```
            public void actionPerformed(ActionEvent ae)
            {
                String str = ae.getActionCommand ();
                if(str.equals ("Insert into Design Repository"))
                {
                    for(int i1=2;i1<=n1;i1++)
                    {
                        for(int j1=2;j1<=n1;j1++)
                        {
                            if(i1!=j1)
                            {
                                String abb = lbl[i1][j1].getText ();
                                if(abb.equals ("*"))
                                {
                                   a[i1-1][j1-1] = 1;
                                }
                                else
                                    a[i1-1][j1-1] = 0;
                            }
                        }
                    }
                    int response = JOptionPane.showConfirmDialog (this,"Are you sure you want to
update the existing interaction matrix?","Confirm Update
Dialog",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
                    if(response == JOptionPane.YES_OPTION)
                    {
                        try
                        {
                            stmt.executeUpdate ("drop table if exists interaction;");
                            stmt.executeUpdate("create table interaction (baseid1
varchar(15),baseid2 varchar(15),interact int(1));");
                            for(int i1=1;i1<n1;i1++)
                            {
                                for(int j1=1;j1<n1;j1++)
                                {
                                    if(i1!=j1)
                                    {
                                        stmt.executeUpdate("insert into interaction
(baseid1,baseid2, interact) values(\'" + anb[i1] + "\',\'"+ anb[j1] + "\'," + a[i1][j1] + ");");
                                    }
                                }
                            }
                            stmt.close();
                            stmt2.close();
                            conn.close();
                        }
                        catch (Exception ex)
                        {
                            System.out.println(ex);
                        }
                    }
                }
                else
                {
                    this.dispose();
                    main2 mm = new main2();
                    mm.Gen2 ();
                }
            }
           /*public static void main(String[] args)
            {
                interaction mo = new interaction();
                mo.init1();
            }*/
        }
```

8) main1.java

```
//Shows the GUI screen for the Main Menu
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.sql.*;
import java.net.*;
import java.awt.event.*;
public class main1 extends JFrame implements ActionListener
```

```
{
    public javax.swing.JButton jButton1;
    JButton designrep,genconp,genrule,genDFA,back2;
    public javax.swing.JLabel jLabel1,jLabel0,jLabe20,jLabe30;
    public javax.swing.JLabel jLabel2,jLabel4;
    public javax.swing.JPasswordField jPasswordField1;
    public javax.swing.JTextField jTextField1;
    JPanel p;
     public void Gen1()
    {
        p = new JPanel(new GridBagLayout());
        this.setSize(700,700);
        this.setResizable(false);
        this.setTitle("Main Menu");
        jLabe20 = new JLabel("Main Menu");
        jLabe20.setFont(new Font("System",Font.BOLD,25));
        genconp = new JButton("Generate Conceptual Designs");
        genconp.addActionListener (this);
        designrep = new JButton("Populate Design Repository");
        designrep.addActionListener(this);
        JButton back1 = new JButton("Exit to Start-up Page");
        back1.addActionListener (this);
        GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
        gridBagConstraintsx00.gridx = 0;
        gridBagConstraintsx00.gridy = 0;
        gridBagConstraintsx00.gridwidth = 1;
        gridBagConstraintsx00.insets = new Insets(5,5,5,5);
        p.add(jLabe20,gridBagConstraintsx00);
        GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
        gridBagConstraintsx01.gridx = 0;
        gridBagConstraintsx01.gridy = 1;
        gridBagConstraintsx01.gridwidth = 1;
        gridBagConstraintsx01.insets = new Insets(5,5,5,5);
        p.add(genconp,gridBagConstraintsx01);
        GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
        gridBagConstraintsx02.gridx = 0;
        gridBagConstraintsx02.gridy = 2;
        gridBagConstraintsx02.gridwidth = 1;
        gridBagConstraintsx02.insets = new Insets(5,5,5,5);
        p.add(designrep,gridBagConstraintsx02);
        GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
        gridBagConstraintsx03.gridx = 0;
        gridBagConstraintsx03.gridy = 3;
        gridBagConstraintsx03.gridwidth = 1;
        gridBagConstraintsx03.insets = new Insets(5,5,5,5);
        p.add(back1,gridBagConstraintsx03);
        this.getContentPane().add(p);
        setVisible(true);
    }
     public void actionPerformed(ActionEvent ae)
    {
               String str = ae.getActionCommand();
               if(str.equals("Generate Conceptual Designs"))
               {
                    this.dispose();
                    inprules ir = new inprules();
                    ir.init1();
                }
                else if(str.equals("Populate Design Repository"))
                {
                    this.dispose();
                    main2 gg = new main2();
                    gg.Gen2();
                }
                else if(str.equals("Exit to Start-up Page"))
                {
                    authen1 aa = new authen1();
                    this.dispose();
                    aa.authen2 ();
                }
     }

}
```

9) main2.java

```
//Shows the Design Repository menu
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.sql.*;
import java.net.*;
import java.awt.event.*;
public class main2 extends JFrame implements ActionListener
{
    public javax.swing.JButton jButton1;
    JButton designrep,genconp,genrule,genDFA,back2,dispcomp,disprule,interaction;
    public javax.swing.JLabel jLabel1,jLabel0,jLabe20,jLabe30;
    public javax.swing.JLabel jLabel2,jLabel4;
    public javax.swing.JPasswordField jPasswordField1;
    public javax.swing.JTextField jTextField1;
    JPanel p;
     public void Gen2()
    {
            p = new JPanel(new GridBagLayout());
            this.setSize(700,700);
            this.setResizable(false);
            this.setTitle ("Design Repository Menu");
            jLabe30 = new JLabel("Design Repository");
            jLabe30.setFont(new Font("System",Font.BOLD,25));
            genrule = new JButton("Input Rule(s)");
            genrule.addActionListener (this);
            genDFA = new JButton("Input Component(s)");
            genDFA.addActionListener (this);
            back2 = new JButton("Back to Main Menu");
            back2.addActionListener (this);
            dispcomp = new JButton("View Component(s)");
            dispcomp.addActionListener(this);
            interaction = new JButton("Input Interaction(s)");
            interaction.addActionListener (this);
            disprule = new JButton("View Rule(s)");
            disprule.addActionListener(this);
            GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
            gridBagConstraintsx00.gridx = 0;
            gridBagConstraintsx00.gridy = 0;
            gridBagConstraintsx00.gridwidth = 1;
            gridBagConstraintsx00.insets = new Insets(5,5,5,5);
            p.add(jLabe30,gridBagConstraintsx00);
            GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
            gridBagConstraintsx01.gridx = 0;
            gridBagConstraintsx01.gridy = 1;
            gridBagConstraintsx01.gridwidth = 1;
            gridBagConstraintsx01.insets = new Insets(5,5,5,5);
            p.add(genrule,gridBagConstraintsx01);
            GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
            gridBagConstraintsx02.gridx = 0;
            gridBagConstraintsx02.gridy = 3;
            gridBagConstraintsx02.gridwidth = 1;
            gridBagConstraintsx02.insets = new Insets(5,5,5,5);
            p.add(genDFA,gridBagConstraintsx02);
            GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
            gridBagConstraintsx03.gridx = 0;
            gridBagConstraintsx03.gridy = 4;
            gridBagConstraintsx03.gridwidth = 1;
            gridBagConstraintsx03.insets = new Insets(5,5,5,5);
            p.add(dispcomp,gridBagConstraintsx03);
            GridBagConstraints gridBagConstraintsx03a = new GridBagConstraints();
            gridBagConstraintsx03a.gridx = 0;
            gridBagConstraintsx03a.gridy = 2;
            gridBagConstraintsx03a.gridwidth = 1;
            gridBagConstraintsx03a.insets = new Insets(5,5,5,5);
            p.add(disprule,gridBagConstraintsx03a);
            GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
            gridBagConstraintsx04.gridx = 0;
            gridBagConstraintsx04.gridy = 5;
            gridBagConstraintsx04.gridwidth = 1;
            gridBagConstraintsx04.insets = new Insets(5,5,5,5);
            p.add(interaction,gridBagConstraintsx04);
            GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
            gridBagConstraintsx05.gridx = 0;
            gridBagConstraintsx05.gridy = 6;
            gridBagConstraintsx05.gridwidth = 1;
            gridBagConstraintsx05.insets = new Insets(5,5,5,5);
            p.add(back2,gridBagConstraintsx05);
            this.getContentPane().add(p);
            setVisible(true);
    }
     public void actionPerformed(ActionEvent ae)
    {
               String str = ae.getActionCommand();
               if(str.equals("Input Rule(s)"))
               {
                    this.dispose();
                    rules rr = new rules();
                    rr.init1();
                }
                else if(str.equals("Input Component(s)"))
                {
                    this.dispose();
                    DFA gg = new DFA();
                    gg.init1 ();
                }
                else if(str.equals("View Component(s)"))
                {
                    this.dispose();
                    DFA2 gg1 = new DFA2();
                    gg1.Gen1 ();
                }
                else if(str.equals("View Rule(s)"))
                {
                    this.dispose();
                    viewRules gg1 = new viewRules();
                    gg1.Gen1 ();
                }
                else if(str.equals("Back to Main Menu"))
                {
                    main1 aa = new main1();
                    this.dispose();
                    aa.Gen1 ();
                }
                else if(str.equals("Input Interaction(s)"))
                {
                    interaction mo = new interaction();
                    this.dispose();
                    mo.init1 ();
                }
     }

}
```

10) mod.java

```
// Displays the modules after the decomposition algorithm
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.sql.*;
import java.net.*;
import javax.imageio.*;
import javax.swing.border.*;
import java.io.*;
import java.awt.event.*;
public class mod extends JFrame implements ActionListener
{
    JPanel p;
    JTextField[] tf = new JTextField[100];
    String driver,url;
    Connection conn;
    int l=0,ddx1,ab1,num1,ddy1,mst=0,mts=0,a1,a2;
    Statement stmt,stmt1;
    ResultSet rs, rse;
    BufferedImage imageo;
    ImageIcon image1;
    StringTokenizer st;
```

```
            JButton b1,main1,dfv1;
            JLabel[] img = new JLabel[100];
            JPanel[] p1 = new JPanel[100];
            Border[] linec = new Border[100];
            JLabel[] lbl = new JLabel[100];
            JLabel md;
            GridBagConstraints[][] gridBagConstraints = new GridBagConstraints[10][100];
            GridBagConstraints[] gridBagConstraints1 = new GridBagConstraints[100];
            GridBagConstraints[] gridBagConstraints2 = new GridBagConstraints[100];
            GridBagConstraints[] gridBagConstraints3 = new GridBagConstraints[100];
            String[] mod = new String[20];
            int[][] dd1 = new int[10][100];
            public void init1(int num, int[][] dd, int ddx, int ddy, int bb)
            {
                ab1 = bb; //Keeps a track of whether 1st or 2nd design alternative.
                ddx1 = ddx;
                num1 = num;
                ddy1 = ddy;
                dd1=dd;
                p = new JPanel(new GridBagLayout());
                this.setSize(700,700);
                this.setTitle("Modules after Decomposition Algorithm");
                md = new JLabel("Module(s) Obtained after DA");
                main1 = new JButton("Design Repository Menu");
                main1.addActionListener(this);
                dfv1 = new JButton("Generate DFV Index");
                dfv1.addActionListener (this);
                md.setFont(new Font("System",Font.BOLD,25));
                try
                {
                    driver = "com.mysql.jdbc.Driver";
                    url = "jdbc:mysql://localhost:3306/test";
                    Class.forName(driver);
                    conn =DriverManager.getConnection(url,"root","pennstate");
                    stmt = conn.createStatement();
                    stmt1 = conn.createStatement ();
                    rs = stmt.executeQuery("Select * from final_info where num = " + num + "
order by dfa_index_sum ASC;");
                    while(rs.next())
                    {
                        st = new StringTokenizer(rs.getString ("compon_id"),"$");
                        while(st.hasMoreTokens ())
                        {
                            rse = stmt1.executeQuery("select * from DFA where compid = \'" +
st.nextToken () + "\';");
                            while(rse.next())
                            {
                                l++;
                                imageo = ImageIO.read(new File(rse.getString ("image1")));
                                int w = (int)(imageo.getWidth()*0.4);
                                int h = (int)(imageo.getHeight ()*0.4);
                                Image imagei = imageo.getScaledInstance(w, h,
Image.SCALE_AREA_AVERAGING);
                                BufferedImage image00 = new BufferedImage(w, h,
BufferedImage.TYPE_INT_RGB);
                                Graphics2D g = image00.createGraphics();
                                g.drawImage(imagei, 0, 0, null);
                                g.dispose();
                                image1 = new ImageIcon(image00);
                                img[l] = new JLabel(image1);
                            }
                        }
                    }
                    GridBagConstraints gridBagConstraintsx011a = new GridBagConstraints();
                    gridBagConstraintsx011a.gridx = 0;
                    gridBagConstraintsx011a.gridy = 0;
                    gridBagConstraintsx011a.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx011a.gridwidth = 4;
                    gridBagConstraintsx011a.fill = GridBagConstraints.BOTH;
                    p.add(md, gridBagConstraintsx011a);
                    for(int u=1; u<=ddx; u++)
                    {
                        mts++;
                        p1[u] = new JPanel(new GridBagLayout());
                        linec[u] = new LineBorder(Color.BLACK);
                        p1[u].setBorder(linec[u]);
                        lbl[u] = new JLabel("Enter Name for Module #" + u + " : ");
                        tf[u] = new JTextField();
                        mst=0;
                        for(int v=1; v<=ddy; v++)
                        {
                            if(dd[u][v]!=0)
                            {
                                mst++;
                                gridBagConstraints[u][v] = new GridBagConstraints();
                                gridBagConstraints[u][v].gridx = v;
                                gridBagConstraints[u][v].gridy = 0;
                                gridBagConstraints[u][v].gridwidth = 1;
                                gridBagConstraints[u][v].insets = new Insets(5,5,5,5);
                                p1[u].add(img[dd[u][v]],gridBagConstraints[u][v]);
                            }
                        }
                        gridBagConstraints1[u] = new GridBagConstraints();
                        gridBagConstraints1[u].gridx = 0;
                        gridBagConstraints1[u].gridy = 1;
                        //gridBagConstraints1[u].insets = new Insets(5,5,5,5);
                        gridBagConstraints1[u].gridwidth = mst+1;
                        gridBagConstraints1[u].fill = GridBagConstraints.BOTH;
                        p1[u].add(lbl[u], gridBagConstraints1[u]);
                        gridBagConstraints2[u] = new GridBagConstraints();
                        gridBagConstraints2[u].gridx = 0;
                        gridBagConstraints2[u].gridy = 2;
                        //gridBagConstraints2[u].insets = new Insets(5,5,5,5);
                        gridBagConstraints2[u].gridwidth = mst+1;
                        gridBagConstraints2[u].fill = GridBagConstraints.BOTH;
                        p1[u].add(tf[u], gridBagConstraints2[u]);
                        gridBagConstraints3[u] = new GridBagConstraints();
                        gridBagConstraints3[u].gridx = u-1;
                        gridBagConstraints3[u].gridy = 1;
                        gridBagConstraints3[u].insets = new Insets(5,5,5,5);
                        gridBagConstraints3[u].gridwidth = 1;
                        gridBagConstraints3[u].fill = GridBagConstraints.BOTH;
                        p.add(p1[u], gridBagConstraints3[u]);
                    }
                    if(mts%2==0)//number of modules!
                    {
                        a1=(int)mts/2;
                        a2=(int)mts/2;
                    }
                    else
                    {
                        a1=(int)((mts-1)/2);
                        a2=(int)((mts+1)/2);
                    }
                    GridBagConstraints gridBagConstraintsx015 = new GridBagConstraints();
                    gridBagConstraintsx015.gridx = 0;
                    gridBagConstraintsx015.gridy = 2;
                    gridBagConstraintsx015.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx015.gridwidth = a1;
                    gridBagConstraintsx015.fill = GridBagConstraints.BOTH;
                    p.add(dfv1, gridBagConstraintsx015);
                    GridBagConstraints gridBagConstraintsx016 = new GridBagConstraints();
                    gridBagConstraintsx016.gridx = a1;
                    gridBagConstraintsx016.gridy = 2;
                    gridBagConstraintsx016.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx016.gridwidth = a2;
                    gridBagConstraintsx016.fill = GridBagConstraints.BOTH;
                    p.add(main1, gridBagConstraintsx016);
                    this.getContentPane().add(p);
                    setVisible(true);
                }
                catch (Exception ex)
                {
                    System.out.println(ex);
                }
            }
            public void actionPerformed(ActionEvent ae)
            {
                String str = ae.getActionCommand();
                if(str.equals("Design Repository Menu"))
                {
                    this.dispose();
```

```
                    main1 mm = new main1();
                    mm.Gen1 ();
                }
                if(str.equals("Generate DFV Index"))
                {
                    for(int i=1; i<=ddx1; i++)
                    {
                        mod[i] = tf[i].getText();
                    }
                    this.dispose();
                    if(ab1==1)
                    {
                        QFD1 qf = new QFD1();
                        qf.init1(mod,ddx1,ab1,dd1,ddy1,num1);
                    }
                    if(ab1==2)//for second design combination, no need to go through QFD1
                    {
                        QFD2 qf = new QFD2();
                        qf.init1(mod,ddx1,ab1,dd1,ddy1,num1);
                    }
                }
            }
        }
```

11) modul.java

```
        //Inputs the Suitability Matrix as it would depend on the selected design
        import javax.swing.*;
        import java.util.*;
        import java.awt.*;
        import java.awt.image.BufferedImage;
        import java.sql.*;
        import java.net.*;
        import javax.imageio.*;
        import javax.swing.border.*;
        import java.io.*;
        import java.awt.event.*;
        public class modul extends JFrame implements 
MouseMotionListener,MouseListener,ActionListener
        {
            JPanel p,p1;
            JTextField tf;
            public int step,bbx,bby;
            String driver,url;
            Connection conn;
            Statement stmt,stmt1;
            ResultSet rs,rse;
            int l=0,n=4, num1,ab1,m=0;
            BufferedImage imageo, image00;
            Image imagei;
            JButton modul;
            ImageIcon image1;
            JLabel[] img = new JLabel[100];
            int[] abc = new int[100];
            JLabel[] img2 = new JLabel[100];
            JLabel[][] lbl = new JLabel[10][10];
            public static int[][] dd1 = new int[10][100];
            String[][] ddstng = new String[10][100];
            JPanel[][] ai = new JPanel[10][10];
            Border lineb,linec;
            JLabel newp;
            StringTokenizer st;
            char[][] a = new char[10][10];
            String[] anb = new String[100];
            public void init1(int num, int step1, int[] abc1, int[][] dd, int ddx, int ddy,int 
bb)
            {
                ab1 = bb; //Keeps a track of whether 1st or 2nd design alternative.
                step = step1;
                abc = abc1;
                dd1 = dd;
                bbx = ddx;
                bby = ddy;
                num1 = num;
                p1 = new JPanel(new GridBagLayout());
                p = new JPanel();
                setSize(700,700);
                p.setLayout(new GridLayout(step,step));
                p.setForeground(Color.WHITE);
                setResizable(false);
                this.setTitle("Suitability Matrix");
                lineb = new LineBorder(Color.gray);
                linec = new LineBorder(Color.BLACK);
                try
                {
                    driver = "com.mysql.jdbc.Driver";
                    url = "jdbc:mysql://localhost:3306/test";
                    Class.forName(driver);
                    conn =DriverManager.getConnection(url,"root","pennstate");
                    stmt = conn.createStatement();
                    stmt1 = conn.createStatement ();
                    rs = stmt.executeQuery("Select * from final_info where num = " + num + "
order by dfa_index_sum ASC;");
                    while(rs.next())
                    {
                        st = new StringTokenizer(rs.getString ("compon_id"),"$");
                        while(st.hasMoreTokens ())
                        {
                            m++;
                            anb[m] = st.nextToken ();
                            rse = stmt1.executeQuery("select * from DFA where compid = \'" +
anb[m] + "\';");
                            while(rse.next())
                            {
                                l++;
                                imageo = ImageIO.read(new File(rse.getString ("image1")));
                                int w = (int)(imageo.getWidth()*0.4);
                                int h = (int)(imageo.getHeight ()*0.4);
                                Image imagei = imageo.getScaledInstance(w, h,
Image.SCALE_AREA_AVERAGING);
                                BufferedImage image00 = new BufferedImage(w, h,
BufferedImage.TYPE_INT_RGB);
                                Graphics2D g = image00.createGraphics();
                                g.drawImage(imagei, 0, 0, null);
                                g.dispose();
                                image1 = new ImageIcon(image00);
                                img[l] = new JLabel(image1);
                                img2[l] = new JLabel(image1);
                            }
                        }
                    }
                    modul = new JButton("Modularize the Design");
                    modul.addActionListener (this);
                    for(int x=1;x<=bbx;x++)
                    {
                        for(int y=1;y<=bby;y++)
                        {
                            if(dd[x][y]!=0)
                            {
                                ddstng[x][y] = anb[dd[x][y]];
                            }
                            else ddstng[x][y] = "";
                        }
                    }
                    /*for(int x=1;x<=ddx;x++)
                    {
                        for(int y=1;y<=ddy;y++)
                        {
                            if(dd[x][y]!=0)
                            {
                                System.out.println ("Mod: " + ddstng[x][y]);
                            }
                        }
                        System.out.print("\n");
                    }*/
                    ai[1][1] = new JPanel();
                    p.add(ai[1][1]);
                    for(int m=2;m<=step;m++)
                    {
                        ai[1][m] = new JPanel();
                        ai[1][m].add(img[m-1]);
                        p.add (ai[1][m]);
```

```
        }
        for(int i = 2; i<=step; i++)
        {
            for(int j=1;j<=step;j++)
            {
                if(j==1)
                {
                    ai[i][1] = new JPanel();
                    ai[i][1].add(img2[i-1]);
                    p.add (ai[i][1]);
                }
                else if(i==j)
                {
                    JLabel plus = new JLabel("+");
                    plus.setFont(new Font("System",Font.BOLD,30));
                    ai[i][j] = new JPanel();
                    ai[i][j].setBorder (lineb);
                    ai[i][j].add(plus);
                    p.add (ai[i][j]);
                }
                else
                {
                    ai[i][j] = new JPanel();
                    ai[i][j].setBorder (lineb);
                    lbl[i][j] = new JLabel();
                    lbl[i][j].setText (" ");
                    ai[i][j].add (lbl[i][j]);
                    ai[i][j].addMouseMotionListener(this);
                    ai[i][j].addMouseListener(this);
                    ai[i][j].setForeground(Color.WHITE);
                    p.add(ai[i][j]);
                }
            }
        }
        newp = new JLabel("Input the Suitability Matrix");
        newp.setFont(new Font("System",Font.BOLD,30));
        GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
        gridBagConstraintsx00.gridx = 0;
        gridBagConstraintsx00.gridy = 0;
        gridBagConstraintsx00.gridwidth = 2;
        gridBagConstraintsx00.insets = new Insets(5,5,5,5);
        p1.add(newp,gridBagConstraintsx00);
        GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
        gridBagConstraintsx01.gridx = 0;
        gridBagConstraintsx01.gridy = 1;
        gridBagConstraintsx01.gridwidth = 2;
        gridBagConstraintsx01.insets = new Insets(5,5,5,5);
        p1.add(p, gridBagConstraintsx01);
        GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
        gridBagConstraintsx02.gridx = 0;
        gridBagConstraintsx02.gridy = 2;
        gridBagConstraintsx02.insets = new Insets(5,5,5,5);
        gridBagConstraintsx02.gridwidth = 1;
        gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
        p1.add(modul, gridBagConstraintsx02);
        GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
        gridBagConstraintsx03.gridx = 1;
        gridBagConstraintsx03.gridy = 2;
        gridBagConstraintsx03.insets = new Insets(5,5,5,5);
        gridBagConstraintsx03.gridwidth = 1;
        gridBagConstraintsx03.fill = GridBagConstraints.BOTH;
        p1.add(new JButton("Main Menu"), gridBagConstraintsx03);
        this.getContentPane().add(p1);
        setVisible(true);
    }
    catch (Exception ex)
    {
        System.out.println(ex.getMessage());
    }
}
public void mouseMoved(MouseEvent e)
{
   for(int i1=2;i1<=step;i1++)
   {
     for(int j1=2;j1<=step;j1++)
      {
            if(e.getComponent () == ai[i1][j1])
            {
                ai[i1][j1].setBorder(linec);
                p.revalidate();
            }
            else
            {
                ai[i1][j1].setBorder(lineb);
                p.revalidate();
            }
       }
   }
 }
public void mouseDragged(MouseEvent e)
{
   //System.out.println("Mouse dragged");
}
public void mousePressed(MouseEvent e)
{
   //System.out.println("Mouse Pressed");
}
public void mouseReleased(MouseEvent e)
{
   //System.out.println("Mouse Released");
}
public void mouseEntered(MouseEvent e)
{
   //System.out.println("Mouse Entered");
}
public void mouseExited(MouseEvent e)
{
   //System.out.println ("Mouse Exited");
}
public void mouseClicked(MouseEvent e)
{
   for(int i1=2;i1<=step;i1++)
   {
     for(int j1=2;j1<=step;j1++)
      {
            if(e.getComponent () == ai[i1][j1])
            {
                String abb = lbl[i1][j1].getText();
                if(abb.equals(" "))
                {
                    lbl[i1][j1].setText ("a");
                    lbl[i1][j1].setFont(new Font("System",Font.BOLD,30));
                }
                else if(abb.equals("a"))
                {
                    lbl[i1][j1].setText ("e");
                    lbl[i1][j1].setFont(new Font("System",Font.BOLD,30));
                }
                else if(abb.equals("e"))
                {
                    lbl[i1][j1].setText ("o");
                    lbl[i1][j1].setFont(new Font("System",Font.BOLD,30));
                }
                else if(abb.equals("o"))
                {
                    lbl[i1][j1].setText ("u");
                    lbl[i1][j1].setFont(new Font("System",Font.BOLD,30));
                }
                else if(abb.equals("u"))
                {
                    lbl[i1][j1].setText (" ");
                    lbl[i1][j1].setFont(new Font("System",Font.BOLD,30));
                }
                p.revalidate();
            }
       }
   }
}
public void actionPerformed (ActionEvent ae)
{
    String str = ae.getActionCommand ();
    String bb;
```

```
                int[][] ab2 = new int[10][10]; //stores the numeric value corresponding to a, e,
o,u.
                if(str.equals ("Modularize the Design"))
                {
                    for(int i1=1;i1<=step;i1++)
                    {
                        for(int j1=1;j1<=step;j1++)
                        {
                            if((i1 == 1)&&(j1 == 1))
                            {
                                bb = Integer.toString(0);
                                a[i1][j1] = bb.charAt (0);
                            }
                            if((i1 == 1)&&(j1 > 1))
                            {
                                bb = Integer.toString(j1-1);
                                a[i1][j1] = bb.charAt (0);
                            }
                            if((i1 > 1)&&(j1 == 1))
                            {
                                bb = Integer.toString(i1-1);
                                a[i1][j1] = bb.charAt (0);
                            }
                            if((i1 > 1)&&(j1 > 1))
                            {
                                if(i1!=j1)
                                {
                                   bb = lbl[i1][j1].getText ();
                                    a[i1][j1] = bb.charAt (0);
                                }
                                else
                                    a[i1][j1] = '+';
                            }
                        }
                    }
                    for(int i1=2;i1<=step;i1++)
                    {
                        for(int j1=2;j1<=step;j1++)
                        {
                            if(i1!=j1)
                            {
                                if(a[i1][j1] == 'a')
                                    ab2[i1-1][j1-1] = 2;
                                else if(a[i1][j1] == 'e')
                                    ab2[i1-1][j1-1] = 1;
                                else if(a[i1][j1] == 'o')
                                    ab2[i1-1][j1-1] = -2;
                                else if(a[i1][j1] == 'u')
                                    ab2[i1-1][j1-1] = -1;
                            }
                        }
                    }
                    try
                    {
                        stmt.executeUpdate ("drop table if exists suitability;");
                        stmt.executeUpdate("create table suitability (comp1 varchar(15),comp2
varchar(15),suitable int(2));");
                        for(int i1=1;i1<step;i1++)
                        {
                            for(int j1=1;j1<step;j1++)
                            {
                                if(i1!=j1)
                                {
                                        stmt.executeUpdate("insert into suitability (comp1,comp2,
suitable) values(\'" + anb[i1] + "\',\'"+ anb[j1] + "\'," + ab2[i1][j1] + ");");
                                }
                            }
                        }
                        stmt.close();
                        conn.close();
                   }
                   catch (Exception ex)
                   {
                        System.out.println(ex);
                   }
                    /*for(int hy=1; hy<=step; hy++)
                    {
                        for(int hy1 = 1; hy1<=step; hy1++)
                        {
                            if((hy == 1)&&(hy1 == 1))System.out.print(0 + " ");
                            if((hy == 1)&&(hy1 > 1)) System.out.print(a[1][abc[hy1-1]+1] + " ");
                            if((hy > 1)&&(hy1 == 1)) System.out.print(a[abc[hy-1]+1][1] + " ");
                            if((hy > 1)&&(hy1 > 1))  System.out.print (a[abc[hy-1]+1][abc[hy1-
1]+1] + " ");
                        }
                        System.out.println ("");
                    }*/
                    suitability sub = new suitability();
                    sub.init1 (ddstng,bbx,bby,anb,ab1,step,num1);
                    this.dispose();
                }
                if(str.equals("Main Menu"))
                {
                    this.dispose();
                    main1 mm = new main1();
                    mm.Gen1 ();
                }
            }
        }
```

12) MySet.java

```
//Uses set operations/Tarjan's algorithm for Kusiak’s Decomposition Algorithm
public class MySet
{
    private static int N = 100; // Max. value of any integer in the set
    private boolean[] setrep;
    boolean flga = false;
    boolean glchk = false;
    int lvno=1,gpno;
    int ifin, jfin,seq=0,mm=0,uht=0,cc=0,i23;
    int[][] inc = new int[50][50];
    int[][] inc1 = new int[50][50];
    int[] ghy = new int[100];
    int[][] GG = new int[50][50];
    int[] OA = new int[50];
    int[] E = new int[50];
    int[] bb = new int[100];
    public int[][] gh = new int[10][100];
    public int[][] gh1 = new int[10][100];
    public int gh1x=0, gh1y=0;
    int num1,ab1;
    MySet EE;
    public MySet()
    {
        setrep = new boolean[N];
    }
    public MySet(int[] vals)
    {
        setrep = new boolean[N];
        for(int i=0; i<vals.length; i++)
        {
            setrep[vals[i]] = true;
        }
    }
    public void print()
    {
        for(int i=0; i < N; i++)
            if(setrep[i])
                System.out.print(i + " ");
        System.out.println("");

    }
    public int[] check()
    {
        int sequ=0;
        int[] ret = new int[100];
        for(int i=0; i<N; i++)
        {
            if(setrep[i])
            {
                sequ = sequ + 1;
                ret[sequ] = i;
            }
        }
        return ret;
    }
    public void add(int a)
    {
        setrep[a] = true;
    }
    public void remove(int a)
    {
        setrep[a] = false;
    }
    public boolean exists(int a)
    {
        return setrep[a];
    }
    public MySet union(MySet b)
    {
        MySet out = new MySet();
        for(int i=0; i < N; i++)
            if(setrep[i] || b.setrep[i])
                out.add(i);
        return out;
    }
    public MySet intersect(MySet b)
    {
        MySet out = new MySet();
        for(int i=0; i < N; i++)
            if(setrep[i] && b.setrep[i])
                out.add(i);
        return out;
    }
    public boolean equals(MySet b)
    {
        boolean out = true;
        for(int i=0; i < N; i++)
        {
            if(setrep[i] != b.setrep[i])
                out = false;
        }
        return out;
    }
    public void copy(MySet b)
    {
        for(int i=0; i < N; i++)
        {
            setrep[i] = b.setrep[i];
        }
    }
    public static boolean isdone = false;
    public static MySet fin = new MySet();
    public static int cyclehead = 0;

    public static void traverse(int start, MySet tset, int[][] G, int M, MySet abc)
    {
        if(isdone)
        {
            return;
        }
        if(tset.exists(start))
        {
            if(start == cyclehead) {
                isdone = true;
                abc.copy(tset);
                return;
            }
            else
                return;
        }
        tset.add(start);
        for(int j=1; j <=M; j++)
        {
            if(G[start][j] == 1)
            {
                traverse(j,tset, G, M, abc);
```

```
                    }
                }
                tset.remove(start);
                return;
             }
             public static int[] findFirstCycle(int[][] G, int step, MySet abc) //Step5 - Find a
cycle E
             {
                int [] cyc = new int[50];
                MySet tset;
                isdone = false;
                for(int st = 1; st < step; st++)
                {
                    // start with st and do DFS
                    tset = new MySet();
                    cyclehead = st;
                    traverse(st, tset, G, step,abc);
                }
                return cyc;
             }
             public static void main1(int[][] G, int M)
             {
                 findFirstCycle(G,M,fin);
                 fin.print ();
             }
             public static MySet[] L = new MySet[50]; //Set of Activities
             public static MySet[] C = new MySet[50]; //Set of Coupled Activities
             public static MySet NUL = new MySet();
             //-----------Decomposition Algorithm---------------------//
             public void step1(int[][] ab, int i1, int j1, int num,int bb) //Initialization
             {
                 gpno=1;
                 lvno=1;
                 num1 = num; //Keep a track of the row in final_info for suitability matrix.
                 i23 = i1+1; //Keep a track of the step size from NewEmpty.java
                 ab1 = bb; //Keeps track of whether 1st or 2nd design alternative.
                 L[lvno] = new MySet();
                 C[gpno] = new MySet();
                 ifin = i1;
                 jfin = j1+1;
                 for(int ia = 0; ia <= ifin; ia++)
                 {
                     for(int ja=0;ja <= jfin; ja++)
                     {
                         if(ia==0)
                         {
                             inc[ia][ja] = ja;
                             inc1[ia][ja] = ja;
                         }
                         else if(ja==0)
                         {
                             inc[ia][ja] = ia;
                             inc1[ia][ja] = ia;
                         }
                         else if(ja == j1+1)
                         {
                             inc[ia][ja] = 1;
                             inc1[ia][ja] = 1;
                         }
                         else
                         {
                             inc[ia][ja] = ab[ia][ja];
                             inc1[ia][ja] = ab[ia][ja];
                         }
                      }
                 }
                 for(int ia = 0; ia <= i1; ia++)
                 {
                     for(int ja=0;ja <= j1+1; ja++)
                     {
                       System.out.print(inc[ia][ja] + " ");
                     }
                     System.out.print("\n");
                 }
                 step2();
             }
```

```
            public void step2() //Check if incidence matrix is empty
            {
                uht=0;
                if((ifin == 0)&&(jfin == 1))
                {
                    System.out.println ("Finished!!");
                    if(L[lvno].equals (NUL)) lvno = lvno - 1;
                    if(C[gpno].equals (NUL)) gpno = gpno - 1;
                    System.out.println ("lvno = " + lvno + "  " + " gpno = " + gpno);
                    /*for(int lno = 1; lno < lvno; lno++)
                    {
                        System.out.println (" Level: ");
                        L[lno].print();
                    }
                    for(int gno = 1; gno < gpno; gno++)
                    {
                        System.out.println (" Coupled: ");
                        C[gno].print();
                    }*/
                    for(int uy=1; uy<=lvno; uy++)//Printing the output.
                    {
                        L[uy].print ();
                        gh[uy] = L[uy].check ();
                    }
                    for(int uy=1; uy<=lvno; uy++)
                    {
                        gh1x++;
                        for(int ug=0; ug<gh.length;ug++)
                       {
                           if(gh[uy][ug]!=0)
                           {
                               gh1y++;
                               gh1[uy][ug] = gh[uy][ug];
                               uht++;
                               ghy[uht] = gh[uy][ug];
                           }
                       }
                    }
                    inc[0][0] = 0;
                    for(int hy=0; hy<=uht; hy++)
                    {
                        for(int hy1 = 0; hy1<=uht; hy1++)
                        {
                           if((hy == 0)&&(hy1 == 0))System.out.print(inc1[0][0] + " ");
                           if((hy == 0)&&(hy1 > 0)) System.out.print(inc1[0][ghy[hy1]] + " ");
                           if((hy > 0)&&(hy1 == 0)) System.out.print(inc1[ghy[hy]][0] + " ");
                           if((hy > 0)&&(hy1 > 0))  System.out.print (inc1[ghy[hy]][ghy[hy1]] +
" ");
                        }
                        System.out.println ("");
                    }
                    modul mo = new modul();
                    mo.init1(num1,i23,ghy,gh1,gh1x,gh1y,ab1);
                }
                else
                {
                    step3();
                }
            }
            public void step3() //Identify the original actitivites (OA)
            {
                boolean flag;
                flga = false;
                MySet fin1 = new MySet();
                seq=0;
                for (int i2 = 1; i2 <= ifin; i2++)
                {
                    flag = true;
                    for(int j2 = 1; j2 < jfin; j2++)
                    {
                        if(inc[i2][j2]==0) flag = true;
                        else
                        {
                            flag = false;
                            break;
                        }
```

```
                     }
                     if(flag==true)
                     {
                         seq = seq + 1;
                         OA[seq] = inc[i2][0];
                     }
                 }
                 if(seq > 0) //OA is not empty
                 {
                     MySet t = new MySet();
                     for(int ii = 1; ii<=seq; ii++) //for each t
                     {
                         t.add (OA[ii]);
                         System.out.print ("T is: ");
                         t.print ();
                         for(int jj = 1; jj<=gpno; jj++) //for each C
                         {
                            MySet ct = C[jj].intersect(t);
                            if(ct.equals (NUL)) // t does not belongs to set C
                            {
                               flga = true; //Checking all C's
                            }
                            if(!ct.equals(NUL)) //t belongs to C
                            {
                                System.out.println("t E C" + lvno);
                                C[jj].print();

                                flga = false;
                                for(int f=1; f<=ifin; f++)
                                {
                                    if(inc[0][f] == OA[ii])
                                    {
                                        L[inc[f][jfin]] = L[inc[f][jfin]].union (C[jj]);
                                        L[inc[f][jfin]].print();
                                        glchk = true;
                                    }
                                }
                                break;
                            }
                         }
                         if(flga==true)
                         {
                             L[lvno] = L[lvno].union (t);
                             System.out.println ("T not E C");
                         }
                         t.remove (OA[ii]);
                     }
                     step4();
                  }
                  else //if OA is empty, proceed to cycle finding
                  {
                     System.out.println ("Cycle Found");
                     /*for(int ui = 1; ui <= ifin; ui++)
                     {
                         for(int uj=1; uj <= jfin-1; uj++)
                         {
                             GG[ui][uj] = inc[ui][uj];
                             //System.out.print (GG[ui][uj] + " ");
                         }
                         //System.out.print ("\n");
                     }*/
                     findFirstCycle(inc, ifin, fin);
                     bb = fin.check();
                     EE = new MySet();
                     cc = 0;
                     for(int u2 = 0; u2<bb.length; u2++) //E[cc] contains the elements of the
cycle.
                     {
                         if(bb[u2]!=0)
                         {
                             cc++;
                             E[cc] = inc[0][bb[u2]];
                             EE.add(E[cc]);
                             System.out.println(E[cc]);
                             //System.out.println ("CC: " + cc);
                         }
```

```
        }
        step6();
    }
}
public void step4() //Deleting from incidence matrix all OA entries
{
    System.out.println ("Step4");
    int[][] temp = new int[50][50];
    for(int u=1; u<=seq; u++)
    {
        if(OA[u]!=0)
        {
            for(int y = 0; y <= ifin; y++)
            {
                if(inc[0][y] == OA[u])
                {
                    for(int z = 1; z <= ifin; z++)
                    {
                        if(inc[z][y]==1)
                        {
                            if(glchk == true)
                            {
                                inc[z][jfin] = lvno;
                                glchk = false;
                            }
                            else inc[z][jfin] = lvno + 1;
                        }
                    }
                }
            }
            for(int y = 0; y <= ifin; y++)
            {
                for(int z = 0; z <= jfin; z++)
                {
                   if(inc[y][0] < OA[u])
                   {
                       if(inc[0][z] < OA[u])
                           temp[y][z] = inc[y][z];
                       if(inc[0][z] > OA[u])
                           temp[y][z-1] = inc[y][z];
                   }
                   if(inc[y][0] > OA[u])
                   {
                       if(inc[0][z] < OA[u])
                           temp[y-1][z] = inc[y][z];
                       if(inc[0][z] > OA[u])
                           temp[y-1][z-1] = inc[y][z];
                   }
               }
            }
            ifin = ifin - 1;
            jfin = jfin - 1;
            System.out.println ("----------------------");
            for(int y=0; y<=ifin; y++)
            {
               for(int z = 0; z<= jfin; z++)
               {
                   inc[y][z] = temp[y][z];
                  System.out.print(inc[y][z] + " ");
               }
               System.out.print("\n");
            }
        }
    }
    if(!L[lvno].equals(NUL))//&&(flga == true)) //L[i] not equal to NULL
    {
        lvno = lvno + 1;
        System.out.println("LVNO incremented");
        L[lvno] = new MySet();
    }
    //else System.out.println ("SOmething wrong!!"); Does not increment L
    seq = 0;
    step2();
}
public void step6() //Merge cycles found
{
```

```
for(int s = 0; s < bb.length; s++ )
{
    for(int t = 0; t < bb.length; t++)
    {
        if((bb[s]>0)&&(bb[t]>0)&&(s!=t))// While s not equal to t
        {
            for(int i = 1; i <= ifin; i++)
            {
                if(inc[bb[s]][i] == 1)
                {
                    inc[bb[t]][i] = 1;
                    inc[bb[s]][i] = 0;
                }
                if(inc[i][bb[s]] == 1)
                {
                    inc[i][bb[t]] = 1;
                    inc[i][bb[s]] = 0;
                }
            }
        }
        if((bb[s]>0)&&(bb[t]>0)&&(s==t)) inc[bb[s]][t] = 0; //Only for debugging
    }
}
/*for(int ub = 0; ub<=ifin; ub ++)
{
    for(int uc = 0;uc <= jfin; uc++)
    {
        System.out.print(inc[ub][uc] + " ");
    }
    System.out.print("\n");
}*/
int[][] temp = new int[50][50];
int[] tempo = new int[50];
int uyc = 0,unyc=0,u,y,z;
for(u=1; u<=cc; u++)
{
   for(y=0; y <= ifin; y++)
   {
       if(inc[0][y]== E[u])
       {
           if(u==1) unyc = y;
           uyc++;
           tempo[uyc] = inc[y][jfin];
       }
   }
}
for(u=2; u<=uyc; u++) //Determining the maximum activity corresponding to a coupled activity
{
        if(tempo[u]>=tempo[u-1])
        {
            inc[unyc][jfin] = tempo[u];
        }
        else
        {
            tempo[u] = tempo[u-1];
            inc[unyc][jfin] = tempo[u-1];
        }
}
for(u=2; u<=cc; u++) //Deleting the rows from the incidence matrix corresponding to Cycles
{
    if(E[u] != 0)
    {
        System.out.println("+++++ Step 6 ++++++");
        for(y = 0; y <= ifin; y++)
        {
            for(z = 0; z <= jfin; z++)
            {
                 if((inc[0][z] < E[u])&&(inc[y][0] < E[u]))
                {
                   temp[y][z] = inc[y][z];
                }
                if((inc[0][z] > E[u])&&(inc[y][0] < E[u]))
                {
                   temp[y][z-1] = inc[y][z];
```

```
                         }
                         if((inc[y][0] > E[u])&&(inc[0][z] < E[u]))
                         {
                            temp[y-1][z] = inc[y][z];
                         }
                         if((inc[y][0] > E[u])&&(inc[0][z] > E[u]))
                         {
                            temp[y-1][z-1] = inc[y][z];
                         }
                     }
                 }
                 ifin = ifin - 1;
                 jfin = jfin - 1;
                 for(y=0; y<=ifin; y++)
                 {
                    for(z = 0; z<= jfin; z++)
                    {
                        if(y==z) inc[y][z] = 0;
                        else inc[y][z] = temp[y][z];
                        System.out.print(inc[y][z] + " ");
                    }
                    System.out.print("\n");
                 }
             }
         }
         boolean fl=true;
         for(int gn=1; gn<=gpno; gn++)
         {
            MySet CE = C[gn].intersect (EE);
            if(!CE.equals (NUL)) //C n E = C(q) for some q
            {
                C[gn] = C[gn].union(EE);
                System.out.print ("C U E: "); C[gn].print();
                fl=false;
                break;
            }
         }
         if(fl==true)
         {
            C[gpno] = C[gpno].union(EE);
            gpno = gpno + 1;
            C[gpno] = new MySet();
         }
            step3();
     }
}
```

13) NewEmpty.java

```
//Generates all the possible combinations of components to meet overall function
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.sql.*;
import java.net.*;
import java.awt.event.*;
import java.applet.*;
import javax.imageio.*;
import java.awt.image.BufferedImage;
import java.io.*;

public class NewEmpty extends JFrame implements ActionListener, ItemListener
{
    String url,driver;
    JTextField[] baseid = new JTextField[1000];
    JLabel jbl;
    JButton main1;
    JButton[] save = new JButton[1000];
    JLabel[] modul = new JLabel[1000];
    JCheckBox[] mod = new JCheckBox[1000]; //Needs ItemListener
    FileDialog fld;
    Connection conn;
    Statement stmt,stmt1,stmt2;
    ResultSet rset,rse,rse2;
    String[] baid = new String[1000];
    int i=1,record=0,j=0,k=0,l=0,m=0,ab1;
```

```
            public static int step=1,flag=0;
            public static int[][] a = new int[10][10];
            public static int[][] inc = new int[20][20];
            StringTokenizer st;
            JPanel[] jp = new JPanel[1000];
            JPanel p;
            JTabbedPane tp;
            BufferedImage imageo;
            ImageIcon image1;
            GridBagConstraints[] bag = new GridBagConstraints[1000];
            GridBagConstraints[] bg2 = new GridBagConstraints[1000];
            GridBagConstraints[] bg3 = new GridBagConstraints[1000];
            GridBagConstraints[] bg4 = new GridBagConstraints[1000];
            GridBagConstraints[] bg5 = new GridBagConstraints[1000];
            public void new2(int ab)
            {
                ab1 = ab; //Keeps a track of whether the 1st or 2nd alternative.
                p = new JPanel();
                setSize(700,700);
                p.setLayout(new GridBagLayout());
                setResizable(false);
                this.setTitle("Generated Conceptual Designs");
                try
                {
                    tp = new JTabbedPane();
                    driver = "com.mysql.jdbc.Driver";
                    url = "jdbc:mysql://localhost:3306/test";
                    Class.forName(driver);
                    conn =DriverManager.getConnection(url,"root","pennstate");
                    stmt = conn.createStatement();
                    stmt1 = conn.createStatement();
                    stmt2 = conn.createStatement ();
                    rse = stmt.executeQuery("Select * from final_info order by dfa_index_sum
ASC;");
                    while(rse.next ())
                    {
                        record++;
                    }
                    jbl = new JLabel("Total Number of Designs Generated: " + record);
                    jbl.setFont(new Font("System",Font.BOLD,18));
                    main1 = new JButton("Design Repository Menu");
                    main1.addActionListener(this);
                    rset = stmt.executeQuery("Select * from final_info order by num ASC;");
                    GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
                    gridBagConstraintsx00.gridx = 0;
                    gridBagConstraintsx00.gridy = 0;
                    gridBagConstraintsx00.insets = new Insets(5,5,5,5);
                    p.add(jbl, gridBagConstraintsx00);
                    while (rset.next())
                    {
                        j++;
                        l=0;m=0;
                        jp[j] = new JPanel();
                        jp[j].setLayout(new GridBagLayout());
                        save[j] = new JButton("Save This Design");
                        save[j].addActionListener (this);
                        baseid[j] = new JTextField(25);
                        baseid[j].setEditable(false);
                        mod[j] = new JCheckBox("" + j);
                        mod[j].addItemListener (this);
                        modul[j] = new JLabel("Select for Modularization");
                        st = new StringTokenizer(rset.getString ("compon_id"),"$");
                        baseid[j].setText ("DFA Index: " + rset.getString ("dfa_index_sum"));
                        while(st.hasMoreTokens ())
                        {
                            rse = stmt1.executeQuery("select * from DFA where compid = \'" +
st.nextToken () + "\';");
                            while(rse.next())
                            {
                                if(flag==0)
                                {
                                    baid[step] = new String(rse.getString ("baseid"));
                                    step++;
                                }
                                imageo = ImageIO.read(new File(rse.getString ("image1")));
                                imageo.getScaledInstance(200, 200, Image.SCALE_SMOOTH);
                                image1 = new ImageIcon(imageo);
                                bag[k] = new GridBagConstraints();
                              bag[k].gridx = m;
                              bag[k].gridy = l;
                              bag[k].insets = new Insets(5,5,5,5);
                              bag[k].gridwidth = 1;
                              bag[k].fill = GridBagConstraints.BOTH;
                              jp[j].add(new JLabel(image1), bag[k]);
                                k++;
                                m++;
                                if(m==3)
                                {
                                    m=0;
                                    l++;
                                }
                             }
                        }
                        flag=1;
                        l=l+1;
                        bg2[j] = new GridBagConstraints();
                        bg2[j].gridx = 0;
                        bg2[j].gridy = l;
                        bg2[j].insets = new Insets(5,5,5,5);
                       bg2[j].gridwidth = 3;
                              bg2[j].fill = GridBagConstraints.BOTH;
                        jp[j].add(baseid[j], bg2[j]);
                        l=l+1;
                        bg3[j] = new GridBagConstraints();
                        bg3[j].gridx = 0;
                        bg3[j].gridy = l;
                        bg3[j].insets = new Insets(5,5,5,5);
                       bg3[j].gridwidth = 3;
                              bg3[j].fill = GridBagConstraints.BOTH;
                        jp[j].add (save[j],bg3[j]);
                        l=l+1;
                        bg4[j] = new GridBagConstraints();
                        bg4[j].gridx = 0;
                        bg4[j].gridy = l;
                        bg4[j].insets = new Insets(5,5,5,5);
                       bg4[j].gridwidth = 1;
                              bg4[j].fill = GridBagConstraints.BOTH;
                        jp[j].add (mod[j],bg4[j]);
                        bg5[j] = new GridBagConstraints();
                        bg5[j].gridx = 1;
                        bg5[j].gridy = l;
                        bg5[j].insets = new Insets(5,5,5,5);
                       bg5[j].gridwidth = 2;
                              bg5[j].fill = GridBagConstraints.BOTH;
                        jp[j].add (modul[j],bg5[j]);
                        tp.addTab(""+j,jp[j]);
                     }
                    for(int u1=1;u1<step;u1++)
                        //System.out.println (baid[u1]); //Printout the names of the components
                    for(int i1=1;i1<step;i1++)
                    {
                        for(int j1=1;j1<step;j1++)
                        {
                            if(i1!=j1)
                            {
                                rse2 = stmt2.executeQuery("select * from interaction where
baseid1 = \'" + baid[i1] + "\' and baseid2 = \'" + baid[j1] + "\';");
                                while(rse2.next())
                                {
                                    a[i1][j1] = (int) rse2.getInt ("interact");
                                }
                            }
                            else
                                a[i1][j1]=0;
                        }
                    }
                    for(int i1=0;i1<step;i1++)
                    {
                        for(int j1=0;j1<step;j1++)
                        {
                           if((i1 == 0)&&(j1 > 0)) inc[0][j1] = j1;
                           if((i1 > 0)&&(j1 == 0)) inc[i1][0] = i1;
                           if((i1>0)&&(j1>0)) inc[i1][j1] = a[i1][j1];
                           //System.out.print(inc[i1][j1] + " ");
                        }
                        //System.out.print("\n");
                    }
                    GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
                    gridBagConstraintsx01.gridx = 0;
                    gridBagConstraintsx01.gridy = 1;
                    gridBagConstraintsx01.insets = new Insets(5,5,5,5);
                    p.add(tp, gridBagConstraintsx01);
                    GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
                    gridBagConstraintsx02.gridx = 0;
                    gridBagConstraintsx02.gridy = 2;
                    gridBagConstraintsx02.insets = new Insets(5,5,5,5);
                    p.add(main1, gridBagConstraintsx02);
                    this.getContentPane().add(p);
                    setVisible(true);
                }
                catch(Exception ex)
                {
                    System.out.println(ex.getMessage());
                }
                //-----------------------------------------------------------MySet Application---
----//
            }
            public void actionPerformed(ActionEvent ae)
            {
                       String str = ae.getActionCommand();
                       if(str.equals("Design Repository Menu"))
                        {
                            this.dispose();
                            main1 mm = new main1();
                            mm.Gen1 ();
                        }
                        else
                        {
                            int countn = tp.getSelectedIndex();
                            countn = countn + 1;
                            int width = jp[countn].getWidth();
                            int height = jp[countn].getHeight();
                            String abc;
                            save[countn].setVisible (false);
                            fld = new FileDialog(this, "Save File As", FileDialog.SAVE);
                            fld.setVisible(true);
                            if(fld.getFile () != null)
                            {
                                abc = fld.getDirectory()+File.separator + fld.getFile()+".png";
                            }
                            else
                                abc = fld.getDirectory() + File.separator + "design.png";
                            BufferedImage image2 = new BufferedImage(width,
height,BufferedImage.TYPE_INT_RGB);
                            Graphics2D g2 = image2.createGraphics();
                            jp[countn].paint(g2);
                            g2.dispose();
                            try
                            {
                                ImageIO.write(image2, "png", new File(abc));
                            }
                            catch(IOException ioe)
                            {
                                System.out.println(ioe.getMessage());
                            }
                        }
            }
            public void itemStateChanged(ItemEvent ie)
            {
                int ub = 0;
                JCheckBox jj = (JCheckBox) ie.getItem ();
                int state = ie.getStateChange();
                if (state == ItemEvent.SELECTED)
                {
                    ub = Integer.parseInt ((String)jj.getText());
                    //System.out.println (step); Prints Step value (no of components + 1)
                    main1(ub,ab1);
                }
```

```
    }
    public void main1(int num,int bb)
    {
        int num1 = num; //Column for Final_Table for which suitability matrix is needed.
        this.dispose();
        MySet mm1 = new MySet();
        mm1.step1(inc, step-1,step-1, num1,bb);
    }
   public static void main(String[] args)
    {
        NewEmpty aa = new NewEmpty();
        aa.new2(1);
    }
}
```

14) QFD1.java

```
//Inputs the QFD Phase - I data to generate the QFD-I matrix
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.sql.*;
import java.net.*;
import javax.imageio.*;
import javax.swing.border.*;
import java.io.*;
import java.awt.event.*;
public class QFD1 extends JFrame implements ActionListener
{
    JPanel p,p1,p2;
    JTextField tcust,temet;
    JLabel h1,h2,cust,emet,newp;
    JTextField[] tf = new JTextField[20];
    JLabel l[] = new JLabel[20];
    JTextField[] tf1 = new JTextField[20];
    JLabel l1[] = new JLabel[20];
    Border lineb,lineb1;
    JButton bdone,main1,qfd1;
    String[] mi = new String[20];
    int[][] dd = new int[10][100];
    public static int ddx,ab1,ddy,num;
    GridBagConstraints[] bag = new GridBagConstraints[100];
    GridBagConstraints[] bag1 = new GridBagConstraints[100];
    GridBagConstraints[] bag2 = new GridBagConstraints[100];
    GridBagConstraints[] bag3 = new GridBagConstraints[100];
    public void init1(String[] moi, int ddx1,int bb,int[][] dd1, int ddy1, int num1)
    {
        mi = moi;
        ddx = ddx1;
        dd = dd1;
        ddy = ddy1;
        num = num1;
        ab1=bb;//Keeps a track of whether 1st or 2nd design alternative.
        p = new JPanel(new GridBagLayout());
        setSize(700,700);
        setResizable(false);
        this.setTitle("Customer Needs/Engineering Metrics");
        p1 = new JPanel(new GridBagLayout());
        p2 = new JPanel(new GridBagLayout());
        lineb = new LineBorder(Color.BLACK);
        lineb1 = new LineBorder(Color.BLACK);
        p1.setBorder (lineb);
        p2.setBorder (lineb1);
        cust = new JLabel("Enter the Number of Customer Needs: ");
        tcust = new JTextField(25);
        emet = new JLabel("Enter the Number of Engineering Metrics: ");
        temet = new JTextField(25);
        bdone = new JButton("Done");
        main1 = new JButton("Design Repository Menu");
        main1.addActionListener(this);
        qfd1 = new JButton("Generate QFD-I Matrix");
        qfd1.addActionListener(this);
        bdone.addActionListener(this);
        newp = new JLabel("Customer Needs & Engineering Metrics");
        newp.setFont(new Font("System",Font.BOLD,25));
        GridBagConstraints gridBagConstraintsx22a = new GridBagConstraints();
        gridBagConstraintsx22a.gridx = 0;
        gridBagConstraintsx22a.gridy = 0;
        gridBagConstraintsx22a.gridwidth = 3;
        gridBagConstraintsx22a.insets = new Insets(5,5,5,5);
        p.add(newp,gridBagConstraintsx22a);
        GridBagConstraints gridBagConstraintsx22 = new GridBagConstraints();
        gridBagConstraintsx22.gridx = 0;
        gridBagConstraintsx22.gridy = 1;
        gridBagConstraintsx22.gridwidth = 1;
        gridBagConstraintsx22.insets = new Insets(5,5,5,5);
        p.add(cust,gridBagConstraintsx22);
        GridBagConstraints gridBagConstraintsx23 = new GridBagConstraints();
        gridBagConstraintsx23.gridx = 1;
        gridBagConstraintsx23.gridy = 1;
        gridBagConstraintsx23.gridwidth = 1;
        gridBagConstraintsx23.insets = new Insets(5,5,5,5);
        p.add(tcust,gridBagConstraintsx23);
        GridBagConstraints gridBagConstraintsx24 = new GridBagConstraints();
        gridBagConstraintsx24.gridx = 0;
        gridBagConstraintsx24.gridy = 2;
        gridBagConstraintsx24.gridwidth = 1;
        gridBagConstraintsx24.insets = new Insets(5,5,5,5);
        p.add(emet,gridBagConstraintsx24);
        GridBagConstraints gridBagConstraintsx25 = new GridBagConstraints();
        gridBagConstraintsx25.gridx = 1;
        gridBagConstraintsx25.gridy = 2;
        gridBagConstraintsx25.gridwidth = 1;
        gridBagConstraintsx25.insets = new Insets(5,5,5,5);
        p.add(temet,gridBagConstraintsx25);
        GridBagConstraints gridBagConstraintsx26 = new GridBagConstraints();
        gridBagConstraintsx26.gridx = 1;
        gridBagConstraintsx26.gridy = 3;
        gridBagConstraintsx26.gridwidth = 1;
        gridBagConstraintsx26.insets = new Insets(5,5,5,5);
        p.add(bdone,gridBagConstraintsx26);
        this.getContentPane().add(p);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent ae)
    {
        int custno = Integer.parseInt(tcust.getText());
        int emetno = Integer.parseInt(temet.getText());
        String str = ae.getActionCommand();
        if(str.equals("Design Repository Menu"))
        {
            this.dispose();
            main1 mm = new main1();
            mm.Gen1 ();
        }
        if(str.equals("Generate QFD-I Matrix"))
        {
            String cn[] = new String[20];
            String em[] = new String[20];
            this.dispose();
            QFD12 jj = new QFD12();
            for(int i=1; i<= custno; i++)
            {
                cn[i] = tf[i].getText ();
            }
            for(int j=1; j<= emetno; j++)
            {
                em[j] = tf1[j].getText ();
            }
            jj.init1 (mi,ddx,custno,emetno,cn,em,ab1,dd,ddy,num);
        }
        if(str.equals("Done"))
        {
            h1 = new JLabel("Input the Customer Needs");
            h1.setFont(new Font("System",Font.BOLD,15));
            GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
            gridBagConstraintsx00.gridx = 0;
            gridBagConstraintsx00.gridy = 0;
            gridBagConstraintsx00.gridwidth = 2;
            gridBagConstraintsx00.insets = new Insets(5,5,5,5);
            p1.add(h1,gridBagConstraintsx00);
            bdone.setVisible(false);
            for(int i = 1; i<= custno; i++)
            {
                l[i] = new JLabel("Enter the Customer Need #" + i + ":");
                tf[i] = new JTextField(25);
                bag[i] = new GridBagConstraints();
                bag[i].gridx = 0;
                bag[i].gridy = i;
                bag[i].gridwidth = 1;
                bag[i].insets = new Insets(5,5,5,5);
                p1.add(l[i], bag[i]);
                bag1[i] = new GridBagConstraints();
                bag1[i].gridx = 1;
                bag1[i].gridy = i;
                bag1[i].insets = new Insets(5,5,5,5);
                bag1[i].gridwidth = 1;
                bag1[i].fill = GridBagConstraints.BOTH;
                p1.add(tf[i], bag1[i]);
            }
            h2 = new JLabel("Input the Engineering Metrics");
            h2.setFont(new Font("System",Font.BOLD,15));
            GridBagConstraints gridBagConstraintsx011 = new GridBagConstraints();
            gridBagConstraintsx011.gridx = 0;
            gridBagConstraintsx011.gridy = 0;
            gridBagConstraintsx011.gridwidth = 2;
            gridBagConstraintsx011.insets = new Insets(5,5,5,5);
            p2.add(h2,gridBagConstraintsx011);
            for(int j = 1; j<= emetno; j++)
            {
                l1[j] = new JLabel("Enter the Engineering Metric #" + j + ":");
                tf1[j] = new JTextField(25);
                bag2[j] = new GridBagConstraints();
                bag2[j].gridx = 0;
                bag2[j].gridy = j;
                bag2[j].gridwidth = 1;
                bag2[j].insets = new Insets(5,5,5,5);
                p2.add(l1[j], bag2[j]);
                bag3[j] = new GridBagConstraints();
                bag3[j].gridx = 1;
                bag3[j].gridy = j;
                bag3[j].insets = new Insets(5,5,5,5);
                bag3[j].gridwidth = 1;
                bag3[j].fill = GridBagConstraints.BOTH;
                p2.add(tf1[j], bag3[j]);
            }
            GridBagConstraints gridBagConstraintsx26 = new GridBagConstraints();
            gridBagConstraintsx26.gridx = 0;
            gridBagConstraintsx26.gridy = 3;
            gridBagConstraintsx26.gridwidth = 2;
            gridBagConstraintsx26.insets = new Insets(5,5,5,5);
            p.add(p1,gridBagConstraintsx26);
            GridBagConstraints gridBagConstraintsx27 = new GridBagConstraints();
            gridBagConstraintsx27.gridx = 0;
            gridBagConstraintsx27.gridy = 4;
            gridBagConstraintsx27.gridwidth = 2;
            gridBagConstraintsx27.insets = new Insets(5,5,5,5);
            p.add(p2,gridBagConstraintsx27);
            GridBagConstraints gridBagConstraintsx28 = new GridBagConstraints();
            gridBagConstraintsx28.gridx = 0;
            gridBagConstraintsx28.gridy = 5;
            gridBagConstraintsx28.gridwidth = 1;
            gridBagConstraintsx28.insets = new Insets(5,5,5,5);
            p.add(qfd1,gridBagConstraintsx28);
            GridBagConstraints gridBagConstraintsx29 = new GridBagConstraints();
            gridBagConstraintsx29.gridx = 1;
            gridBagConstraintsx29.gridy = 5;
            gridBagConstraintsx29.gridwidth = 1;
            gridBagConstraintsx29.insets = new Insets(5,5,5,5);
            p.add(main1,gridBagConstraintsx29);
        }
    }
}
```

15) QFD2.java

```
        // Inputs the QFD Phase - II data to generate the QFD-II matrix

        import javax.swing.*;
        import java.util.*;
        import java.awt.*;
        import java.awt.image.BufferedImage;
        import java.sql.*;
        import java.net.*;
        import javax.imageio.*;
        import javax.swing.border.*;
        import java.io.*;
        import java.awt.event.*;
        public class QFD2 extends JFrame implements
ActionListener,MouseMotionListener,MouseListener
        {
            JPanel p,p1;
            JTextField tf;
            int l=0,n=4,col=0;
            BufferedImage imageo;
            ImageIcon image1;
            JButton main,dfv;
            JLabel[] img = new JLabel[100];
            JLabel[] img2 = new JLabel[100];
            JLabel[][] lbl = new JLabel[10][10];
            String[][] abb = new String[10][10];
            JPanel[][] ai = new JPanel[10][10];
            JComboBox[] wt = new JComboBox[10];
            Border lineb,linec;
            JLabel newp;
            String[] em1 = new String[20];
            int[] change = new int[20];
            int count,ddx,ddy,ab1,num;
            String[] modul = new String[20];
            int[][] dd = new int[10][100];
            String driver,url;
            Connection conn;
            Statement stmt,stmt1;
            ResultSet rs,rse;
            public void init1(String[] mod, int ddx1,int bb,int[][] dd1,int ddy1,int num1)
            {
                modul = mod;
                ddx = ddx1;
                ab1 = bb;//Keeps a track of whether 1st or 2nd design alternative.
                dd = dd1;
                ddy = ddy1;
                num = num1;
                try
                {
                    driver = "com.mysql.jdbc.Driver";
                    url = "jdbc:mysql://localhost:3306/test";
                    Class.forName(driver);
                    conn =DriverManager.getConnection(url,"root","pennstate");
                    stmt = conn.createStatement();
                    rs = stmt.executeQuery("Select * from qfd1;");
                    while(rs.next())
                    {
                        col++;
                        em1[col] = rs.getString ("em1");
                        change[col] = rs.getInt ("chang");
                    }
                    stmt.close();
                    conn.close();
                }
                catch(Exception ex)
                {
                    System.out.println(ex);
                }
                for(int i =1; i<=ddx1; i++) //To find out number of modules;
                {
                     if(!modul[i].equals("")) count++;
                }
                p1 = new JPanel(new GridBagLayout());
                this.setSize(700,700);
                p = new JPanel();
```

```
p.setLayout(new GridLayout(col+1,count+1));
p.setForeground(Color.BLUE);
this.setTitle("QFD-II Matrix");
lineb = new LineBorder(Color.gray);
linec = new LineBorder(Color.BLACK);
main = new JButton("Design Repository Menu");
main.addActionListener(this);
dfv = new JButton("Generate DFV Index");
dfv.addActionListener (this);
for(int i=1; i<=count; i++)
{
    img[i] = new JLabel(modul[i]);
    img[i].setFont(new Font("System",Font.BOLD,10));
}
for(int j=1; j<=col; j++)
{
    img2[j] = new JLabel(em1[j]);
    img2[j].setFont(new Font("System",Font.BOLD,10));
}
try
{
    ai[1][1] = new JPanel();
    p.add(ai[1][1]);
    for(int m=2;m<=count+1;m++)
    {
        ai[1][m] = new JPanel();
        ai[1][m].add(img[m-1]);
        p.add (ai[1][m]);
    }
    for(int i = 2; i<=col+1; i++)
    {
        for(int j=1;j<=count+1;j++)
        {
            if(j==1)
            {
                ai[i][1] = new JPanel();
                ai[i][1].add(img2[i-1]);
                p.add (ai[i][1]);
            }
            else
            {
                ai[i][j] = new JPanel();
                ai[i][j].setBorder (lineb);
                lbl[i][j] = new JLabel();
                lbl[i][j].setText (" ");
                ai[i][j].add (lbl[i][j]);
                ai[i][j].addMouseMotionListener(this);
                ai[i][j].addMouseListener(this);
                ai[i][j].setForeground(Color.WHITE);
                p.add(ai[i][j]);
            }
        }
    }
    newp = new JLabel("QFD-II Matrix");
    newp.setFont(new Font("System",Font.BOLD,25));
    GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
    gridBagConstraintsx00.gridx = 0;
    gridBagConstraintsx00.gridy = 0;
    gridBagConstraintsx00.gridwidth = 2;
    gridBagConstraintsx00.insets = new Insets(5,5,5,5);
    p1.add(newp,gridBagConstraintsx00);
    GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
    gridBagConstraintsx01.gridx = 0;
    gridBagConstraintsx01.gridy = 1;
    gridBagConstraintsx01.gridwidth = 2;
    gridBagConstraintsx01.insets = new Insets(5,5,5,5);
    p1.add(p, gridBagConstraintsx01);
    GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
    gridBagConstraintsx02.gridx = 0;
    gridBagConstraintsx02.gridy = 2;
    gridBagConstraintsx02.insets = new Insets(5,5,5,5);
    gridBagConstraintsx02.gridwidth = 1;
    gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
    p1.add(dfv, gridBagConstraintsx02);
    GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
    gridBagConstraintsx03.gridx = 1;
```

```
        gridBagConstraintsx03.gridy = 2;
        gridBagConstraintsx03.insets = new Insets(5,5,5,5);
        gridBagConstraintsx03.gridwidth = 1;
        gridBagConstraintsx03.fill = GridBagConstraints.BOTH;
        p1.add(main, gridBagConstraintsx03);
        this.getContentPane().add(p1);
        setVisible(true);
    }
    catch (Exception ex)
    {
        System.out.println(ex);
    }
}
public void mouseMoved(MouseEvent e)
{
   for(int i1=2;i1<=col+1;i1++)
   {
     for(int j1=2;j1<=count+1;j1++)
      {
            if(e.getComponent () == ai[i1][j1])
            {
                ai[i1][j1].setBorder(linec);
                p.revalidate();
            }
            else
            {
                ai[i1][j1].setBorder(lineb);
                p.revalidate();
            }
       }
   }
 }
public void actionPerformed(ActionEvent ae)
{
    String str = ae.getActionCommand();
    if(str.equals("Design Repository Menu"))
    {
        this.dispose();
        main1 mm = new main1();
        mm.Gen1 ();
    }
    if(str.equals("Generate DFV Index"))
    {
         for(int i1=1;i1<=col;i1++)
         {
            for(int j1=1;j1<=count;j1++)
            {
               abb[i1][j1] = lbl[i1+1][j1+1].getText();
                if(abb[i1][j1].equals("X"))
                abb[i1][j1] = Integer.toString(change[i1]);
                else abb[i1][j1] = Integer.toString (0);
                System.out.print(abb[i1][j1] + " ");
            }
            System.out.print ("\n");
         }
        this.dispose();
        DFV ddxy = new DFV();
        ddxy.init1 (modul,ddx,col,em1,change,abb,ab1,dd,ddy,num);
     }
}
public void mouseDragged(MouseEvent e)
{
   //System.out.println("Mouse dragged");
}
public void mousePressed(MouseEvent e)
{
   //System.out.println("Mouse Pressed");
}
public void mouseReleased(MouseEvent e)
{
   //System.out.println("Mouse Released");
}
public void mouseEntered(MouseEvent e)
{
   //System.out.println("Mouse Entered");
}
```

```
    public void mouseExited(MouseEvent e)
    {
       //System.out.println ("Mouse Exited");
    }
    public void mouseClicked(MouseEvent e)
    {
       for(int i1=2;i1<=col+1;i1++)
       {
         for(int j1=2;j1<=count+1;j1++)
          {
                if(e.getComponent () == ai[i1][j1])
                {
                    String abb = lbl[i1][j1].getText();
                    if(abb.equals(" "))
                    {
                        lbl[i1][j1].setText ("X");
                        lbl[i1][j1].setFont(new Font("System",Font.BOLD,18));
                    }
                    else
                        lbl[i1][j1].setText (" ");
                    p.revalidate();
                }
           }
       }
    }
 }
```

16) QFD12.java

```
        //Generates the QFD matrix from QFD1.java
        import javax.swing.*;
        import java.util.*;
        import java.awt.*;
        import java.awt.image.BufferedImage;
        import java.sql.*;
        import java.net.*;
        import javax.imageio.*;
        import javax.swing.border.*;
        import java.io.*;
        import java.awt.event.*;
        public class QFD12 extends JFrame implements
ActionListener,MouseMotionListener,MouseListener
        {
            JPanel p,p1;
            JTextField tf;
            int l=0,n=4;
            BufferedImage imageo;
            JButton main1,qfd2;
            ImageIcon image1;
            JLabel[] img = new JLabel[100];
            JLabel[] img2 = new JLabel[100];
            JLabel[][] lbl = new JLabel[10][10];
            JPanel[][] ai = new JPanel[10][10];
            JComboBox[] wt = new JComboBox[10];
            String[] modul = new String[20];
            Border lineb,linec;
            JLabel newp;
            public static int row,col,ddx=0,ddy,num,ab1;
            int[][] dd = new int[10][100];
            String cn1[] = new String[20];
            String em1[] = new String[20];
            public void init1(String[] mod, int ddx1,int x, int y, String[] cn, String[] em,int
bb,int[][] dd1,int ddy1,int num1)
            {
                ddx = ddx1;
                row = x;
                col = y;
                modul = mod;
                cn1 = cn;
                em1 = em;
                ab1=bb;//Keeps a track of whether 1st or 2nd design alternative.
                dd = dd1;
                ddy = ddy1;
                num = num1;
                p1 = new JPanel(new GridBagLayout());
                this.setSize(700,700);
```

```
//setResizable(false);
p = new JPanel();
p.setLayout(new GridLayout(row+1,col+2));
p.setForeground(Color.BLUE);
this.setTitle("QFD-I Matrix");
lineb = new LineBorder(Color.gray);
linec = new LineBorder(Color.BLACK);
main1 = new JButton("Design Repository Menu");
main1.addActionListener (this);
qfd2 = new JButton("Generate QFD-II Matrix");
qfd2.addActionListener(this);
for(int i = 1; i<= row; i++)
{
    img2[i] = new JLabel(cn[i]);
    img2[i].setFont(new Font("System",Font.BOLD,10));
}
for(int j = 1; j<= col; j++)
{
    img[j] = new JLabel(em[j]);
    img[j].setFont(new Font("System",Font.BOLD,10));
}
try
{
    ai[1][1] = new JPanel();
    p.add(ai[1][1]);
    for(int m=2;m<=col+1;m++)
    {
        ai[1][m] = new JPanel();
        ai[1][m].add(img[m-1]);
        p.add (ai[1][m]);
    }
    ai[1][col+2] = new JPanel();
    ai[1][col+2].add(new JLabel("Expected Change"));
    p.add (ai[1][col+2]);
    for(int i = 2; i<=row+1; i++)
    {
        for(int j=1;j<=col+2;j++)
        {
            if(j==1)
            {
                ai[i][1] = new JPanel();
                ai[i][1].add(img2[i-1]);
                p.add (ai[i][1]);
            }
            else if(j==col+2)
            {
                ai[i][j] = new JPanel();
                wt[i] = new JComboBox();
                wt[i].addItem("High");
                wt[i].addItem("Medium");
                wt[i].addItem("Low");
                ai[i][j].add(wt[i]);
                p.add (ai[i][j]);
            }
            else
            {
                ai[i][j] = new JPanel();
                ai[i][j].setBorder (lineb);
                lbl[i][j] = new JLabel();
                lbl[i][j].setText (" ");
                ai[i][j].add (lbl[i][j]);
                ai[i][j].addMouseMotionListener(this);
                ai[i][j].addMouseListener(this);
                ai[i][j].setForeground(Color.WHITE);
                p.add(ai[i][j]);
            }
        }
    }
    newp = new JLabel("QFD-I Matrix");
    newp.setFont(new Font("System",Font.BOLD,25));
    GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
    gridBagConstraintsx00.gridx = 0;
    gridBagConstraintsx00.gridy = 0;
    gridBagConstraintsx00.gridwidth = 2;
    gridBagConstraintsx00.insets = new Insets(5,5,5,5);
    p1.add(newp,gridBagConstraintsx00);
```

```
        GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
        gridBagConstraintsx01.gridx = 0;
        gridBagConstraintsx01.gridy = 1;
        gridBagConstraintsx01.gridwidth = 2;
        gridBagConstraintsx01.insets = new Insets(5,5,5,5);
        p1.add(p, gridBagConstraintsx01);
        GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
        gridBagConstraintsx02.gridx = 0;
        gridBagConstraintsx02.gridy = 2;
        gridBagConstraintsx02.insets = new Insets(5,5,5,5);
        gridBagConstraintsx02.gridwidth = 1;
        gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
        p1.add(main1, gridBagConstraintsx02);
        GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
        gridBagConstraintsx03.gridx = 1;
        gridBagConstraintsx03.gridy = 2;
        gridBagConstraintsx03.insets = new Insets(5,5,5,5);
        gridBagConstraintsx03.gridwidth = 1;
        gridBagConstraintsx03.fill = GridBagConstraints.BOTH;
        p1.add(qfd2, gridBagConstraintsx03);
        this.getContentPane().add(p1);
        setVisible(true);
    }
    catch (Exception ex)
    {
        System.out.println(ex);
    }
}
public void mouseMoved(MouseEvent e)
{
   for(int i1=2;i1<=row+1;i1++)
   {
     for(int j1=2;j1<=col+1;j1++)
      {
            if(e.getComponent () == ai[i1][j1])
            {
                ai[i1][j1].setBorder(linec);
                p.revalidate();
            }
            else
            {
                ai[i1][j1].setBorder(lineb);
                p.revalidate();
            }
       }
   }
 }
public void mouseDragged(MouseEvent e)
{
   //System.out.println("Mouse dragged");
}
public void mousePressed(MouseEvent e)
{
   //System.out.println("Mouse Pressed");
}
public void mouseReleased(MouseEvent e)
{
   //System.out.println("Mouse Released");
}
public void mouseEntered(MouseEvent e)
{
   //System.out.println("Mouse Entered");
}
public void mouseExited(MouseEvent e)
{
   //System.out.println ("Mouse Exited");
}
public void mouseClicked(MouseEvent e)
{
   for(int i1=2;i1<=row+1;i1++)
   {
     for(int j1=2;j1<=col+1;j1++)
      {
            if(e.getComponent () == ai[i1][j1])
            {
                String abb = lbl[i1][j1].getText();
```

```
                            if(abb.equals(" "))
                            {
                                lbl[i1][j1].setText ("X");
                                lbl[i1][j1].setFont(new Font("System",Font.BOLD,20));
                            }
                            else
                                lbl[i1][j1].setText (" ");
                            p.revalidate();
                        }
                   }
               }
            }
            public void actionPerformed(ActionEvent ae)
            {
                String driver,url;
                Connection conn;
                Statement stmt,stmt1;
                ResultSet rs,rse;
                String str = ae.getActionCommand();
                int[] chan_em = new int[50];
                if(str.equals("Design Repository Menu"))
                {
                    this.dispose();
                    main1 mm = new main1();
                    mm.Gen1 ();
                }
                if(str.equals("Generate QFD-II Matrix"))
                {
                    int chang[] = new int[20];
                    for(int r = 1; r<= row; r++)
                    {
                        chang[r] = 3 - (wt[r+1].getSelectedIndex ());
                    }
                    for(int i1=2;i1<=row+1;i1++)
                    {
                        for(int j1=2;j1<=col+1;j1++)
                        {
                             String abb = lbl[i1][j1].getText();
                             if(abb.equals("X"))
                             {
                                chan_em[j1-1]+=chang[i1-1];
                             }
                        }
                    }
                    try
                    {
                        driver = "com.mysql.jdbc.Driver";
                        url = "jdbc:mysql://localhost:3306/test";
                        Class.forName(driver);
                        conn =DriverManager.getConnection(url,"root","pennstate");
                        stmt = conn.createStatement();
                        stmt.executeUpdate ("drop table if exists qfd1;");
                        stmt.executeUpdate("create table qfd1(em1 varchar(50),chang int(2));");
                        for(int i3=1;i3<=col;i3++)
                        {
                            stmt.executeUpdate("insert into qfd1 (em1,chang) values(\'" + em1[i3]
+ "\',"+ chan_em[i3] + ");");
                        }
                        stmt.close();
                        conn.close();
                    }
                    catch(Exception ex)
                    {
                        System.out.println(ex);
                    }
                    this.dispose();
                    QFD2 kk = new QFD2();
                    kk.init1 (modul,ddx,ab1,dd,ddy,num);
                }
            }
        }
```

17) rules.java

```java
//Inputs the graph grammar rules in the design repository
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.*;

public class rules extends JFrame implements ActionListener
{
        JTextField baseid,rec;
        JComboBox rul1in,rul2in,rul3in,rul1sub,rul2sub,rul3sub,rul1out,rul2out,rul3out;
        JLabel inpu,subf,outpu,baseid1,and1,and2,jLabel0;
        JButton inser,main1;
        int rul=0,record=0;
        Connection conn;
        Statement stmt;
        ResultSet rset;
        String a, driver,url;
        FileDialog dialog1;
        public void init1()
        {
                JPanel p = new JPanel();
                 setSize(700,700);
                 p.setLayout(new GridBagLayout());
                 setResizable(false);
                 this.setTitle("Input Rule(s) Menu");
                 jLabel0 = new JLabel("Input Rule(s) Menu");
                 jLabel0.setFont(new Font("System",Font.BOLD,25));
                 try
                {
                        driver = "com.mysql.jdbc.Driver";
                        url = "jdbc:mysql://localhost:3306/test";
                        Class.forName(driver);
                        conn =DriverManager.getConnection(url,"root","pennstate");
                        stmt = conn.createStatement();
                        rset = stmt.executeQuery("Select * from rules");
                        while (rset.next())
                        {
                                record++;
                        }
                }
                catch(Exception ex)
                {
                        System.out.println(ex);
                        return;
                }
                inser = new JButton("Insert into Repository");
                inser.addActionListener(this);
                main1 = new JButton("Design Repository Menu");
                main1.addActionListener(this);
                inpu = new JLabel("Input Flow");
                subf = new JLabel("Sub-Function");
                outpu = new JLabel("Output Flow");
                baseid1 = new JLabel("Base ID: ");
                and1 = new JLabel("AND");
                and2 = new JLabel("AND");
                baseid = new JTextField(15);
                rec = new JTextField(15);
                rec.setText("Rule(s) in Repository: " + Integer.toString(record));
                rec.setEditable(false);

                rul1in = new JComboBox();
                rul1in.addItem("");
                rul1in.addItem("Acoustic Energy");
                 rul1in.addItem("Analog Control");
                 rul1in.addItem("Auditory");
                rul1in.addItem("Biological Energy");
                rul1in.addItem("Chemical Energy");
                 rul1in.addItem("Colloidal");
                 rul1in.addItem("Composite");
                 rul1in.addItem("Discrete Control");
                rul1in.addItem("Electrical Energy");
                rul1in.addItem("Electromagnetic Energy");
                 rul1in.addItem("Gas");
```

```
 rul1in.addItem("Gas-Gas Mixture");
rul1in.addItem("Human Energy");
rul1in.addItem("Human Material");
rul1in.addItem("Hydraulic Energy");
 rul1in.addItem("Liquid");
rul1in.addItem("Liquid-Gas");
 rul1in.addItem("Liquid-Liquid");
rul1in.addItem("Magnetic Energy");
rul1in.addItem("Mechanical Energy");
rul1in.addItem("Object");
 rul1in.addItem("Olfactory");
rul1in.addItem("Optical Energy");
 rul1in.addItem("Particulate");
 rul1in.addItem("Plasma");
rul1in.addItem("Pneumatic Energy");
rul1in.addItem("Radioactive/Nuclear");
rul1in.addItem("Rotational Energy");
rul1in.addItem("Solar Energy");
 rul1in.addItem("Solid");
 rul1in.addItem("Solid-Gas");
rul1in.addItem("Solid-Liquid");
 rul1in.addItem("Solid-Liquid-Gas");
rul1in.addItem("Solid-Solid");
 rul1in.addItem("Thermal Energy");
 rul1in.addItem("Translational Energy");
 rul1in.addItem("Tactile");
rul1in.addItem("Taste");
 rul1in.addItem("Visual");

 rul2in = new JComboBox();
rul2in.addItem("");
rul2in.addItem("Acoustic Energy");
 rul2in.addItem("Analog Control");
 rul2in.addItem("Auditory");
rul2in.addItem("Biological Energy");
rul2in.addItem("Chemical Energy");
 rul2in.addItem("Colloidal");
 rul2in.addItem("Composite");
 rul2in.addItem("Discrete Control");
rul2in.addItem("Electrical Energy");
rul2in.addItem("Electromagnetic Energy");
 rul2in.addItem("Gas");
 rul2in.addItem("Gas-Gas Mixture");
rul2in.addItem("Human Energy");
rul2in.addItem("Human Material");
rul2in.addItem("Hydraulic Energy");
 rul2in.addItem("Liquid");
rul2in.addItem("Liquid-Gas");
 rul2in.addItem("Liquid-Liquid");
rul2in.addItem("Magnetic Energy");
rul2in.addItem("Mechanical Energy");
rul2in.addItem("Object");
 rul2in.addItem("Olfactory");
rul2in.addItem("Optical Energy");
 rul2in.addItem("Particulate");
 rul2in.addItem("Plasma");
rul2in.addItem("Pneumatic Energy");
rul2in.addItem("Radioactive/Nuclear");
rul2in.addItem("Rotational Energy");
rul2in.addItem("Solar Energy");
 rul2in.addItem("Solid");
 rul2in.addItem("Solid-Gas");
rul2in.addItem("Solid-Liquid");
 rul2in.addItem("Solid-Liquid-Gas");
rul2in.addItem("Solid-Solid");
 rul2in.addItem("Thermal Energy");
 rul2in.addItem("Translational Energy");
 rul2in.addItem("Tactile");
rul2in.addItem("Taste");
 rul2in.addItem("Visual");

 rul3in = new JComboBox();
rul3in.addItem("");
rul3in.addItem("Acoustic Energy");
 rul3in.addItem("Analog Control");
 rul3in.addItem("Auditory");
```

```
rul3in.addItem("Biological Energy");
rul3in.addItem("Chemical Energy");
 rul3in.addItem("Colloidal");
 rul3in.addItem("Composite");
 rul3in.addItem("Discrete Control");
rul3in.addItem("Electrical Energy");
rul3in.addItem("Electromagnetic Energy");
 rul3in.addItem("Gas");
 rul3in.addItem("Gas-Gas Mixture");
rul3in.addItem("Human Energy");
rul3in.addItem("Human Material");
rul3in.addItem("Hydraulic Energy");
 rul3in.addItem("Liquid");
rul3in.addItem("Liquid-Gas");
 rul3in.addItem("Liquid-Liquid");
rul3in.addItem("Magnetic Energy");
rul3in.addItem("Mechanical Energy");
rul3in.addItem("Object");
 rul3in.addItem("Olfactory");
rul3in.addItem("Optical Energy");
 rul3in.addItem("Particulate");
 rul3in.addItem("Plasma");
rul3in.addItem("Pneumatic Energy");
rul3in.addItem("Radioactive/Nuclear");
rul3in.addItem("Rotational Energy");
rul3in.addItem("Solar Energy");
 rul3in.addItem("Solid");
 rul3in.addItem("Solid-Gas");
rul3in.addItem("Solid-Liquid");
 rul3in.addItem("Solid-Liquid-Gas");
rul3in.addItem("Solid-Solid");
 rul3in.addItem("Thermal Energy");
 rul3in.addItem("Translational Energy");
 rul3in.addItem("Tactile");
rul3in.addItem("Taste");
 rul3in.addItem("Visual");

rul1out = new JComboBox();
rul1out.addItem("");
rul1out.addItem("Acoustic Energy");
 rul1out.addItem("Analog Control");
 rul1out.addItem("Auditory");
rul1out.addItem("Biological Energy");
rul1out.addItem("Chemical Energy");
 rul1out.addItem("Colloidal");
 rul1out.addItem("Composite");
 rul1out.addItem("Discrete Control");
rul1out.addItem("Electrical Energy");
rul1out.addItem("Electromagnetic Energy");
 rul1out.addItem("Gas");
 rul1out.addItem("Gas-Gas Mixture");
rul1out.addItem("Human Energy");
rul1out.addItem("Human Material");
rul1out.addItem("Hydraulic Energy");
 rul1out.addItem("Liquid");
rul1out.addItem("Liquid-Gas");
 rul1out.addItem("Liquid-Liquid");
rul1out.addItem("Magnetic Energy");
rul1out.addItem("Mechanical Energy");
rul1out.addItem("Object");
 rul1out.addItem("Olfactory");
rul1out.addItem("Optical Energy");
 rul1out.addItem("Particulate");
 rul1out.addItem("Plasma");
rul1out.addItem("Pneumatic Energy");
rul1out.addItem("Radioactive/Nuclear");
rul1out.addItem("Rotational Energy");
rul1out.addItem("Solar Energy");
 rul1out.addItem("Solid");
 rul1out.addItem("Solid-Gas");
rul1out.addItem("Solid-Liquid");
 rul1out.addItem("Solid-Liquid-Gas");
rul1out.addItem("Solid-Solid");
 rul1out.addItem("Thermal Energy");
 rul1out.addItem("Translational Energy");
 rul1out.addItem("Tactile");
```

```
rul1out.addItem("Taste");
 rul1out.addItem("Visual");

 rul2out = new JComboBox();
rul2out.addItem("");
rul2out.addItem("Acoustic Energy");
 rul2out.addItem("Analog Control");
 rul2out.addItem("Auditory");
rul2out.addItem("Biological Energy");
rul2out.addItem("Chemical Energy");
 rul2out.addItem("Colloidal");
 rul2out.addItem("Composite");
 rul2out.addItem("Discrete Control");
rul2out.addItem("Electrical Energy");
rul2out.addItem("Electromagnetic Energy");
 rul2out.addItem("Gas");
 rul2out.addItem("Gas-Gas Mixture");
rul2out.addItem("Human Energy");
rul2out.addItem("Human Material");
rul2out.addItem("Hydraulic Energy");
 rul2out.addItem("Liquid");
rul2out.addItem("Liquid-Gas");
 rul2out.addItem("Liquid-Liquid");
rul2out.addItem("Magnetic Energy");
rul2out.addItem("Mechanical Energy");
rul2out.addItem("Object");
 rul2out.addItem("Olfactory");
rul2out.addItem("Optical Energy");
 rul2out.addItem("Particulate");
 rul2out.addItem("Plasma");
rul2out.addItem("Pneumatic Energy");
rul2out.addItem("Radioactive/Nuclear");
rul2out.addItem("Rotational Energy");
rul2out.addItem("Solar Energy");
 rul2out.addItem("Solid");
 rul2out.addItem("Solid-Gas");
rul2out.addItem("Solid-Liquid");
 rul2out.addItem("Solid-Liquid-Gas");
rul2out.addItem("Solid-Solid");
 rul2out.addItem("Thermal Energy");
 rul2out.addItem("Translational Energy");
 rul2out.addItem("Tactile");
rul2out.addItem("Taste");
 rul2out.addItem("Visual");

 rul3out = new JComboBox();
rul3out.addItem("");
rul3out.addItem("Acoustic Energy");
 rul3out.addItem("Analog Control");
 rul3out.addItem("Auditory");
rul3out.addItem("Biological Energy");
rul3out.addItem("Chemical Energy");
 rul3out.addItem("Colloidal");
 rul3out.addItem("Composite");
 rul3out.addItem("Discrete Control");
rul3out.addItem("Electrical Energy");
rul3out.addItem("Electromagnetic Energy");
 rul3out.addItem("Gas");
 rul3out.addItem("Gas-Gas Mixture");
rul3out.addItem("Human Energy");
rul3out.addItem("Human Material");
rul3out.addItem("Hydraulic Energy");
 rul3out.addItem("Liquid");
rul3out.addItem("Liquid-Gas");
 rul3out.addItem("Liquid-Liquid");
rul3out.addItem("Magnetic Energy");
rul3out.addItem("Mechanical Energy");
rul3out.addItem("Object");
 rul3out.addItem("Olfactory");
rul3out.addItem("Optical Energy");
 rul3out.addItem("Particulate");
 rul3out.addItem("Plasma");
rul3out.addItem("Pneumatic Energy");
rul3out.addItem("Radioactive/Nuclear");
rul3out.addItem("Rotational Energy");
rul3out.addItem("Solar Energy");
```

```
 rul3out.addItem("Solid");
 rul3out.addItem("Solid-Gas");
rul3out.addItem("Solid-Liquid");
 rul3out.addItem("Solid-Liquid-Gas");
rul3out.addItem("Solid-Solid");
 rul3out.addItem("Thermal Energy");
 rul3out.addItem("Translational Energy");
 rul3out.addItem("Tactile");
rul3out.addItem("Taste");
 rul3out.addItem("Visual");

rul1sub = new JComboBox();
rul1sub.addItem("");
rul1sub.addItem("Actuate");
 rul1sub.addItem("Allow DOF");
 rul1sub.addItem("Branch");
rul1sub.addItem("Change");
 rul1sub.addItem("Channel");
rul1sub.addItem("Collect");
 rul1sub.addItem("Condition");
 rul1sub.addItem("Connect");
rul1sub.addItem("Contain");
 rul1sub.addItem("Control Magnitude");
 rul1sub.addItem("Convert");
rul1sub.addItem("Couple");
rul1sub.addItem("Decrease");
rul1sub.addItem("Decrement");
rul1sub.addItem("Detect");
rul1sub.addItem("Display");
 rul1sub.addItem("Distribute");
 rul1sub.addItem("Divide");
 rul1sub.addItem("Export");
 rul1sub.addItem("Extract");
 rul1sub.addItem("Guide");
 rul1sub.addItem("Import");
 rul1sub.addItem("Increase");
rul1sub.addItem("Increment");
rul1sub.addItem("Indicate");
rul1sub.addItem("Inhibit");
rul1sub.addItem("Join");
rul1sub.addItem("Link");
 rul1sub.addItem("Measure");
rul1sub.addItem("Mix");
 rul1sub.addItem("Position");
 rul1sub.addItem("Prevent");
rul1sub.addItem("Process");
rul1sub.addItem("Provision");
 rul1sub.addItem("Regulate");
rul1sub.addItem("Remove");
 rul1sub.addItem("Rotate");
rul1sub.addItem("Secure");
 rul1sub.addItem("Sense");
 rul1sub.addItem("Seperate");
 rul1sub.addItem("Shape");
 rul1sub.addItem("Signal");
 rul1sub.addItem("Stabilize");
 rul1sub.addItem("Stop");
 rul1sub.addItem("Store");
 rul1sub.addItem("Supply");
 rul1sub.addItem("Support");
 rul1sub.addItem("Track");
 rul1sub.addItem("Transfer");
rul1sub.addItem("Translate");
 rul1sub.addItem("Transmit");
 rul1sub.addItem("Transport");

rul2sub = new JComboBox();
rul2sub.addItem("");
rul2sub.addItem("Actuate");
 rul2sub.addItem("Allow DOF");
 rul2sub.addItem("Branch");
rul2sub.addItem("Change");
 rul2sub.addItem("Channel");
rul2sub.addItem("Collect");
 rul2sub.addItem("Condition");
 rul2sub.addItem("Connect");
```

```
rul2sub.addItem("Contain");
 rul2sub.addItem("Control Magnitude");
 rul2sub.addItem("Convert");
rul2sub.addItem("Couple");
rul2sub.addItem("Decrease");
rul2sub.addItem("Decrement");
rul2sub.addItem("Detect");
rul2sub.addItem("Display");
 rul2sub.addItem("Distribute");
 rul2sub.addItem("Divide");
 rul2sub.addItem("Export");
 rul2sub.addItem("Extract");
 rul2sub.addItem("Guide");
 rul2sub.addItem("Import");
 rul2sub.addItem("Increase");
rul2sub.addItem("Increment");
rul2sub.addItem("Indicate");
rul2sub.addItem("Inhibit");
rul2sub.addItem("Join");
rul2sub.addItem("Link");
 rul2sub.addItem("Measure");
rul2sub.addItem("Mix");
 rul2sub.addItem("Position");
 rul2sub.addItem("Prevent");
rul2sub.addItem("Process");
rul2sub.addItem("Provision");
 rul2sub.addItem("Regulate");
rul2sub.addItem("Remove");
 rul2sub.addItem("Rotate");
rul2sub.addItem("Secure");
 rul2sub.addItem("Sense");
 rul2sub.addItem("Seperate");
 rul2sub.addItem("Shape");
 rul2sub.addItem("Signal");
 rul2sub.addItem("Stabilize");
 rul2sub.addItem("Stop");
 rul2sub.addItem("Store");
 rul2sub.addItem("Supply");
 rul2sub.addItem("Support");
 rul2sub.addItem("Track");
 rul2sub.addItem("Transfer");
rul2sub.addItem("Translate");
 rul2sub.addItem("Transmit");
 rul2sub.addItem("Transport");

rul3sub = new JComboBox();
rul3sub.addItem("");
rul3sub.addItem("Actuate");
 rul3sub.addItem("Allow DOF");
 rul3sub.addItem("Branch");
rul3sub.addItem("Change");
 rul3sub.addItem("Channel");
rul3sub.addItem("Collect");
 rul3sub.addItem("Condition");
 rul3sub.addItem("Connect");
rul3sub.addItem("Contain");
 rul3sub.addItem("Control Magnitude");
 rul3sub.addItem("Convert");
rul3sub.addItem("Couple");
rul3sub.addItem("Decrease");
rul3sub.addItem("Decrement");
rul3sub.addItem("Detect");
rul3sub.addItem("Display");
 rul3sub.addItem("Distribute");
 rul3sub.addItem("Divide");
 rul3sub.addItem("Export");
 rul3sub.addItem("Extract");
 rul3sub.addItem("Guide");
 rul3sub.addItem("Import");
 rul3sub.addItem("Increase");
rul3sub.addItem("Increment");
rul3sub.addItem("Indicate");
rul3sub.addItem("Inhibit");
rul3sub.addItem("Join");
rul3sub.addItem("Link");
 rul3sub.addItem("Measure");
```

```
rul3sub.addItem("Mix");
 rul3sub.addItem("Position");
 rul3sub.addItem("Prevent");
rul3sub.addItem("Process");
rul3sub.addItem("Provision");
 rul3sub.addItem("Regulate");
rul3sub.addItem("Remove");
 rul3sub.addItem("Rotate");
rul3sub.addItem("Secure");
 rul3sub.addItem("Sense");
 rul3sub.addItem("Seperate");
 rul3sub.addItem("Shape");
 rul3sub.addItem("Signal");
 rul3sub.addItem("Stabilize");
 rul3sub.addItem("Stop");
 rul3sub.addItem("Store");
 rul3sub.addItem("Supply");
 rul3sub.addItem("Support");
 rul3sub.addItem("Track");
 rul3sub.addItem("Transfer");
rul3sub.addItem("Translate");
 rul3sub.addItem("Transmit");
 rul3sub.addItem("Transport");

setLayout(new GridBagLayout());
 GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
gridBagConstraintsx00.gridx = 0;
gridBagConstraintsx00.gridy = 0;
gridBagConstraintsx00.insets = new Insets(5,5,5,5);
gridBagConstraintsx00.gridwidth = 3;
gridBagConstraintsx00.fill = GridBagConstraints.BOTH;
p.add(jLabel0, gridBagConstraintsx00);
GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
gridBagConstraintsx01.gridx = 0;
gridBagConstraintsx01.gridy = 1;
gridBagConstraintsx01.insets = new Insets(5,5,5,5);
p.add(inpu, gridBagConstraintsx01);
GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
gridBagConstraintsx02.gridx = 1;
gridBagConstraintsx02.gridy = 1;
gridBagConstraintsx02.insets = new Insets(5,5,5,5);
gridBagConstraintsx02.gridwidth = 1;
gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
p.add(subf, gridBagConstraintsx02);
GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
gridBagConstraintsx03.gridx = 2;
gridBagConstraintsx03.gridy = 1;
gridBagConstraintsx03.insets = new Insets(5,5,5,5);
p.add(outpu, gridBagConstraintsx03);

GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
gridBagConstraintsx04.gridx = 0;
gridBagConstraintsx04.gridy = 2;
gridBagConstraintsx04.insets = new Insets(5,5,5,5);
gridBagConstraintsx04.gridwidth = 1;
gridBagConstraintsx04.fill = GridBagConstraints.BOTH;
p.add(rul1in, gridBagConstraintsx04);
GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
gridBagConstraintsx05.gridx = 1;
gridBagConstraintsx05.gridy = 2;
gridBagConstraintsx05.insets = new Insets(5,5,5,5);
gridBagConstraintsx05.gridwidth = 1;
gridBagConstraintsx05.fill = GridBagConstraints.BOTH;
p.add(rul1sub, gridBagConstraintsx05);
GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
gridBagConstraintsx06.gridx = 2;
gridBagConstraintsx06.gridy = 2;
gridBagConstraintsx06.insets = new Insets(5,5,5,5);
p.add(rul1out, gridBagConstraintsx06);

GridBagConstraints gridBagConstraintsx06b = new GridBagConstraints();
gridBagConstraintsx06b.gridx = 1;
gridBagConstraintsx06b.gridy = 3;
gridBagConstraintsx06b.insets = new Insets(5,5,5,5);
gridBagConstraintsx06b.gridwidth = 1;
gridBagConstraintsx06b.fill = GridBagConstraints.BOTH;
```

```
p.add(and1, gridBagConstraintsx06b);

GridBagConstraints gridBagConstraintsx07 = new GridBagConstraints();
gridBagConstraintsx07.gridx = 0;
gridBagConstraintsx07.gridy = 4;
gridBagConstraintsx07.insets = new Insets(5,5,5,5);
gridBagConstraintsx07.gridwidth = 1;
gridBagConstraintsx07.fill = GridBagConstraints.BOTH;
p.add(rul2in, gridBagConstraintsx07);
GridBagConstraints gridBagConstraintsx08 = new GridBagConstraints();
gridBagConstraintsx08.gridx = 1;
gridBagConstraintsx08.gridy = 4;
gridBagConstraintsx08.insets = new Insets(5,5,5,5);
gridBagConstraintsx08.gridwidth = 1;
gridBagConstraintsx08.fill = GridBagConstraints.BOTH;
p.add(rul2sub, gridBagConstraintsx08);
GridBagConstraints gridBagConstraintsx09 = new GridBagConstraints();
gridBagConstraintsx09.gridx = 2;
gridBagConstraintsx09.gridy = 4;
gridBagConstraintsx09.insets = new Insets(5,5,5,5);
p.add(rul2out, gridBagConstraintsx09);

GridBagConstraints gridBagConstraintsx09b = new GridBagConstraints();
gridBagConstraintsx09b.gridx = 1;
gridBagConstraintsx09b.gridy = 5;
gridBagConstraintsx09b.insets = new Insets(5,5,5,5);
gridBagConstraintsx09b.gridwidth = 1;
gridBagConstraintsx09b.fill = GridBagConstraints.BOTH;
p.add(and2, gridBagConstraintsx09b);

GridBagConstraints gridBagConstraintsx010 = new GridBagConstraints();
gridBagConstraintsx010.gridx = 0;
gridBagConstraintsx010.gridy = 6;
gridBagConstraintsx010.insets = new Insets(5,5,5,5);
gridBagConstraintsx010.gridwidth = 1;
gridBagConstraintsx010.fill = GridBagConstraints.BOTH;
p.add(rul3in, gridBagConstraintsx010);
GridBagConstraints gridBagConstraintsx011 = new GridBagConstraints();
gridBagConstraintsx011.gridx = 1;
gridBagConstraintsx011.gridy = 6;
gridBagConstraintsx011.insets = new Insets(5,5,5,5);
gridBagConstraintsx011.gridwidth = 1;
gridBagConstraintsx011.fill = GridBagConstraints.BOTH;
p.add(rul3sub, gridBagConstraintsx011);
GridBagConstraints gridBagConstraintsx012 = new GridBagConstraints();
gridBagConstraintsx012.gridx = 2;
gridBagConstraintsx012.gridy = 6;
gridBagConstraintsx012.insets = new Insets(5,5,5,5);
p.add(rul3out, gridBagConstraintsx012);

GridBagConstraints gridBagConstraintsx013 = new GridBagConstraints();
gridBagConstraintsx013.gridx = 0;
gridBagConstraintsx013.gridy = 7;
gridBagConstraintsx013.insets = new Insets(5,5,5,5);
p.add(baseid1, gridBagConstraintsx013);
GridBagConstraints gridBagConstraintsx014 = new GridBagConstraints();
gridBagConstraintsx014.gridx = 1;
gridBagConstraintsx014.gridy = 7;
gridBagConstraintsx014.insets = new Insets(5,5,5,5);
gridBagConstraintsx014.gridwidth = 1;
gridBagConstraintsx014.fill = GridBagConstraints.BOTH;
p.add(baseid, gridBagConstraintsx014);
GridBagConstraints gridBagConstraintsx015 = new GridBagConstraints();
gridBagConstraintsx015.gridx = 2;
gridBagConstraintsx015.gridy = 7;
gridBagConstraintsx015.insets = new Insets(5,5,5,5);
p.add(inser, gridBagConstraintsx015);

GridBagConstraints gridBagConstraintsx016 = new GridBagConstraints();
gridBagConstraintsx016.gridx = 0;
gridBagConstraintsx016.gridy = 8;
gridBagConstraintsx016.insets = new Insets(5,5,5,5);
gridBagConstraintsx016.gridwidth = 2;
gridBagConstraintsx016.fill = GridBagConstraints.BOTH;
p.add(rec, gridBagConstraintsx016);
GridBagConstraints gridBagConstraintsx017 = new GridBagConstraints();
```

```
                       gridBagConstraintsx017.gridx = 2;
                       gridBagConstraintsx017.gridy = 8;
                       gridBagConstraintsx017.insets = new Insets(5,5,5,5);
                       p.add(main1, gridBagConstraintsx017);
                        this.getContentPane().add(p);
                        setVisible(true);

               }
               public void actionPerformed(ActionEvent ae)
               {
                       String str = ae.getActionCommand();
                       if(str.equals("Insert into Repository"))
                       {
                            int response = JOptionPane.showConfirmDialog (this,"Are you sure you
want to insert into design repository?","Confirm Insert
Dialog",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
                            if(response == JOptionPane.YES_OPTION)
                            {
                                String basid = baseid.getText();
                              baseid.setText("");
                              record = 0;
                                    try
                                    {
                                            rset = stmt.executeQuery("Select * from rules");
                                            while (rset.next())
                                            {
                                                    record++;
                                            }
                                            record=record+1;

//if((rul1in.getSelectedIndex()!=0)&&(rul1sub.getSelectedIndex()!=0)&&(rul1out.getSelectedIndex()
!=0))
                                            stmt.executeUpdate("insert into rules values(" +
record +",\'" + rul1in.getSelectedItem() + "\',\'" + rul2in.getSelectedItem() + "\',\'"+
rul3in.getSelectedItem() + "\',\'"+ rul1sub.getSelectedItem() + "\',\'" +
rul2sub.getSelectedItem()+ "\',\'"+ rul3sub.getSelectedItem()+ "\',\'" +
rul1out.getSelectedItem() + "\',\'" + rul2out.getSelectedItem()+ "\',\'" +
rul3out.getSelectedItem()+"\',\'" + basid+"\');");

//if((rul2in.getSelectedIndex()!=0)&&(rul2sub.getSelectedIndex()!=0)&&(rul2out.getSelectedIndex()
!=0))
                                            //stmt.executeUpdate("insert into rules values(" +
record +",\'" + rul2in.getSelectedItem() + "\',\'" + rul2sub.getSelectedItem() + "\',\'" +
rul2out.getSelectedItem() +"\',\'" + basid+"\');");

//if((rul3in.getSelectedIndex()!=0)&&(rul3sub.getSelectedIndex()!=0)&&(rul3out.getSelectedIndex()
!=0))
                                            //stmt.executeUpdate("insert into rules values(" +
record +",\'" + rul3in.getSelectedItem() + "\',\'" + rul3sub.getSelectedItem() + "\',\'" +
rul3out.getSelectedItem() +"\',\'" + basid+"\');");
                                            rec.setText("Rule(s) in Repository: " +
Integer.toString(record));
                                    }
                                    catch(Exception ex)
                                    {
                                            System.out.println(ex);
                                            return;
                                    }
                                    rul1in.setSelectedIndex(0);
                                    rul2in.setSelectedIndex(0);
                                    rul3in.setSelectedIndex(0);
                                    rul1out.setSelectedIndex(0);
                                    rul2out.setSelectedIndex(0);
                                    rul3out.setSelectedIndex(0);
                                    rul1sub.setSelectedIndex(0);
                                    rul2sub.setSelectedIndex(0);
                                    rul3sub.setSelectedIndex(0);
                                    this.dispose();
                                    main2 mm = new main2();
                                    mm.Gen2 ();
                            }
                       }
                       else
                       {
                              try
                              {
                                    this.dispose();
                                    main2 mm = new main2();
                                    mm.Gen2 ();
                                    stmt.close();
                                    conn.close();
                              }
                              catch(Exception ex)
                              {}
                       }
               }
        }
```

18) suitability.java

```
        //Calculates the data to generate Suitability Matrix
        import javax.swing.*;
        import java.util.*;
        import java.awt.*;
        import java.awt.image.BufferedImage;
        import java.sql.*;
        import java.net.*;
        import javax.imageio.*;
        import javax.swing.border.*;
        import java.io.*;
        import java.awt.event.*;
        public class suitability extends JFrame
        {
            String mod[][] = new String [10][100];
            int[][] mod_int = new int[10][100];
            int[][] total = new int[10][10];
            int x,y,k=0,ab1,step,num1;
            String driver,url,aab,comp2,comp1;
            Connection conn;
            Statement stmt,stmt1,stmt2,stmt3;
            ResultSet rs,rs1,rs2;
            int total1,total2;
            String[] anb = new String[100];
            boolean flag = true,smflag=true;
            public void init1(String[][] mod_t,int ddx,int ddy,String[] anb1,int ab,int step1,int
num) //removing modules
            {
                mod = mod_t;
                x=ddx;
                y=ddy;
                ab1=ab;
                anb = anb1;
                step = step1;
                num1 = num;
                try
                {
                    driver = "com.mysql.jdbc.Driver";
                    url = "jdbc:mysql://localhost:3306/test";
                    Class.forName(driver);
                    conn =DriverManager.getConnection(url,"root","pennstate");
                    stmt = conn.createStatement ();
                    stmt1=conn.createStatement();
                    stmt2=conn.createStatement ();
                    stmt3=conn.createStatement();
                    for(int i=1;i<=x;i++)
                    {
                        for(int j=1;j<=y;j++)
                        {
                            if(!mod[i][j].equals (""))
                            {
                                rs = stmt.executeQuery("Select * from suitability where comp1 =
\'" + mod[i][j] + "\';");
                                while(rs.next())
                                {
                                    aab = Integer.toString (rs.getInt("suitable"));
                                    if(aab.equals("-2"))
                                    {
                                        comp2 = rs.getString ("comp2");
                                         for(int x1=1; x1<=y;x1++) //if comp2 is not in the same
module no need to worry about removing it!
                                        {
                                              if(comp2.equals(mod[i][x1]))
```

```
                                             {
                                                flag=true;
                                                break;
                                             }
                                             else flag=false;
                                         }
                                         if(flag==true)
                                         {
                                            total1=0;
                                            total2=0;
                                            for(int x1=1;x1<=y;x1++)
                                            {
                                               if((x1!=j)&&(!mod[i][x1].equals
(""))&&(!mod[i][x1].equals(comp2)))
                                               {
                                                    rs1 = stmt1.executeQuery("Select * from
suitability where comp1 = \'" + mod[i][j] + "\' and comp2 = \'" + mod[i][x1] + "\';");
                                                     while(rs1.next ())
                                                         total1+=rs1.getInt ("suitable");
                                                    rs1.close();
                                               }
                                               else total1=0;
                                            }
                                            for(int y1=1;y1<=y;y1++)
                                            {
                                               if((y1!=j)&&(!mod[i][y1].equals ("")))
                                               {
                                                   if(!comp2.equals (mod[i][y1]))
                                                   {
                                                       rs2 = stmt2.executeQuery("Select * from
suitability where comp1 = \'" + comp2 + "\' and comp2 = \'" + mod[i][y1] + "\';");
                                                       while(rs2.next ())
                                                           total2+=rs2.getInt ("suitable");
                                                   }
                                                   else total2 = 0;
                                                }
                                            }
                                            if(total1<total2)
                                            {
                                               k=k+1;
                                               System.out.println ("Module Removed: " +
mod[i][j]);

                                               mod[x+k][1] = mod[i][j];//adding model
                                               mod[i][j] = "";//removal
                                               for(int k12=2;k12<=y;k12++)
                                                   mod[x+k][k12]="";
                                            }
                                            if((total1==total2)&&(!mod[i][j].equals
(""))&&(!comp2.equals ("")))
                                            {
                                               for(int y1=1;y1<=y;y1++)
                                               {
                                                   if(mod[i][y1].equals (mod[i][j]))
                                                       total1=y1;
                                                   if(mod[i][y1].equals (comp2))
                                                       total2=y1;
                                               }
                                               if(total1>total2)
                                               {
                                                   k=k+1;
                                                   System.out.println ("Module Removed: " +
mod[i][total1]);

                                                   mod[x+k][1] = mod[i][total1];//adding model
                                                   mod[i][total1] = "";//removal
                                                   for(int k12=2;k12<=y;k12++)
                                                   mod[x+k][k12]="";
                                               }
                                               if(total2>total1)
                                               {
                                                   k=k+1;
                                                   System.out.println ("Module Removed: " +
mod[i][total2]);

                                                   mod[x+k][1] = mod[i][total2];//adding model
                                                   mod[i][total2] = "";//removal
                                                   for(int k12=2;k12<=y;k12++)
                                                   mod[x+k][k12]="";
                                                }
                                             }
                                        }
                                    }
                                }
                                rs.close();
                            }
                        }
                    }
                    x=x+k;
                    k=0;
                    System.out.println("---Modules after Removal Process is done----");
                    for(int i=1;i<=x;i++)
                    {
                        for(int j=1;j<=y;j++)
                        {
                            System.out.print (mod[i][j] + " ");
                        }
                        System.out.println();
                    }
                    System.out.println("----------------");
                    merge1();

                }
                catch (Exception ex)
                {
                    System.out.println(ex);
                }
            }
            public void merge1() //Merging modules
            {
                int count1=0,count2=0,temp,j2=0;
                boolean flag=true,runflg = true;;
                try
                {
                    Marker1: while(true)
                    {
                        for(int i=1;i<=x;i++) //Reseting total matrix.
                        {
                            for(int j=1;j<=x;j++)
                            {
                                total[i][j] = 0;
                            }
                        }
                        for(int i=1;i<=x;i++)
                        {
                            for(int j=1;j<=y;j++)
                            {
                                if(!mod[i][j].equals (""))
                                {
                                    for(int k=1;k<=x;k++)
                                    {
                                        for(int l=1;l<=y;l++)
                                        {
                                            if((k!=i)&&(!mod[k][l].equals("")))
                                            {
                                                rs = stmt3.executeQuery("Select * from
suitability where comp1 = \'" + mod[i][j] + "\' and comp2 = \'" + mod[k][l] + "\';");
                                                while(rs.next())
                                                {
                                                    total[i][k]+=rs.getInt("suitable");
                                                }
                                            }
                                            else total[k][k] = 0;
                                        }
                                    }
                                }
                            }
                        }
                        for(int i=1;i<=x;i++) //Actual merging takes place here
                        {
                            for(int j=1;j<=x;j++)
                            {
                                if((i!=j)&&(total[i][j]>=2))
                                {
                                    if(i>j)
                                    {
                                        temp = i;
                                        i = j;
                                        j = temp;
                                    }
                                    System.out.println ("Merging Module" + i + " and Module" + j
+" :");
                                    for(int u1=1;u1<=y;u1++)
                                     {
                                         if(!mod[i][u1].equals(""))
                                             count1++;
                                     }
                                     for(int u1=1;u1<=y;u1++)
                                     {
                                         if(!mod[j][u1].equals (""))
                                         {
                                              mod[i][count1+u1] = mod[j][u1];
                                              mod[j][u1] = "";
                                         }
                                         else
                                         {
                                             y = count1 + u1;
                                             break;
                                         }
                                     }
                                     for(int u1=1;u1<=y;u1++)
                                     {
                                         if(mod[j][u1].equals(""))
                                             flag=true;
                                         else
                                         {
                                             flag=false;
                                             break;
                                         }
                                     }
                                     if(flag==true)
                                     {
                                         if(j==x)
                                         {
                                            x=x-1;
                                         }
                                         else
                                         {
                                             for(int u2=j;u2<x;u2++)
                                             {
                                                 for(int u3=1;u3<=y;u3++)
                                                 {
                                                    mod[u2][u3] = mod[u2+1][u3];
                                                 }
                                             }
                                             x=x-1;
                                         }
                                         continue Marker1;
                                     }
                                     else
                                         System.out.println("Something wrong");
                                }
                            }
                        }
                        break Marker1;
                    }
                    System.out.println("---Modules are Merging Process is done----");
                    for(int i1=1;i1<=x;i1++)
                    {
                        for(int j1=1;j1<=y;j1++)
                        {
                            if(!mod[i1][j1].equals(""))
                            {
                                j2++;
                                for(int k1=1;k1<step;k1++)
                                {
                                    if(mod[i1][j1].equals (anb[k1]))
                                        mod_int[i1][j2] = k1;
                                }
                            }
                            System.out.print (mod[i1][j1] + " ");
                        }
                        j2=0;
                        System.out.println();
                    }
                    for(int i1=1;i1<=x;i1++)
                    {
                        for(int j1=1;j1<=y;j1++)
                        {
                            System.out.print (mod_int[i1][j1] + " ");
                        }
                        System.out.println ();
                    }
                    System.out.println("----------------");
                    mod mm = new mod();
                    mm.init1(num1, mod_int,x,y,ab1);

                }
                catch (Exception ex)
                {
                    System.out.println(ex);
                }
            }
        }
```

19) updateDFA.java

```
        //Updates the specifications of the selected component
        import java.awt.*;
        import java.applet.*;
        import java.awt.event.*;
        import java.sql.*;
        import javax.swing.*;
        import java.util.*;

        public class updateDFA extends JFrame implements ItemListener,ActionListener
        {
               JTextField compid,dfa,rec;
               JComboBox wt, uc,stiff,hardness,len,shape,size,compos,sym,join,baseid;
               JLabel
wt1,uc1,stiff1,hard1,len1,shape1,size1,compos1,bas1,mov1,align1,join1,sym1,round1,compid1,baseid1
;
               JButton calcdfa,submit,dfa1,main1,loadim;
               JRadioButton yes,no,round,notroun,st,notst,chamf,notchamf;
               ButtonGroup roun,cham,bas,mov;
               int rou=0,record=0,a2;
               double DFA = 0.0;
               Connection conn;
               Statement stmt,stmt2;
               ResultSet rset,rs2;
               String driver,url,abc,abc1,cid,bid,iid,dfid;
               FileDialog dialog1;
                ItemSelectable a;
                JPanel p;
                int key1;
               public void init1(int key)
               {
                      key1 = key;
                       try
                      {
                             driver = "com.mysql.jdbc.Driver";
                             url = "jdbc:mysql://localhost:3306/test";
                             Class.forName(driver);
                             conn =DriverManager.getConnection(url,"root","pennstate");
                             stmt = conn.createStatement();
                               stmt2 = conn.createStatement ();
                               rset = stmt.executeQuery ("select * from DFA where number = " +
key1 + ";");
                               while(rset.next ())
                               {
                                   bid = rset.getString ("baseid");
                                   cid = rset.getString ("compid");
                                   dfid = rset.getString ("dfa");
                                   iid = rset.getString ("image1");
                               }
                      }
                      catch(Exception ex)
```

```
{
     System.out.println(ex);
 }
 p = new JPanel();
 this.setSize(700,700);
 this.setResizable(false);
 this.setTitle ("Updating Component Menu");
 p.setLayout(new GridBagLayout());
 this.setBackground(Color.WHITE);
submit = new JButton("Update Component");
dfa1 = new JButton("Calculate DFA Index");
dfa1.addActionListener(this);
submit.addActionListener(this);
loadim = new JButton("Insert Image File");
loadim.addActionListener(this);
wt1 = new JLabel("What is the approximate weight range in grams?");
bas1 = new JLabel("Does the component have a base?");
uc1 = new JLabel("What are the number of unique components present?");
stiff1 = new JLabel("What is the stiffness (Young's Modulus) in Pa?");
hard1 = new JLabel("What is the vulnerability hardness in Kgf/mm-2 ?");
round1 = new JLabel("What is the overall structure?");
len1 = new JLabel("What is the maximum length in mm ?");
shape1 = new JLabel("What is the shape?");
size1 = new JLabel("What is the size?");
compos1 = new JLabel("What is the composing direction?");
mov1 = new JLabel("What is the composing movement?");
align1 =  new JLabel("What is the alignment characteristic?");
join1 = new JLabel("What is the joining method?");
sym1 = new JLabel("What is the symmetry?");
compid1 = new JLabel("Component ID: ");
baseid1 = new JLabel("Base ID: ");
compid = new JTextField(15);
 compid.setText (cid);
dfa = new JTextField(6);
dfa.setEditable(false);
 dfa.setText (dfid);
rec = new JTextField(15);
rec.setText(iid);
rec.setEditable(false);
roun = new ButtonGroup();
cham = new ButtonGroup();
bas = new ButtonGroup();
mov = new ButtonGroup();
yes = new JRadioButton("Yes",true);
 yes.setActionCommand ("yes1");
 no = new JRadioButton("No",false);
 no.setActionCommand ("no1");
 round = new JRadioButton("Round (L,D)",true);
 round.setActionCommand ("rou");
 notroun = new JRadioButton("Not Round (A,B,C)",false);
round.addItemListener(this);
notroun.addItemListener(this);
st = new JRadioButton("Straight Line",true);
notst = new JRadioButton("Not Straight Line",false);
chamf = new JRadioButton("Chamfer",true);
notchamf = new JRadioButton("No Chamfer",false);
notroun.setActionCommand ("notrou");
 st.setActionCommand ("Stline");
 notst.setActionCommand ("NotSt");
 chamf.setActionCommand ("chamf");
 notchamf.setActionCommand ("notchamf");
 cham.add(chamf);
 cham.add (notchamf);
 bas.add(yes);
 bas.add (no);
 mov.add (st);
 mov.add (notst);
 roun.add (round);
 roun.add (notroun);
 baseid = new JComboBox();
try
{
       String c,b="";
       rs2 = stmt2.executeQuery("Select * from rules");
       while (rs2.next())
       {
```

```
                c = rs2.getString("base_id");
                if(!c.equals(b))
                baseid.addItem(c);
                b=c;
        }
          baseid.setSelectedItem (bid);
}
catch(Exception ex)
{
        System.out.println(ex);
 }
wt = new JComboBox();
wt.addItem("0.1 < G < 2000");
wt.addItem("0.01 <= G <= 0.1");
wt.addItem("2000 <= G <= 6000");
wt.addItem("G < 0.01");
wt.addItem("G > 6000");

join = new JComboBox();
join.addItem("Snap/Screwing/Adhesive Bonding");
join.addItem("Press Fitting/Welding/Riveting/Soldering");

uc = new JComboBox();
uc.addItem("UC < 10");
uc.addItem("UC >= 10");

stiff = new JComboBox();
stiff.addItem("YM > (7.0E + 10)Pa");
stiff.addItem("YM < (7.0E + 10)Pa");

hardness = new JComboBox();
hardness.addItem("H <= 80");
hardness.addItem("80 < H <= 150");
hardness.addItem("H > 150");

len = new JComboBox();
len.addItem("5 < L <= 50");
len.addItem("2 <= L <= 5");
len.addItem("50 <= L <= 2000");
len.addItem("L < 2");
len.addItem("L > 2000");

shape = new JComboBox();
shape.addItem("L/D < 0.8");
shape.addItem("0.8 <= L/D <= 1.5");
shape.addItem("L/D > 1.5");

size = new JComboBox();
size.addItem("0.25 < t <= 50");
size.addItem("t <= 0.25");
size.addItem("t > 50");

compos = new JComboBox();
compos.addItem("Top-Down");
compos.addItem("Side-In");
compos.addItem("Bottom-Up");
compos.addItem("Other");

sym = new JComboBox();
sym.addItem("alpha AND beta symmetric");
sym.addItem("alpha symmetric AND beta asymmetric");
sym.addItem("beta symmetric AND alpha symmetric");
sym.addItem("apha AND beta asymmetric");

GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
gridBagConstraintsx01.gridx = 0;
gridBagConstraintsx01.gridy = 0;
gridBagConstraintsx01.insets = new Insets(5,5,5,5);
p.add(compid1, gridBagConstraintsx01);
GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
gridBagConstraintsx02.gridx = 1;
gridBagConstraintsx02.gridy = 0;
gridBagConstraintsx02.insets = new Insets(5,5,5,5);
gridBagConstraintsx02.gridwidth = 1;
gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
p.add(compid, gridBagConstraintsx02);
```

```
GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
gridBagConstraintsx03.gridx = 2;
gridBagConstraintsx03.gridy = 0;
gridBagConstraintsx03.insets = new Insets(5,5,5,5);
p.add(loadim, gridBagConstraintsx03);

 GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
gridBagConstraintsx04.gridx = 0;
gridBagConstraintsx04.gridy = 1;
gridBagConstraintsx04.insets = new Insets(5,5,5,5);
gridBagConstraintsx04.gridwidth = 1;
gridBagConstraintsx04.fill = GridBagConstraints.BOTH;
p.add(rec, gridBagConstraintsx04);
 GridBagConstraints gridBagConstraintsx04a = new GridBagConstraints();
gridBagConstraintsx04a.gridx = 1;
gridBagConstraintsx04a.gridy = 1;
gridBagConstraintsx04a.insets = new Insets(5,5,5,5);
gridBagConstraintsx04a.gridwidth = 1;
gridBagConstraintsx04a.fill = GridBagConstraints.BOTH;
p.add(baseid1, gridBagConstraintsx04a);
GridBagConstraints gridBagConstraintsx04b = new GridBagConstraints();
gridBagConstraintsx04b.gridx = 2;
gridBagConstraintsx04b.gridy = 1;
gridBagConstraintsx04b.insets = new Insets(5,5,5,5);
gridBagConstraintsx04b.gridwidth = 1;
gridBagConstraintsx04b.fill = GridBagConstraints.BOTH;
p.add(baseid, gridBagConstraintsx04b);

GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
gridBagConstraintsx05.gridx = 0;
gridBagConstraintsx05.gridy = 2;
gridBagConstraintsx05.insets = new Insets(5,5,5,5);
p.add(wt1, gridBagConstraintsx05);
GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
gridBagConstraintsx06.gridx = 1;
gridBagConstraintsx06.gridy = 2;
gridBagConstraintsx06.insets = new Insets(5,5,5,5);
gridBagConstraintsx06.gridwidth = 2;
gridBagConstraintsx06.fill = GridBagConstraints.BOTH;
p.add(wt, gridBagConstraintsx06);

GridBagConstraints gridBagConstraintsx07 = new GridBagConstraints();
gridBagConstraintsx07.gridx = 0;
gridBagConstraintsx07.gridy = 3;
gridBagConstraintsx07.insets = new Insets(5,5,5,5);
p.add(uc1, gridBagConstraintsx07);
GridBagConstraints gridBagConstraintsx08 = new GridBagConstraints();
gridBagConstraintsx08.gridx = 1;
gridBagConstraintsx08.gridy = 3;
gridBagConstraintsx08.insets = new Insets(5,5,5,5);
gridBagConstraintsx08.gridwidth = 2;
gridBagConstraintsx08.fill = GridBagConstraints.BOTH;
p.add(uc, gridBagConstraintsx08);

GridBagConstraints gridBagConstraintsx09 = new GridBagConstraints();
gridBagConstraintsx09.gridx = 0;
gridBagConstraintsx09.gridy = 4;
gridBagConstraintsx09.insets = new Insets(5,5,5,5);
p.add(bas1, gridBagConstraintsx09);
GridBagConstraints gridBagConstraintsx010 = new GridBagConstraints();
gridBagConstraintsx010.gridx = 1;
gridBagConstraintsx010.gridy = 4;
gridBagConstraintsx010.insets = new Insets(5,5,5,5);
p.add(yes, gridBagConstraintsx010);
GridBagConstraints gridBagConstraintsx011 = new GridBagConstraints();
gridBagConstraintsx011.gridx = 2;
gridBagConstraintsx011.gridy = 4;
gridBagConstraintsx011.insets = new Insets(5,5,5,5);
p.add(no, gridBagConstraintsx011);

GridBagConstraints gridBagConstraintsx012 = new GridBagConstraints();
gridBagConstraintsx012.gridx = 0;
gridBagConstraintsx012.gridy = 5;
gridBagConstraintsx012.insets = new Insets(5,5,5,5);
p.add(stiff1, gridBagConstraintsx012);
GridBagConstraints gridBagConstraintsx013 = new GridBagConstraints();
```

```
gridBagConstraintsx013.gridx = 1;
gridBagConstraintsx013.gridy = 5;
gridBagConstraintsx013.insets = new Insets(5,5,5,5);
gridBagConstraintsx013.gridwidth = 2;
gridBagConstraintsx013.fill = GridBagConstraints.BOTH;
p.add(stiff, gridBagConstraintsx013);

GridBagConstraints gridBagConstraintsx014 = new GridBagConstraints();
gridBagConstraintsx014.gridx = 0;
gridBagConstraintsx014.gridy = 6;
gridBagConstraintsx014.insets = new Insets(5,5,5,5);
p.add(hard1, gridBagConstraintsx014);
GridBagConstraints gridBagConstraintsx015 = new GridBagConstraints();
gridBagConstraintsx015.gridx = 1;
gridBagConstraintsx015.gridy = 6;
gridBagConstraintsx015.insets = new Insets(5,5,5,5);
gridBagConstraintsx015.gridwidth = 2;
gridBagConstraintsx015.fill = GridBagConstraints.BOTH;
p.add(hardness, gridBagConstraintsx015);

GridBagConstraints gridBagConstraintsx016 = new GridBagConstraints();
gridBagConstraintsx016.gridx = 0;
gridBagConstraintsx016.gridy = 7;
gridBagConstraintsx016.insets = new Insets(5,5,5,5);
p.add(round1, gridBagConstraintsx016);
GridBagConstraints gridBagConstraintsx017 = new GridBagConstraints();
gridBagConstraintsx017.gridx = 1;
gridBagConstraintsx017.gridy = 7;
gridBagConstraintsx017.insets = new Insets(5,5,5,5);
p.add(round, gridBagConstraintsx017);
GridBagConstraints gridBagConstraintsx018 = new GridBagConstraints();
gridBagConstraintsx018.gridx = 2;
gridBagConstraintsx018.gridy = 7;
gridBagConstraintsx018.insets = new Insets(5,5,5,5);
p.add(notroun, gridBagConstraintsx018);

GridBagConstraints gridBagConstraintsx019 = new GridBagConstraints();
gridBagConstraintsx019.gridx = 0;
gridBagConstraintsx019.gridy = 8;
gridBagConstraintsx019.insets = new Insets(5,5,5,5);
p.add(len1, gridBagConstraintsx019);
GridBagConstraints gridBagConstraintsx020 = new GridBagConstraints();
gridBagConstraintsx020.gridx = 1;
gridBagConstraintsx020.gridy = 8;
gridBagConstraintsx020.insets = new Insets(5,5,5,5);
gridBagConstraintsx020.gridwidth = 2;
gridBagConstraintsx020.fill = GridBagConstraints.BOTH;
p.add(len, gridBagConstraintsx020);

GridBagConstraints gridBagConstraintsx021 = new GridBagConstraints();
gridBagConstraintsx021.gridx = 0;
gridBagConstraintsx021.gridy = 9;
gridBagConstraintsx021.insets = new Insets(5,5,5,5);
p.add(shape1, gridBagConstraintsx021);
GridBagConstraints gridBagConstraintsx022 = new GridBagConstraints();
gridBagConstraintsx022.gridx = 1;
gridBagConstraintsx022.gridy = 9;
gridBagConstraintsx022.insets = new Insets(5,5,5,5);
gridBagConstraintsx022.gridwidth = 2;
gridBagConstraintsx022.fill = GridBagConstraints.BOTH;
p.add(shape, gridBagConstraintsx022);

GridBagConstraints gridBagConstraintsx023 = new GridBagConstraints();
gridBagConstraintsx023.gridx = 0;
gridBagConstraintsx023.gridy = 10;
gridBagConstraintsx023.insets = new Insets(5,5,5,5);
p.add(size1, gridBagConstraintsx023);
GridBagConstraints gridBagConstraintsx024 = new GridBagConstraints();
gridBagConstraintsx024.gridx = 1;
gridBagConstraintsx024.gridy = 10;
gridBagConstraintsx024.insets = new Insets(5,5,5,5);
gridBagConstraintsx024.gridwidth = 2;
gridBagConstraintsx024.fill = GridBagConstraints.BOTH;
p.add(size, gridBagConstraintsx024);

GridBagConstraints gridBagConstraintsx025 = new GridBagConstraints();
```

```
gridBagConstraintsx025.gridx = 0;
gridBagConstraintsx025.gridy = 11;
gridBagConstraintsx025.insets = new Insets(5,5,5,5);
p.add(compos1, gridBagConstraintsx025);
GridBagConstraints gridBagConstraintsx026 = new GridBagConstraints();
gridBagConstraintsx026.gridx = 1;
gridBagConstraintsx026.gridy = 11;
gridBagConstraintsx026.insets = new Insets(5,5,5,5);
gridBagConstraintsx026.gridwidth = 2;
gridBagConstraintsx026.fill = GridBagConstraints.BOTH;
p.add(compos, gridBagConstraintsx026);

GridBagConstraints gridBagConstraintsx027 = new GridBagConstraints();
gridBagConstraintsx027.gridx = 0;
gridBagConstraintsx027.gridy = 12;
gridBagConstraintsx027.insets = new Insets(5,5,5,5);
p.add(mov1, gridBagConstraintsx027);
GridBagConstraints gridBagConstraintsx028 = new GridBagConstraints();
gridBagConstraintsx028.gridx = 1;
gridBagConstraintsx028.gridy = 12;
gridBagConstraintsx028.insets = new Insets(5,5,5,5);
p.add(st, gridBagConstraintsx028);
GridBagConstraints gridBagConstraintsx029 = new GridBagConstraints();
gridBagConstraintsx029.gridx = 2;
gridBagConstraintsx029.gridy = 12;
gridBagConstraintsx029.insets = new Insets(5,5,5,5);
p.add(notst, gridBagConstraintsx029);

GridBagConstraints gridBagConstraintsx030 = new GridBagConstraints();
gridBagConstraintsx030.gridx = 0;
gridBagConstraintsx030.gridy = 13;
gridBagConstraintsx030.insets = new Insets(5,5,5,5);
p.add(align1, gridBagConstraintsx030);
GridBagConstraints gridBagConstraintsx031 = new GridBagConstraints();
gridBagConstraintsx031.gridx = 1;
gridBagConstraintsx031.gridy = 13;
gridBagConstraintsx031.insets = new Insets(5,5,5,5);
p.add(chamf, gridBagConstraintsx031);
GridBagConstraints gridBagConstraintsx032 = new GridBagConstraints();
gridBagConstraintsx032.gridx = 2;
gridBagConstraintsx032.gridy = 13;
gridBagConstraintsx032.insets = new Insets(5,5,5,5);
p.add(notchamf, gridBagConstraintsx032);

GridBagConstraints gridBagConstraintsx033 = new GridBagConstraints();
gridBagConstraintsx033.gridx = 0;
gridBagConstraintsx033.gridy = 14;
gridBagConstraintsx033.insets = new Insets(5,5,5,5);
p.add(join1, gridBagConstraintsx033);
GridBagConstraints gridBagConstraintsx034 = new GridBagConstraints();
gridBagConstraintsx034.gridx = 1;
gridBagConstraintsx034.gridy = 14;
gridBagConstraintsx034.insets = new Insets(5,5,5,5);
gridBagConstraintsx034.gridwidth = 2;
gridBagConstraintsx034.fill = GridBagConstraints.BOTH;
p.add(join, gridBagConstraintsx034);

GridBagConstraints gridBagConstraintsx035 = new GridBagConstraints();
gridBagConstraintsx035.gridx = 0;
gridBagConstraintsx035.gridy = 15;
gridBagConstraintsx035.insets = new Insets(5,5,5,5);
p.add(sym1, gridBagConstraintsx035);
GridBagConstraints gridBagConstraintsx036 = new GridBagConstraints();
gridBagConstraintsx036.gridx = 1;
gridBagConstraintsx036.gridy = 15;
gridBagConstraintsx036.insets = new Insets(5,5,5,5);
gridBagConstraintsx036.gridwidth = 2;
gridBagConstraintsx036.fill = GridBagConstraints.BOTH;
p.add(sym, gridBagConstraintsx036);

GridBagConstraints gridBagConstraintsx037 = new GridBagConstraints();
gridBagConstraintsx037.gridx = 0;
gridBagConstraintsx037.gridy = 16;
gridBagConstraintsx037.insets = new Insets(5,5,5,5);
p.add(dfa1, gridBagConstraintsx037);
GridBagConstraints gridBagConstraintsx038 = new GridBagConstraints();
```

```
                gridBagConstraintsx038.gridx = 1;
                gridBagConstraintsx038.gridy = 16;
                gridBagConstraintsx038.insets = new Insets(5,5,5,5);
                gridBagConstraintsx038.gridwidth = 2;
                gridBagConstraintsx038.fill = GridBagConstraints.BOTH;
                p.add(dfa, gridBagConstraintsx038);

                GridBagConstraints gridBagConstraintsx040 = new GridBagConstraints();
                gridBagConstraintsx040.gridx = 0;
                gridBagConstraintsx040.gridy = 17;
                gridBagConstraintsx040.insets = new Insets(5,5,5,5);
                gridBagConstraintsx040.gridwidth = 3;
                gridBagConstraintsx040.fill = GridBagConstraints.BOTH;
                p.add(submit, gridBagConstraintsx040);
                 this.getContentPane().add(p);
                 this.setVisible(true);
        }
        public void itemStateChanged(ItemEvent ie)
        {
                a = ie.getItemSelectable ();
                if(a == round)
                {
                        shape.removeAllItems();
                        shape.addItem("L/D < 0.8");
                        shape.addItem("0.8 <= L/D <= 1.5");
                        shape.addItem("L/D > 1.5");

                        size.removeAllItems();
                        size.addItem("0.25 < t <= 50");
                        size.addItem("t <= 0.25");
                        size.addItem("t > 50");

                        sym.removeAllItems();
                        sym.addItem("alpha AND beta symmetric");
                        sym.addItem("alpha symmetric AND beta asymmetric");
                        sym.addItem("beta symmetric AND alpha symmetric");
                        sym.addItem("apha AND beta asymmetric");
                }
                else
                {
                        shape.removeAllItems();
                        shape.addItem("A/B <= 3 AND A/C > 4");
                        shape.addItem("A/B > 3");
                        shape.addItem("A/B <= 3 AND A/C <= 4");

                        size.removeAllItems();
                        size.addItem("5 < L <= 50");
                        size.addItem("2 <= L <= 5");
                        size.addItem("L < 2");
                        size.addItem("50 <= L <= 2000");
                        size.addItem("L > 2000");


                        sym.removeAllItems();
                        sym.addItem("180 deg symmetric about many axis");
                        sym.addItem("180 deg symmetric about one axis");
                        sym.addItem("Non Symmetric");
                }
        }
        public void actionPerformed(ActionEvent ae)
        {
                String str = ae.getActionCommand();
                if(str.equals("Calculate DFA Index"))
                {
                        DFA = 0.0;
                        String wei = (String) wt.getSelectedItem();
                        if(wei.equals("0.1 < G < 2000")) DFA=1;
                        else if(wei.equals("0.01 <= G <= 0.1")) DFA=2;
                        else if(wei.equals("2000 <= G <= 6000")) DFA=2;
                        else if(wei.equals("G < 0.01")) DFA=4;
                        else DFA=4;

                        String uco = (String) uc.getSelectedItem();
                        if(uco.equals("UC < 10")) DFA+=1;
                        else DFA+=4;
```

```
                               String a1 = (String) bas.getSelection().getActionCommand();
                               if(a1.equals("yes1")) DFA+=1;
                               else DFA+=4;

                               String sti = (String) stiff.getSelectedItem();
                               if(sti.equals("YM > (7.0E + 10)Pa")) DFA+=1;
                               else DFA+=4;

                               String har = (String) hardness.getSelectedItem();
                               if(har.equals("H <= 80")) DFA+=1;
                               else if(har.equals("80 < H <= 150")) DFA+=2;
                               else DFA+=4;

                               String leng = (String) len.getSelectedItem();
                               if(leng.equals("5 < L <= 50")) DFA+=1;
                               else if(leng.equals("2 <= L <= 5")) DFA+=2;
                               else if(leng.equals("50 <= L <= 2000")) DFA+=2;

                               else if(leng.equals("L < 2")) DFA+=4;

                               else DFA+=4;

                               String shap = (String) shape.getSelectedItem();
                               if(shap.equals("L/D < 0.8")) DFA+=1;
                               else if(shap.equals("0.8 <= L/D <= 1.5")) DFA+=2;
                               else if(shap.equals("L/D > 1.5")) DFA+=4;
                               else if(shap.equals("A/B <= 3 AND A/C > 4")) DFA+=1;
                               else if(shap.equals("A/B > 3")) DFA+=1;
                               else DFA+=2;

                               String siz = (String) size.getSelectedItem();
                               if(siz.equals("0.25 < t <= 50")) DFA+=1;
                               else if(siz.equals("t <= 0.25")) DFA+=4;
                               else if(siz.equals("t > 50")) DFA+=4;
                               else if(siz.equals("5 < L <= 50")) DFA+=1;
                               else if(siz.equals("2 <= L <= 5")) DFA+=2;
                               else if(siz.equals("L < 2")) DFA+=4;
                               else if(siz.equals("50 <= L <= 2000")) DFA+=2;
                               else DFA+=4;

                               String com = (String) compos.getSelectedItem();
                               if(com.equals("Top-Down")) DFA+=1;
                               else if(com.equals("Side-In")) DFA+=2;
                               else if(com.equals("Bottom-Up")) DFA+=4;
                               else DFA+=6;

                               String movem = mov.getSelection().getActionCommand();
                               if(movem.equals("Stline")) DFA+=1;
                               else DFA+=2;

                               String cha = cham.getSelection().getActionCommand();
                               if(cha.equals("chamf")) DFA+=1;
                               else DFA+=4;

                               String joi = (String) join.getSelectedItem();
                               if(joi.equals("Snap/Screwing/Adhesive Bonding")) DFA+=1;
                               else if(joi.equals("Press Fitting/Welding/Riveting/Soldering"))
DFA+=2;

                               String symm = (String) sym.getSelectedItem();
                               if(symm.equals("alpha AND beta symmetric")) DFA+=1;
                               else if(symm.equals("alpha symmetric AND beta asymmetric")) DFA+=2;
                               else if(symm.equals("beta symmetric AND alpha symmetric")) DFA+=2;
                               else if(symm.equals("apha AND beta asymmetric")) DFA+=4;

                               else if(symm.equals("180 deg symmetric about many axis")) DFA+=1;
                               else if(symm.equals("180 deg symmetric about one axis")) DFA+=2;
                               else if(symm.equals("Non Symmetric")) DFA+=4;
                               DFA = (10*(DFA-13.0))/37.0;
                               dfa.setText(Double.toString(DFA));
                       }
                       else if(str.equals("Update Component"))
                       {
                            int response = JOptionPane.showConfirmDialog (this,"Are you sure you
want to update the existing component?","Confirm Update
Dialog",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
```

```
                            if(response == JOptionPane.YES_OPTION)
                            {
                                String comid = compid.getText();
                              String basid = (String) baseid.getSelectedItem();
                              sym.setSelectedIndex(0);
                              join.setSelectedIndex(0);
                              wt.setSelectedIndex(0);
                              uc.setSelectedIndex(0);
                              stiff.setSelectedIndex(0);
                              hardness.setSelectedIndex(0);
                              len.setSelectedIndex(0);
                              shape.setSelectedIndex(0);
                              size.setSelectedIndex(0);
                              compos.setSelectedIndex(0);
                                ButtonModel model1 = yes.getModel();
                              bas.setSelected(model1,true);
                                ButtonModel model2 = st.getModel();
                              mov.setSelected (model2,true);
                                ButtonModel model3 = chamf.getModel();
                              cham.setSelected(model3, true);
                              try
                              {
                                    stmt2.executeUpdate("update DFA set baseid = \'" + basid
+"\', compid = \'" + comid +"\', dfa = " + Float.parseFloat (dfa.getText ()) +", image1 = \'"+
rec.getText () +"\' where number = " + key1 + ";");
                                    stmt.close();
                                    stmt2.close();
                                    rset.close();
                                    rs2.close();
                                    conn.close();
                                    this.dispose();
                                    DFA2 dd = new DFA2();
                                    dd.Gen1();

                              }
                              catch(Exception ex)
                              {
                                      System.out.println(ex);
                                      return;
                              }
                            }
                            else
                            {
                                this.dispose();
                                DFA2 gg1 = new DFA2();
                                gg1.Gen1 ();
                            }
                       }
                       else if(str.equals("Insert Image File"))
                       {
                              Frame myFrame = getFrame(loadim);
                                dialog1 = new FileDialog(myFrame, "Open File", FileDialog.LOAD);
                              dialog1.setVisible(true);
                              abc = dialog1.getDirectory()+dialog1.getFile();
                              StringTokenizer st1 = new StringTokenizer(abc,"\\");
                              abc1 = st1.nextToken();
                              while(st1.hasMoreTokens())
                              abc1 += ("/" + st1.nextToken());
                              rec.setText (abc1);
                       }
                       else
                       {
                              try
                              {
                                      this.dispose();
                                        main2 m = new main2();
                                        m.Gen2();
                                      stmt.close();
                                      conn.close();
                              }
                              catch(Exception ex)
                              {
                                      System.out.println(ex);
                                      return;
                              }
                       }
```

```
                }
                static Frame getFrame (Component c)
                {
                        Frame frame1 = null;
                         while ((c= c.getParent())!= null)
                        {
                                if (c instanceof Frame)
                                frame1 = (Frame) c;
                         }
                         return frame1;
                 }
        }
```

20) updateRules.java

```
        //Updates the rules that are already stored in the Design Repository
        import java.awt.*;
        import java.applet.*;
        import java.awt.event.*;
        import java.sql.*;
        import javax.swing.*;

        public class updateRules extends JFrame implements ActionListener
        {
                JTextField baseid;
                JComboBox rul1in,rul2in,rul3in,rul1sub,rul2sub,rul3sub,rul1out,rul2out,rul3out;
                JLabel inpu,subf,outpu,baseid1,and1,and2,jLabel0;
                JButton inser,main1;
                int rul=0,record=0,key;
                Connection conn;
                Statement stmt;
                ResultSet rset;
                String a, driver,url;
                FileDialog dialog1;
                 String bas,inp1,inp2,inp3,su1,su2,su3,oup1,oup2,oup3;
                public void init1(int key1)
                {
                        key = key1;
                         JPanel p = new JPanel();
                         setSize(700,700);
                         p.setLayout(new GridBagLayout());
                         setResizable(false);
                         this.setTitle("Update Rule Menu");
                         jLabel0 = new JLabel("Updating the Rule");
                         jLabel0.setFont(new Font("System",Font.BOLD,25));
                         try
                        {
                                driver = "com.mysql.jdbc.Driver";
                                url = "jdbc:mysql://localhost:3306/test";
                                Class.forName(driver);
                                conn =DriverManager.getConnection(url,"root","pennstate");
                                stmt = conn.createStatement();
                                rset = stmt.executeQuery("Select * from rules where ruleno = " +
key + ";");
                                while (rset.next())
                                {
                                      bas = rset.getString ("base_id");
                                      inp1 = rset.getString("input1");
                                      oup1 = rset.getString ("output1");
                                      su1 = rset.getString ("subfunction");
                                      inp2 = rset.getString("input2");
                                      oup2 = rset.getString ("output2");
                                      su2 = rset.getString ("subfunction2");
                                      inp3 = rset.getString("input3");
                                      oup3 = rset.getString ("output3");
                                      su3 = rset.getString ("subfunction3");
                                }
                        }
                        catch(Exception ex)
                        {
                                System.out.println(ex);
                                return;
                        }
                        inser = new JButton("Update Selected Rule");
                        inser.addActionListener(this);
                        main1 = new JButton("Design Repository Menu");
```

```
main1.addActionListener(this);
inpu = new JLabel("Input Flow");
subf = new JLabel("Sub-Function");
outpu = new JLabel("Output Flow");
baseid1 = new JLabel("Base ID: ");
and1 = new JLabel("AND");
and2 = new JLabel("AND");
baseid = new JTextField(15);
 baseid.setEditable(false);
 baseid.setText (bas);

rul1in = new JComboBox();
rul1in.addItem("");
rul1in.addItem("Acoustic Energy");
 rul1in.addItem("Analog Control");
 rul1in.addItem("Auditory");
rul1in.addItem("Biological Energy");
rul1in.addItem("Chemical Energy");
 rul1in.addItem("Colloidal");
 rul1in.addItem("Composite");
 rul1in.addItem("Discrete Control");
rul1in.addItem("Electrical Energy");
rul1in.addItem("Electromagnetic Energy");
 rul1in.addItem("Gas");
 rul1in.addItem("Gas-Gas Mixture");
rul1in.addItem("Human Energy");
rul1in.addItem("Human Material");
rul1in.addItem("Hydraulic Energy");
 rul1in.addItem("Liquid");
rul1in.addItem("Liquid-Gas");
 rul1in.addItem("Liquid-Liquid");
rul1in.addItem("Magnetic Energy");
rul1in.addItem("Mechanical Energy");
rul1in.addItem("Object");
 rul1in.addItem("Olfactory");
rul1in.addItem("Optical Energy");
 rul1in.addItem("Particulate");
 rul1in.addItem("Plasma");
rul1in.addItem("Pneumatic Energy");
rul1in.addItem("Radioactive/Nuclear");
rul1in.addItem("Rotational Energy");
rul1in.addItem("Solar Energy");
 rul1in.addItem("Solid");
 rul1in.addItem("Solid-Gas");
rul1in.addItem("Solid-Liquid");
 rul1in.addItem("Solid-Liquid-Gas");
rul1in.addItem("Solid-Solid");
 rul1in.addItem("Thermal Energy");
 rul1in.addItem("Translational Energy");
 rul1in.addItem("Tactile");
rul1in.addItem("Taste");
 rul1in.addItem("Visual");
 rul1in.setSelectedItem (inp1);

 rul2in = new JComboBox();
rul2in.addItem("");
rul2in.addItem("Acoustic Energy");
 rul2in.addItem("Analog Control");
 rul2in.addItem("Auditory");
rul2in.addItem("Biological Energy");
rul2in.addItem("Chemical Energy");
 rul2in.addItem("Colloidal");
 rul2in.addItem("Composite");
 rul2in.addItem("Discrete Control");
rul2in.addItem("Electrical Energy");
rul2in.addItem("Electromagnetic Energy");
 rul2in.addItem("Gas");
 rul2in.addItem("Gas-Gas Mixture");
rul2in.addItem("Human Energy");
rul2in.addItem("Human Material");
rul2in.addItem("Hydraulic Energy");
 rul2in.addItem("Liquid");
rul2in.addItem("Liquid-Gas");
 rul2in.addItem("Liquid-Liquid");
rul2in.addItem("Magnetic Energy");
rul2in.addItem("Mechanical Energy");
```

```
rul2in.addItem("Object");
 rul2in.addItem("Olfactory");
rul2in.addItem("Optical Energy");
 rul2in.addItem("Particulate");
 rul2in.addItem("Plasma");
rul2in.addItem("Pneumatic Energy");
rul2in.addItem("Radioactive/Nuclear");
rul2in.addItem("Rotational Energy");
rul2in.addItem("Solar Energy");
 rul2in.addItem("Solid");
 rul2in.addItem("Solid-Gas");
rul2in.addItem("Solid-Liquid");
 rul2in.addItem("Solid-Liquid-Gas");
rul2in.addItem("Solid-Solid");
 rul2in.addItem("Thermal Energy");
 rul2in.addItem("Translational Energy");
 rul2in.addItem("Tactile");
rul2in.addItem("Taste");
 rul2in.addItem("Visual");
 rul2in.setSelectedItem (inp2);

 rul3in = new JComboBox();
rul3in.addItem("");
rul3in.addItem("Acoustic Energy");
 rul3in.addItem("Analog Control");
 rul3in.addItem("Auditory");
rul3in.addItem("Biological Energy");
rul3in.addItem("Chemical Energy");
 rul3in.addItem("Colloidal");
 rul3in.addItem("Composite");
 rul3in.addItem("Discrete Control");
rul3in.addItem("Electrical Energy");
rul3in.addItem("Electromagnetic Energy");
 rul3in.addItem("Gas");
 rul3in.addItem("Gas-Gas Mixture");
rul3in.addItem("Human Energy");
rul3in.addItem("Human Material");
rul3in.addItem("Hydraulic Energy");
 rul3in.addItem("Liquid");
rul3in.addItem("Liquid-Gas");
 rul3in.addItem("Liquid-Liquid");
rul3in.addItem("Magnetic Energy");
rul3in.addItem("Mechanical Energy");
rul3in.addItem("Object");
 rul3in.addItem("Olfactory");
rul3in.addItem("Optical Energy");
 rul3in.addItem("Particulate");
 rul3in.addItem("Plasma");
rul3in.addItem("Pneumatic Energy");
rul3in.addItem("Radioactive/Nuclear");
rul3in.addItem("Rotational Energy");
rul3in.addItem("Solar Energy");
 rul3in.addItem("Solid");
 rul3in.addItem("Solid-Gas");
rul3in.addItem("Solid-Liquid");
 rul3in.addItem("Solid-Liquid-Gas");
rul3in.addItem("Solid-Solid");
 rul3in.addItem("Thermal Energy");
 rul3in.addItem("Translational Energy");
 rul3in.addItem("Tactile");
rul3in.addItem("Taste");
 rul3in.addItem("Visual");
 rul3in.setSelectedItem (inp3);

rul1out = new JComboBox();
rul1out.addItem("");
rul1out.addItem("Acoustic Energy");
 rul1out.addItem("Analog Control");
 rul1out.addItem("Auditory");
rul1out.addItem("Biological Energy");
rul1out.addItem("Chemical Energy");
 rul1out.addItem("Colloidal");
 rul1out.addItem("Composite");
 rul1out.addItem("Discrete Control");
rul1out.addItem("Electrical Energy");
rul1out.addItem("Electromagnetic Energy");
```

```
 rul1out.addItem("Gas");
 rul1out.addItem("Gas-Gas Mixture");
rul1out.addItem("Human Energy");
rul1out.addItem("Human Material");
rul1out.addItem("Hydraulic Energy");
 rul1out.addItem("Liquid");
rul1out.addItem("Liquid-Gas");
 rul1out.addItem("Liquid-Liquid");
rul1out.addItem("Magnetic Energy");
rul1out.addItem("Mechanical Energy");
rul1out.addItem("Object");
 rul1out.addItem("Olfactory");
rul1out.addItem("Optical Energy");
 rul1out.addItem("Particulate");
 rul1out.addItem("Plasma");
rul1out.addItem("Pneumatic Energy");
rul1out.addItem("Radioactive/Nuclear");
rul1out.addItem("Rotational Energy");
rul1out.addItem("Solar Energy");
 rul1out.addItem("Solid");
 rul1out.addItem("Solid-Gas");
rul1out.addItem("Solid-Liquid");
 rul1out.addItem("Solid-Liquid-Gas");
rul1out.addItem("Solid-Solid");
 rul1out.addItem("Thermal Energy");
 rul1out.addItem("Translational Energy");
 rul1out.addItem("Tactile");
rul1out.addItem("Taste");
 rul1out.addItem("Visual");
 rul1out.setSelectedItem (oup1);

 rul2out = new JComboBox();
rul2out.addItem("");
rul2out.addItem("Acoustic Energy");
 rul2out.addItem("Analog Control");
 rul2out.addItem("Auditory");
rul2out.addItem("Biological Energy");
rul2out.addItem("Chemical Energy");
 rul2out.addItem("Colloidal");
 rul2out.addItem("Composite");
 rul2out.addItem("Discrete Control");
rul2out.addItem("Electrical Energy");
rul2out.addItem("Electromagnetic Energy");
 rul2out.addItem("Gas");
 rul2out.addItem("Gas-Gas Mixture");
rul2out.addItem("Human Energy");
rul2out.addItem("Human Material");
rul2out.addItem("Hydraulic Energy");
 rul2out.addItem("Liquid");
rul2out.addItem("Liquid-Gas");
 rul2out.addItem("Liquid-Liquid");
rul2out.addItem("Magnetic Energy");
rul2out.addItem("Mechanical Energy");
rul2out.addItem("Object");
 rul2out.addItem("Olfactory");
rul2out.addItem("Optical Energy");
 rul2out.addItem("Particulate");
 rul2out.addItem("Plasma");
rul2out.addItem("Pneumatic Energy");
rul2out.addItem("Radioactive/Nuclear");
rul2out.addItem("Rotational Energy");
rul2out.addItem("Solar Energy");
 rul2out.addItem("Solid");
 rul2out.addItem("Solid-Gas");
rul2out.addItem("Solid-Liquid");
 rul2out.addItem("Solid-Liquid-Gas");
rul2out.addItem("Solid-Solid");
 rul2out.addItem("Thermal Energy");
 rul2out.addItem("Translational Energy");
 rul2out.addItem("Tactile");
rul2out.addItem("Taste");
 rul2out.addItem("Visual");
rul2out.setSelectedItem (oup2);

 rul3out = new JComboBox();
rul3out.addItem("");
```

```
rul3out.addItem("Acoustic Energy");
 rul3out.addItem("Analog Control");
 rul3out.addItem("Auditory");
rul3out.addItem("Biological Energy");
rul3out.addItem("Chemical Energy");
 rul3out.addItem("Colloidal");
 rul3out.addItem("Composite");
 rul3out.addItem("Discrete Control");
rul3out.addItem("Electrical Energy");
rul3out.addItem("Electromagnetic Energy");
 rul3out.addItem("Gas");
 rul3out.addItem("Gas-Gas Mixture");
rul3out.addItem("Human Energy");
rul3out.addItem("Human Material");
rul3out.addItem("Hydraulic Energy");
 rul3out.addItem("Liquid");
rul3out.addItem("Liquid-Gas");
 rul3out.addItem("Liquid-Liquid");
rul3out.addItem("Magnetic Energy");
rul3out.addItem("Mechanical Energy");
rul3out.addItem("Object");
 rul3out.addItem("Olfactory");
rul3out.addItem("Optical Energy");
 rul3out.addItem("Particulate");
 rul3out.addItem("Plasma");
rul3out.addItem("Pneumatic Energy");
rul3out.addItem("Radioactive/Nuclear");
rul3out.addItem("Rotational Energy");
rul3out.addItem("Solar Energy");
 rul3out.addItem("Solid");
 rul3out.addItem("Solid-Gas");
rul3out.addItem("Solid-Liquid");
 rul3out.addItem("Solid-Liquid-Gas");
rul3out.addItem("Solid-Solid");
 rul3out.addItem("Thermal Energy");
 rul3out.addItem("Translational Energy");
 rul3out.addItem("Tactile");
rul3out.addItem("Taste");
 rul3out.addItem("Visual");
 rul3out.setSelectedItem (oup3);

rul1sub = new JComboBox();
rul1sub.addItem("");
rul1sub.addItem("Actuate");
 rul1sub.addItem("Allow DOF");
 rul1sub.addItem("Branch");
rul1sub.addItem("Change");
 rul1sub.addItem("Channel");
rul1sub.addItem("Collect");
 rul1sub.addItem("Condition");
 rul1sub.addItem("Connect");
rul1sub.addItem("Contain");
 rul1sub.addItem("Control Magnitude");
 rul1sub.addItem("Convert");
rul1sub.addItem("Couple");
rul1sub.addItem("Decrease");
rul1sub.addItem("Decrement");
rul1sub.addItem("Detect");
rul1sub.addItem("Display");
 rul1sub.addItem("Distribute");
 rul1sub.addItem("Divide");
 rul1sub.addItem("Export");
 rul1sub.addItem("Extract");
 rul1sub.addItem("Guide");
 rul1sub.addItem("Import");
 rul1sub.addItem("Increase");
rul1sub.addItem("Increment");
rul1sub.addItem("Indicate");
rul1sub.addItem("Inhibit");
rul1sub.addItem("Join");
rul1sub.addItem("Link");
 rul1sub.addItem("Measure");
rul1sub.addItem("Mix");
 rul1sub.addItem("Position");
 rul1sub.addItem("Prevent");
rul1sub.addItem("Process");
```

```
rul1sub.addItem("Provision");
 rul1sub.addItem("Regulate");
rul1sub.addItem("Remove");
 rul1sub.addItem("Rotate");
rul1sub.addItem("Secure");
 rul1sub.addItem("Sense");
 rul1sub.addItem("Seperate");
 rul1sub.addItem("Shape");
 rul1sub.addItem("Signal");
 rul1sub.addItem("Stabilize");
 rul1sub.addItem("Stop");
 rul1sub.addItem("Store");
 rul1sub.addItem("Supply");
 rul1sub.addItem("Support");
 rul1sub.addItem("Track");
 rul1sub.addItem("Transfer");
rul1sub.addItem("Translate");
 rul1sub.addItem("Transmit");
 rul1sub.addItem("Transport");
 rul1sub.setSelectedItem (su1);

rul2sub = new JComboBox();
rul2sub.addItem("");
rul2sub.addItem("Actuate");
 rul2sub.addItem("Allow DOF");
 rul2sub.addItem("Branch");
rul2sub.addItem("Change");
 rul2sub.addItem("Channel");
rul2sub.addItem("Collect");
 rul2sub.addItem("Condition");
 rul2sub.addItem("Connect");
rul2sub.addItem("Contain");
 rul2sub.addItem("Control Magnitude");
 rul2sub.addItem("Convert");
rul2sub.addItem("Couple");
rul2sub.addItem("Decrease");
rul2sub.addItem("Decrement");
rul2sub.addItem("Detect");
rul2sub.addItem("Display");
 rul2sub.addItem("Distribute");
 rul2sub.addItem("Divide");
 rul2sub.addItem("Export");
 rul2sub.addItem("Extract");
 rul2sub.addItem("Guide");
 rul2sub.addItem("Import");
 rul2sub.addItem("Increase");
rul2sub.addItem("Increment");
rul2sub.addItem("Indicate");
rul2sub.addItem("Inhibit");
rul2sub.addItem("Join");
rul2sub.addItem("Link");
 rul2sub.addItem("Measure");
rul2sub.addItem("Mix");
 rul2sub.addItem("Position");
 rul2sub.addItem("Prevent");
rul2sub.addItem("Process");
rul2sub.addItem("Provision");
 rul2sub.addItem("Regulate");
rul2sub.addItem("Remove");
 rul2sub.addItem("Rotate");
rul2sub.addItem("Secure");
 rul2sub.addItem("Sense");
 rul2sub.addItem("Seperate");
 rul2sub.addItem("Shape");
 rul2sub.addItem("Signal");
 rul2sub.addItem("Stabilize");
 rul2sub.addItem("Stop");
 rul2sub.addItem("Store");
 rul2sub.addItem("Supply");
 rul2sub.addItem("Support");
 rul2sub.addItem("Track");
 rul2sub.addItem("Transfer");
rul2sub.addItem("Translate");
 rul2sub.addItem("Transmit");
 rul2sub.addItem("Transport");
 rul2sub.setSelectedItem (su2);
```

```
rul3sub = new JComboBox();
rul3sub.addItem("");
rul3sub.addItem("Actuate");
 rul3sub.addItem("Allow DOF");
 rul3sub.addItem("Branch");
rul3sub.addItem("Change");
 rul3sub.addItem("Channel");
rul3sub.addItem("Collect");
 rul3sub.addItem("Condition");
 rul3sub.addItem("Connect");
rul3sub.addItem("Contain");
 rul3sub.addItem("Control Magnitude");
 rul3sub.addItem("Convert");
rul3sub.addItem("Couple");
rul3sub.addItem("Decrease");
rul3sub.addItem("Decrement");
rul3sub.addItem("Detect");
rul3sub.addItem("Display");
 rul3sub.addItem("Distribute");
 rul3sub.addItem("Divide");
 rul3sub.addItem("Export");
 rul3sub.addItem("Extract");
 rul3sub.addItem("Guide");
 rul3sub.addItem("Import");
 rul3sub.addItem("Increase");
rul3sub.addItem("Increment");
rul3sub.addItem("Indicate");
rul3sub.addItem("Inhibit");
rul3sub.addItem("Join");
rul3sub.addItem("Link");
 rul3sub.addItem("Measure");
rul3sub.addItem("Mix");
 rul3sub.addItem("Position");
 rul3sub.addItem("Prevent");
rul3sub.addItem("Process");
rul3sub.addItem("Provision");
 rul3sub.addItem("Regulate");
rul3sub.addItem("Remove");
 rul3sub.addItem("Rotate");
rul3sub.addItem("Secure");
 rul3sub.addItem("Sense");
 rul3sub.addItem("Seperate");
 rul3sub.addItem("Shape");
 rul3sub.addItem("Signal");
 rul3sub.addItem("Stabilize");
 rul3sub.addItem("Stop");
 rul3sub.addItem("Store");
 rul3sub.addItem("Supply");
 rul3sub.addItem("Support");
 rul3sub.addItem("Track");
 rul3sub.addItem("Transfer");
rul3sub.addItem("Translate");
 rul3sub.addItem("Transmit");
 rul3sub.addItem("Transport");
 rul3sub.setSelectedItem (su3);

setLayout(new GridBagLayout());
 GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
gridBagConstraintsx00.gridx = 0;
gridBagConstraintsx00.gridy = 0;
gridBagConstraintsx00.insets = new Insets(5,5,5,5);
gridBagConstraintsx00.gridwidth = 3;
gridBagConstraintsx00.fill = GridBagConstraints.BOTH;
p.add(jLabel0, gridBagConstraintsx00);
GridBagConstraints gridBagConstraintsx01 = new GridBagConstraints();
gridBagConstraintsx01.gridx = 0;
gridBagConstraintsx01.gridy = 1;
gridBagConstraintsx01.insets = new Insets(5,5,5,5);
p.add(inpu, gridBagConstraintsx01);
GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
gridBagConstraintsx02.gridx = 1;
gridBagConstraintsx02.gridy = 1;
gridBagConstraintsx02.insets = new Insets(5,5,5,5);
gridBagConstraintsx02.gridwidth = 1;
gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
```

```
p.add(subf, gridBagConstraintsx02);
GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
gridBagConstraintsx03.gridx = 2;
gridBagConstraintsx03.gridy = 1;
gridBagConstraintsx03.insets = new Insets(5,5,5,5);
p.add(outpu, gridBagConstraintsx03);

GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
gridBagConstraintsx04.gridx = 0;
gridBagConstraintsx04.gridy = 2;
gridBagConstraintsx04.insets = new Insets(5,5,5,5);
gridBagConstraintsx04.gridwidth = 1;
gridBagConstraintsx04.fill = GridBagConstraints.BOTH;
p.add(rul1in, gridBagConstraintsx04);
GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
gridBagConstraintsx05.gridx = 1;
gridBagConstraintsx05.gridy = 2;
gridBagConstraintsx05.insets = new Insets(5,5,5,5);
gridBagConstraintsx05.gridwidth = 1;
gridBagConstraintsx05.fill = GridBagConstraints.BOTH;
p.add(rul1sub, gridBagConstraintsx05);
GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
gridBagConstraintsx06.gridx = 2;
gridBagConstraintsx06.gridy = 2;
gridBagConstraintsx06.insets = new Insets(5,5,5,5);
p.add(rul1out, gridBagConstraintsx06);

GridBagConstraints gridBagConstraintsx06b = new GridBagConstraints();
gridBagConstraintsx06b.gridx = 1;
gridBagConstraintsx06b.gridy = 3;
gridBagConstraintsx06b.insets = new Insets(5,5,5,5);
gridBagConstraintsx06b.gridwidth = 1;
gridBagConstraintsx06b.fill = GridBagConstraints.BOTH;
p.add(and1, gridBagConstraintsx06b);

GridBagConstraints gridBagConstraintsx07 = new GridBagConstraints();
gridBagConstraintsx07.gridx = 0;
gridBagConstraintsx07.gridy = 4;
gridBagConstraintsx07.insets = new Insets(5,5,5,5);
gridBagConstraintsx07.gridwidth = 1;
gridBagConstraintsx07.fill = GridBagConstraints.BOTH;
p.add(rul2in, gridBagConstraintsx07);
GridBagConstraints gridBagConstraintsx08 = new GridBagConstraints();
gridBagConstraintsx08.gridx = 1;
gridBagConstraintsx08.gridy = 4;
gridBagConstraintsx08.insets = new Insets(5,5,5,5);
gridBagConstraintsx08.gridwidth = 1;
gridBagConstraintsx08.fill = GridBagConstraints.BOTH;
p.add(rul2sub, gridBagConstraintsx08);
GridBagConstraints gridBagConstraintsx09 = new GridBagConstraints();
gridBagConstraintsx09.gridx = 2;
gridBagConstraintsx09.gridy = 4;
gridBagConstraintsx09.insets = new Insets(5,5,5,5);
p.add(rul2out, gridBagConstraintsx09);

GridBagConstraints gridBagConstraintsx09b = new GridBagConstraints();
gridBagConstraintsx09b.gridx = 1;
gridBagConstraintsx09b.gridy = 5;
gridBagConstraintsx09b.insets = new Insets(5,5,5,5);
gridBagConstraintsx09b.gridwidth = 1;
gridBagConstraintsx09b.fill = GridBagConstraints.BOTH;
p.add(and2, gridBagConstraintsx09b);

GridBagConstraints gridBagConstraintsx010 = new GridBagConstraints();
gridBagConstraintsx010.gridx = 0;
gridBagConstraintsx010.gridy = 6;
gridBagConstraintsx010.insets = new Insets(5,5,5,5);
gridBagConstraintsx010.gridwidth = 1;
gridBagConstraintsx010.fill = GridBagConstraints.BOTH;
p.add(rul3in, gridBagConstraintsx010);
GridBagConstraints gridBagConstraintsx011 = new GridBagConstraints();
gridBagConstraintsx011.gridx = 1;
gridBagConstraintsx011.gridy = 6;
gridBagConstraintsx011.insets = new Insets(5,5,5,5);
gridBagConstraintsx011.gridwidth = 1;
gridBagConstraintsx011.fill = GridBagConstraints.BOTH;
```

```
                       p.add(rul3sub, gridBagConstraintsx011);
                       GridBagConstraints gridBagConstraintsx012 = new GridBagConstraints();
                       gridBagConstraintsx012.gridx = 2;
                       gridBagConstraintsx012.gridy = 6;
                       gridBagConstraintsx012.insets = new Insets(5,5,5,5);
                       p.add(rul3out, gridBagConstraintsx012);

                       GridBagConstraints gridBagConstraintsx013 = new GridBagConstraints();
                       gridBagConstraintsx013.gridx = 0;
                       gridBagConstraintsx013.gridy = 7;
                       gridBagConstraintsx013.insets = new Insets(5,5,5,5);
                       p.add(baseid1, gridBagConstraintsx013);
                       GridBagConstraints gridBagConstraintsx014 = new GridBagConstraints();
                       gridBagConstraintsx014.gridx = 1;
                       gridBagConstraintsx014.gridy = 7;
                       gridBagConstraintsx014.insets = new Insets(5,5,5,5);
                       gridBagConstraintsx014.gridwidth = 2;
                       gridBagConstraintsx014.fill = GridBagConstraints.BOTH;
                       p.add(baseid, gridBagConstraintsx014);
                       GridBagConstraints gridBagConstraintsx015 = new GridBagConstraints();
                       gridBagConstraintsx015.gridx = 0;
                       gridBagConstraintsx015.gridy = 8;
                       gridBagConstraintsx015.insets = new Insets(5,5,5,5);
                        gridBagConstraintsx015.gridwidth = 2;
                       gridBagConstraintsx015.fill = GridBagConstraints.BOTH;
                       p.add(inser, gridBagConstraintsx015);
                       GridBagConstraints gridBagConstraintsx016 = new GridBagConstraints();
                       gridBagConstraintsx016.gridx = 2;
                       gridBagConstraintsx016.gridy = 8;
                       gridBagConstraintsx016.insets = new Insets(5,5,5,5);
                       p.add(main1, gridBagConstraintsx016);
                        this.getContentPane().add(p);
                        setVisible(true);
               }
               public void actionPerformed(ActionEvent ae)
               {
                       String str = ae.getActionCommand();
                       if(str.equals("Update Selected Rule"))
                       {
                              String basid = baseid.getText();
                              try
                              {
                                     int response = JOptionPane.showConfirmDialog (this,"Are you
sure you want to update the existing rule?","Confirm Update
Dialog",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
                                       if(response == JOptionPane.YES_OPTION)
                                       {
                                           stmt.executeUpdate("update rules set input1 = \'" +
rul1in.getSelectedItem() + "\', input2 = \'" + rul2in.getSelectedItem() + "\', input3 = \'"+
rul3in.getSelectedItem() + "\', subfunction =  \'"+ rul1sub.getSelectedItem() + "\', subfunction2
= \'" + rul2sub.getSelectedItem()+ "\', subfunction3 = \'"+ rul3sub.getSelectedItem()+
"\',output1 = \'" + rul1out.getSelectedItem() + "\',output2 = \'" + rul2out.getSelectedItem()+
"\', output3 = \'" + rul3out.getSelectedItem()+"\' where base_id =  \'" + basid+"\';");
                                       }
                                       this.dispose();
                                       viewRules vr = new viewRules();
                                       vr.Gen1 ();
                                       stmt.close();
                                       conn.close ();
                              }
                              catch(Exception ex)
                              {
                                     System.out.println(ex);
                                     return;
                              }
                       }
                       else
                       {
                              try
                              {
                                   this.dispose();
                                   main2 mm = new main2();
                                   mm.Gen2 ();
                                   stmt.close();
                                   conn.close();
                              }
                        catch(Exception ex)
                        {}
                }
        }
}
```

21) viewRules.java

```
// To view the graph grammar rules that are stored in the Design Repository
import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.sql.*;
import java.net.*;
import javax.imageio.*;
import java.io.*;
import java.awt.event.*;

public class viewRules extends JFrame implements ActionListener
{
    JPanel p;
    JLabel jLabe20,compid,baseid,and1,compno,and2;
    JButton back,front,main1,delet,updat;
    String driver,url,aa,comp,bas,inp1,su1,oup1,inp2,su2,oup2,inp3,su3,oup3,dfa1;
    Connection conn;
    Statement stmt,stmt2,stmt3;
    ResultSet rs,rs2;
    int key=1,record = 0,nocomp=0,delno;
    public viewRules()
    {
        p = new JPanel(new GridBagLayout());
        this.setSize(700,700);
        this.setResizable(false);
        this.setTitle("View Rule(s) Menu");
    }
    public void Gen1()
    {
        jLabe20 = new JLabel("View Rule(s)");
        jLabe20.setFont(new Font("System",Font.BOLD,25));
        and1 = new JLabel("AND");
        and2 = new JLabel("AND");
        and1.setFont(new Font("System",Font.BOLD,15));
        and2.setFont(new Font("System",Font.BOLD,15));
        compid = new JLabel("Then retrieve all components with Base ID: ");
        compid.setFont(new Font("System",Font.BOLD,15));
        compno = new JLabel("Number of Component(s) utilizing this rule: ");
        compno.setFont(new Font("System",Font.BOLD,15));
        baseid = new JLabel("Base ID:");
        delet = new JButton("Delete Rule");
        delet.addActionListener (this);
        updat = new JButton("Update Rule");
        updat.addActionListener (this);
        JButton back = new JButton("<<");
        back.addActionListener (this);
        if(key <= 1) back.setEnabled (false);
        JButton front = new JButton(">>");
        front.addActionListener (this);
        JButton main1 = new JButton("Design Repository Menu");
        main1.addActionListener (this);
        try
        {
            driver = "com.mysql.jdbc.Driver";
            url = "jdbc:mysql://localhost:3306/test";
            Class.forName(driver);
            conn =DriverManager.getConnection(url,"root","pennstate");
            stmt = conn.createStatement();
            stmt2 = conn.createStatement ();
            stmt3 = conn.createStatement ();
            rs = stmt.executeQuery("select * from rules");
            while(rs.next ())
            {
```

```
                        record++;
                    }
                    if(key >= record) front.setEnabled (false);
                    rs = stmt.executeQuery("select * from rules where ruleno = " + key + " order
by ruleno ASC;");
                    while(rs.next ())
                    {
                        nocomp = 0;
                        bas = rs.getString ("base_id");
                        inp1 = rs.getString("input1");
                        oup1 = rs.getString ("output1");
                        su1 = rs.getString ("subfunction");
                        inp2 = rs.getString("input2");
                        oup2 = rs.getString ("output2");
                        su2 = rs.getString ("subfunction2");
                        inp3 = rs.getString("input3");
                        oup3 = rs.getString ("output3");
                        su3 = rs.getString ("subfunction3");
                        rs2 = stmt2.executeQuery ("select * from DFA where baseid = \'" + bas +
"\';");
                        while(rs2.next ())
                        {
                            nocomp++;
                            delno = Integer.parseInt(rs2.getString("number"));
                        }
                        break;
                     }
                }
                catch(Exception ex)
                {
                    System.out.println(ex);
                    return;
                }
                GridBagConstraints gridBagConstraintsx00 = new GridBagConstraints();
                gridBagConstraintsx00.gridx = 2;
                gridBagConstraintsx00.gridy = 0;
                gridBagConstraintsx00.gridwidth = 2;
                gridBagConstraintsx00.insets = new Insets(5,5,5,5);
                p.add(jLabe20,gridBagConstraintsx00);
                GridBagConstraints gridBagConstraintsx02 = new GridBagConstraints();
                gridBagConstraintsx02.gridx = 0;
                gridBagConstraintsx02.gridy = 1;
                gridBagConstraintsx02.insets = new Insets(5,5,5,5);
                gridBagConstraintsx02.gridwidth = 1;
                gridBagConstraintsx02.fill = GridBagConstraints.BOTH;
                p.add(new JLabel("If Input1 is: "), gridBagConstraintsx02);
                GridBagConstraints gridBagConstraintsx03 = new GridBagConstraints();
                gridBagConstraintsx03.gridx = 1;
                gridBagConstraintsx03.gridy = 1;
                gridBagConstraintsx03.insets = new Insets(5,5,5,5);
                gridBagConstraintsx03.gridwidth = 1;
                gridBagConstraintsx03.fill = GridBagConstraints.BOTH;
                p.add(new JLabel(inp1), gridBagConstraintsx03);
                GridBagConstraints gridBagConstraintsx04 = new GridBagConstraints();
                gridBagConstraintsx04.gridx = 2;
                gridBagConstraintsx04.gridy = 1;
                gridBagConstraintsx04.insets = new Insets(5,5,5,5);
                gridBagConstraintsx04.gridwidth = 1;
                gridBagConstraintsx04.fill = GridBagConstraints.BOTH;
                p.add (new JLabel("Subfunction1 is: "),gridBagConstraintsx04);
                GridBagConstraints gridBagConstraintsx05 = new GridBagConstraints();
                gridBagConstraintsx05.gridx = 3;
                gridBagConstraintsx05.gridy = 1;
                gridBagConstraintsx05.insets = new Insets(5,5,5,5);
                gridBagConstraintsx05.gridwidth = 1;
                gridBagConstraintsx05.fill = GridBagConstraints.BOTH;
                p.add (new JLabel(su1),gridBagConstraintsx05);
                GridBagConstraints gridBagConstraintsx05a = new GridBagConstraints();
                gridBagConstraintsx05a.gridx = 4;
                gridBagConstraintsx05a.gridy = 1;
                gridBagConstraintsx05a.insets = new Insets(5,5,5,5);
                gridBagConstraintsx05a.gridwidth = 1;
                gridBagConstraintsx05a.fill = GridBagConstraints.BOTH;
                p.add (new JLabel("Output1 is: "),gridBagConstraintsx05a);
                GridBagConstraints gridBagConstraintsx06 = new GridBagConstraints();
                gridBagConstraintsx06.gridx = 5;
```

```
gridBagConstraintsx06.gridy = 1;
gridBagConstraintsx06.insets = new Insets(5,5,5,5);
gridBagConstraintsx06.gridwidth = 1;
gridBagConstraintsx06.fill = GridBagConstraints.BOTH;
p.add (new JLabel(oup1),gridBagConstraintsx06);
if((!inp2.equals (""))&&(!oup2.equals (""))&&(!su2.equals ("")))
{
    GridBagConstraints gridBagConstraintsx07 = new GridBagConstraints();
    gridBagConstraintsx07.gridx = 2;
    gridBagConstraintsx07.gridy = 2;
    gridBagConstraintsx07.insets = new Insets(5,5,5,5);
    gridBagConstraintsx07.gridwidth = 2;
    gridBagConstraintsx07.fill = GridBagConstraints.BOTH;
    p.add (and2,gridBagConstraintsx07);
    GridBagConstraints gridBagConstraintsx08 = new GridBagConstraints();
    gridBagConstraintsx08.gridx = 0;
    gridBagConstraintsx08.gridy = 3;
    gridBagConstraintsx08.insets = new Insets(5,5,5,5);
    gridBagConstraintsx08.gridwidth = 1;
    gridBagConstraintsx08.fill = GridBagConstraints.BOTH;
    p.add (new JLabel("If Input2 is: "),gridBagConstraintsx08);
    GridBagConstraints gridBagConstraintsx09 = new GridBagConstraints();
    gridBagConstraintsx09.gridx = 1;
    gridBagConstraintsx09.gridy = 3;
    gridBagConstraintsx09.insets = new Insets(5,5,5,5);
    gridBagConstraintsx09.gridwidth = 1;
    gridBagConstraintsx09.fill = GridBagConstraints.BOTH;
    p.add (new JLabel(inp2),gridBagConstraintsx09);
    GridBagConstraints gridBagConstraintsx010 = new GridBagConstraints();
    gridBagConstraintsx010.gridx = 2;
    gridBagConstraintsx010.gridy = 3;
    gridBagConstraintsx010.insets = new Insets(5,5,5,5);
    gridBagConstraintsx010.gridwidth = 1;
    gridBagConstraintsx010.fill = GridBagConstraints.BOTH;
    p.add (new JLabel("Subfunction2 is: "),gridBagConstraintsx010);
    GridBagConstraints gridBagConstraintsx011 = new GridBagConstraints();
    gridBagConstraintsx011.gridx = 3;
    gridBagConstraintsx011.gridy = 3;
    gridBagConstraintsx011.insets = new Insets(5,5,5,5);
    gridBagConstraintsx011.gridwidth = 1;
    gridBagConstraintsx011.fill = GridBagConstraints.BOTH;
    p.add (new JLabel(su2),gridBagConstraintsx011);
    GridBagConstraints gridBagConstraintsx012 = new GridBagConstraints();
    gridBagConstraintsx012.gridx = 4;
    gridBagConstraintsx012.gridy = 3;
    gridBagConstraintsx012.insets = new Insets(5,5,5,5);
    gridBagConstraintsx012.gridwidth = 1;
    gridBagConstraintsx012.fill = GridBagConstraints.BOTH;
    p.add (new JLabel("Output2 is:"),gridBagConstraintsx012);
    GridBagConstraints gridBagConstraintsx013 = new GridBagConstraints();
    gridBagConstraintsx013.gridx = 5;
    gridBagConstraintsx013.gridy = 3;
    gridBagConstraintsx013.insets = new Insets(5,5,5,5);
    gridBagConstraintsx013.gridwidth = 1;
    gridBagConstraintsx013.fill = GridBagConstraints.BOTH;
    p.add (new JLabel(oup2),gridBagConstraintsx013);
    if((!inp3.equals (""))&&(!oup3.equals (""))&&(!su3.equals ("")))
    {
        GridBagConstraints gridBagConstraintsx014 = new GridBagConstraints();
        gridBagConstraintsx014.gridx = 2;
        gridBagConstraintsx014.gridy = 4;
        gridBagConstraintsx014.insets = new Insets(5,5,5,5);
        gridBagConstraintsx014.gridwidth = 2;
        gridBagConstraintsx014.fill = GridBagConstraints.BOTH;
        p.add (and1,gridBagConstraintsx014);
        GridBagConstraints gridBagConstraintsx015 = new GridBagConstraints();
        gridBagConstraintsx015.gridx = 0;
        gridBagConstraintsx015.gridy = 5;
        gridBagConstraintsx015.insets = new Insets(5,5,5,5);
        gridBagConstraintsx015.gridwidth = 1;
        gridBagConstraintsx015.fill = GridBagConstraints.BOTH;
        p.add (new JLabel("If Input3 is: "),gridBagConstraintsx015);
        GridBagConstraints gridBagConstraintsx016 = new GridBagConstraints();
        gridBagConstraintsx016.gridx = 1;
        gridBagConstraintsx016.gridy = 5;
        gridBagConstraintsx016.insets = new Insets(5,5,5,5);
        gridBagConstraintsx016.gridwidth = 1;
        gridBagConstraintsx016.fill = GridBagConstraints.BOTH;
        p.add (new JLabel(inp3),gridBagConstraintsx016);
        GridBagConstraints gridBagConstraintsx017 = new GridBagConstraints();
        gridBagConstraintsx017.gridx = 2;
        gridBagConstraintsx017.gridy = 5;
        gridBagConstraintsx017.insets = new Insets(5,5,5,5);
        gridBagConstraintsx017.gridwidth = 1;
        gridBagConstraintsx017.fill = GridBagConstraints.BOTH;
        p.add (new JLabel("Subfunction3 is: "),gridBagConstraintsx017);
        GridBagConstraints gridBagConstraintsx018 = new GridBagConstraints();
        gridBagConstraintsx018.gridx = 3;
        gridBagConstraintsx018.gridy = 5;
        gridBagConstraintsx018.insets = new Insets(5,5,5,5);
        gridBagConstraintsx018.gridwidth = 1;
        gridBagConstraintsx018.fill = GridBagConstraints.BOTH;
        p.add (new JLabel(su3),gridBagConstraintsx018);
        GridBagConstraints gridBagConstraintsx019 = new GridBagConstraints();
        gridBagConstraintsx019.gridx = 4;
        gridBagConstraintsx019.gridy = 5;
        gridBagConstraintsx019.insets = new Insets(5,5,5,5);
        gridBagConstraintsx019.gridwidth = 1;
        gridBagConstraintsx019.fill = GridBagConstraints.BOTH;
        p.add (new JLabel("Output3 is: "),gridBagConstraintsx019);
        GridBagConstraints gridBagConstraintsx021 = new GridBagConstraints();
        gridBagConstraintsx021.gridx = 5;
        gridBagConstraintsx021.gridy = 5;
        gridBagConstraintsx021.insets = new Insets(5,5,5,5);
        gridBagConstraintsx021.gridwidth = 1;
        gridBagConstraintsx021.fill = GridBagConstraints.BOTH;
        p.add (new JLabel(oup3),gridBagConstraintsx021);
    }
}
    GridBagConstraints gridBagConstraintsx020 = new GridBagConstraints();
    gridBagConstraintsx020.gridx = 0;
    gridBagConstraintsx020.gridy = 6;
    gridBagConstraintsx020.insets = new Insets(5,5,5,5);
    gridBagConstraintsx020.gridwidth = 5;
    gridBagConstraintsx020.fill = GridBagConstraints.BOTH;
    p.add (compid,gridBagConstraintsx020);
    GridBagConstraints gridBagConstraintsx021 = new GridBagConstraints();
    gridBagConstraintsx021.gridx = 5;
    gridBagConstraintsx021.gridy = 6;
    gridBagConstraintsx021.insets = new Insets(5,5,5,5);
    gridBagConstraintsx021.gridwidth = 1;
    gridBagConstraintsx021.fill = GridBagConstraints.BOTH;
    p.add (new JLabel(bas),gridBagConstraintsx021);

    GridBagConstraints gridBagConstraintsx022 = new GridBagConstraints();
    gridBagConstraintsx022.gridx = 0;
    gridBagConstraintsx022.gridy = 7;
    gridBagConstraintsx022.insets = new Insets(5,5,5,5);
    gridBagConstraintsx022.gridwidth = 5;
    gridBagConstraintsx022.fill = GridBagConstraints.BOTH;
    p.add (compno,gridBagConstraintsx022);
    GridBagConstraints gridBagConstraintsx023 = new GridBagConstraints();
    gridBagConstraintsx023.gridx = 5;
    gridBagConstraintsx023.gridy = 7;
    gridBagConstraintsx023.insets = new Insets(5,5,5,5);
    gridBagConstraintsx023.gridwidth = 1;
    gridBagConstraintsx023.fill = GridBagConstraints.BOTH;
    p.add (new JLabel(Integer.toString(nocomp)),gridBagConstraintsx023);
    GridBagConstraints gridBagConstraintsx024 = new GridBagConstraints();
    gridBagConstraintsx024.gridx = 0;
    gridBagConstraintsx024.gridy = 8;
    gridBagConstraintsx024.insets = new Insets(5,5,5,5);
    gridBagConstraintsx024.gridwidth = 1;
    gridBagConstraintsx024.fill = GridBagConstraints.BOTH;
    p.add (back,gridBagConstraintsx024);
    GridBagConstraints gridBagConstraintsx025 = new GridBagConstraints();
    gridBagConstraintsx025.gridx = 1;
    gridBagConstraintsx025.gridy = 8;
    gridBagConstraintsx025.insets = new Insets(5,5,5,5);
    gridBagConstraintsx025.gridwidth = 1;
    gridBagConstraintsx025.fill = GridBagConstraints.BOTH;
    p.add (updat,gridBagConstraintsx025);
```

```
                    GridBagConstraints gridBagConstraintsx026 = new GridBagConstraints();
                    gridBagConstraintsx026.gridx = 2;
                    gridBagConstraintsx026.gridy = 8;
                    gridBagConstraintsx026.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx026.gridwidth = 2;
                    gridBagConstraintsx026.fill = GridBagConstraints.BOTH;
                    p.add (main1,gridBagConstraintsx026);
                    GridBagConstraints gridBagConstraintsx027 = new GridBagConstraints();
                    gridBagConstraintsx027.gridx = 4;
                    gridBagConstraintsx027.gridy = 8;
                    gridBagConstraintsx027.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx027.gridwidth = 1;
                    gridBagConstraintsx027.fill = GridBagConstraints.BOTH;
                    p.add (delet,gridBagConstraintsx027);
                    GridBagConstraints gridBagConstraintsx028 = new GridBagConstraints();
                    gridBagConstraintsx028.gridx = 5;
                    gridBagConstraintsx028.gridy = 8;
                    gridBagConstraintsx028.insets = new Insets(5,5,5,5);
                    gridBagConstraintsx028.gridwidth = 1;
                    gridBagConstraintsx028.fill = GridBagConstraints.BOTH;
                    p.add (front,gridBagConstraintsx028);
                    this.getContentPane().add(p);
                    setVisible(true);
            }
            public void actionPerformed(ActionEvent ae)
            {
               String str = ae.getActionCommand();
               if(str.equals("Design Repository Menu"))
               {
                    this.dispose();
                    main2 mm = new main2();
                    mm.Gen2 ();
                }
                if(str.equals("Update Rule"))
               {
                    this.dispose();
                    updateRules mm = new updateRules();
                    mm.init1 (key);
                }
                if(str.equals(">>"))
               {
                    key = key + 1;
                    if(key > record)
                    {
                        key = record;
                    }
                    if((key >=1)&&(key <= record))
                    {
                        this.dispose();
                        p.removeAll();
                        Gen1();
                    }
                }
                else if(str.equals("<<"))
               {
                    if(key > 1)
                    {
                        key = key - 1;
                    }
                    if(key >= 1)
                    {
                       this.dispose();
                        p.removeAll();
                        Gen1();
                    }
                }
               if(str.equals("Delete Rule"))
               {
                    int response = JOptionPane.showConfirmDialog (this,"Are you sure you want to
delete the " + nocomp + " component(s) associated with this rule?","Confirm Delete
Dialog",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
                    if(response == JOptionPane.YES_OPTION)
                    {
                        try
                        {
                            stmt3.executeUpdate("update DFA set number = number - " + nocomp + "
where number >= " + delno + ";");
                            stmt3.executeUpdate("delete from DFA where baseid = \'" + bas +
"\';");

                            stmt3.executeUpdate("delete from rules where base_id = \'" + bas +
"\';");

                            stmt3.close();
                            stmt2.close();
                            rs2.close();
                            key = key + 1;
                            if(key > record)
                            {
                                key = record;
                            }
                            if((key >=1)&&(key <= record))
                            {
                                this.dispose();
                                p.removeAll();
                                Gen1();
                            }
                        }
                        catch(Exception ex)
                        {
                            System.out.println(ex);
                        }
                    }
                }
            }
        }
```