UNIVERSITI TEKNOLOGI PETRONAS

Approval by Supervisor

The undersigned certify that they have read, and recommended to The Postgraduate Studies Program for acceptance, a thesis entitled "**Function-Based Computer Aided Conceptual Design Support Tool**" submitted by **Dereje Engida Woldemichael** for the fulfillment of the requirements for the degree of **Doctor of Philosophy in Mechanical Engineering**.

____
Date

Signature	:	______________________________
Main Supervisor	:	A.P. Dr. Fakhruldin Mohd Hashim
Date	:	______________________________

UNIVERSITI TEKNOLOGI PETRONAS

Function-Based Computer Aided Conceptual Design Support Tool

By

Dereje Engida Woldemichael

A THESIS

SUBMITTED TO THE POSTGRADUATE STUDIES PROGRAMME

AS A REQUIREMENT FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

MECHANICAL ENGINEERING PROGRAMME

BANDAR SERI ISKANDAR,

PERAK

NOVEMBER, 2009

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any degree at Universiti Teknologi PETRONAS or other institutions.

Signature	:	
Name	:	Dereje Engida Woldemichael
Date	:	

DEDICATION

To my Father Engida Woldemichael Endaylalu and Uncle Tirfe Woldeamanuel Woldemariam

ACKNOWLEDGEMENTS

First of all I would like to thank my savior the Almighty GOD for giving me the strength and ability to achieve what I have done and guide me throughout my life.

I am deeply indebted to my supervisor Assoc. Prof. Dr. Fakhruldin Mohd Hashim for his invaluable advices and exceptional guidance during my study.

I would also like to extend thanks to the faculty members and postgraduate students who have participated in verification and validation of CDST for their time, suggestions and comments, especially Dr. Syed Ihtsham Ul Haq Gilani, Dr. Dayang Rohaya Binti Awang Rambli, Dr. Suziah Binti Sulaiman , Mr Rahmat Iskandar Khairul Shazi, Mr Azman Bin Zainuddin, Mr Chanyalew Taye, Mr Mesfin Gizaw, and Mr Tamiru Alemu. Special thanks to Mr. Johan Lindberg from CLIPS support group who helped me to learn the language. Thanks to all who have supported me in one way or another, especially fellow Ethiopians in UTP.

Finally, thanks to all my family for their support and encouragement. Special thanks is reserved to my wife Kidist Bisrat who take care of our little baby Selam alone; without her patience, constant support, and encouragement my achievements would not be possible.

ABSTRACT

Conceptual design is considered as the most critical and important phase of design process. It is the stage where product's fundamental features are determined, large proportion of the lifecycle cost of the product is committed, and other major decisions are made, which have significant impact on the downstream design and related manufacturing processes. It is a knowledge intensive process where diverse knowledge and several years of experience are put together to design quality and cost effective products. Unfortunately, computer support systems for this phase are lagging behind compared to the currently available commercial computer aided design (CAD) tools for the later stage of design to reduce the designers workload and product development time.

The overall goal of this research is to provide designers with computational tool that support conceptual design process. To achieve this goal a methodology that integrates systematic design approach with knowledge-based system is proposed in this thesis. Accordingly, a framework of computer based computational tool known as conceptual design support tool (CDST) is developed using the proposed methodology. The tool assists designers in performing functional modeling by providing standard vocabularies of functions in the form of function library, generate concepts stored in the database from previous designs, display the generated concepts on the morphology chart, combine the concepts and evaluate the concepts variants. Concepts from subsea processing equipment design have been collected and saved in the database. The tool also accepts new concepts from the designer through its knowledge acquisition system to be saved in the database for future use. In doing so, it is possible to integrate human creativity with data handling capabilities of computers to perform conceptual design more efficiently than solely manual design. The tool can also be used as a knowledge management system to preserve expert's knowledge and train novice designers. The applicability of the proposed methodology and developed tool is illustrated and validated by using a case study and validation test conducted by independent evaluators.

ABSTRAK

Reka bentuk konseptual adalah dianggap sebagai fasa paling akut dan penting bagi proses rekabentuk. Ia adalah peringkat di mana ciri-ciri utama produk ditentukan, peruntukan besar bagi kos kitaran hasil keluaran terlibat, dan keputusan-keputusan utama lain dibuat, yang mempunyai kesan mendalam ke akhir rekabentuk dan proses pembuatan. Ia satu ilmu proses intensif di mana pengetahuan pelbagai dan pengalaman yang panjang digabungkan untuk membentuk hasil akhir yang berkualiti dan kos yang berpatutan. Malangnya, sistem-sistem sokongan komputer untuk fasa ini agak ketinggalan berbanding dengan alat reka bentuk terbantu komputer (CAD) komersial yang kini boleh didapati banyak dipasaran terutamanya alat bantu komputer untuk peringkat akhir reka bentuk. Dengan ini dapat mengurangkan beban kerja pereka dan masa pembentukan hasil keluaran.

Matlamat keseluruhan kajian ini adalah untuk melengkapkan alat terbantu computer yang menyokong proses rekabentuk konsepsi. Bagi mencapai matlamat ini satu kaedah dicadangkan dalam disertasi ini bagi mengintegrasikan pendekatan reka bentuk sistematik dengan sistem berasaskan pengetahuan. Oleh kerana itu, satu rangka kerja untuk alat terbantu komputer yang dikenali sebagai alat rekabentuk konsepsi terbantu komputer (CDST) adalah dibangunkan menggunakan kaedah yang dicadangkan. Alat terbantu ini dpat membantu pereka dalam melaksanakan model fungsi dengan menyediakan piawaian perbendaharaan kata fungsi dalam bentuk simpanan perpustakaan, menjana konsep-konsep simpanan dalam pangkalan data daripada rekaan-rekaan sebelumnya, paparan konsep-konsep dijanakan pada carta tata kata dan menggabungkan konsep-konsep itu dan menilai konsep-konsep yang berbeza. Konsep-konsep daripada peralatan proses dikumpul dan disimpan dalam pangkalan data. Alat terbantu ini juga menerima konsep-konsep baru daripada pereka melalui sistem tambahan pengetahuan untuk disimpan dalam pangkalan data untuk digunakan kemudian. Dengan ini, ada kemungkinan untuk menyatukan kreativiti insani dengan keupayaan pengendalian data komputer untuk menjalankan rekabentuk konsep dengan lebih cekap daripada rekabentuk tangan. Alat terbantu ini juga boleh digunakan sebagai satu sistem pengurusan pengetahuan untuk mengekalkan pengetahuan pakar dan melatih pereka-pereka baru. Kebolehgunaan kaedah yang dicadangkan dan alat terbantu yang dihasilkan akan digambar dan disahkan dengan menggunakan satu kajian kes dan ujian pengesahan dijalankan oleh para penilai bebas.

TABLE OF CONTENTS

Status of Thesis i
Approval page ii
Title page iii
Declaration iv
Dedication v
Acknowledgements vi
Abstract vii
Abstrak viii
Table of contents x
List of Tables xiv
List of Figures xv
CHAPTER 1: INTRODUCTION 1
1.1 Overview of Engineering Design 1
1.2 Problem Statement 3
1.3 Research Objective 5
1.4 Scope of the Research 7
1.5 Thesis Organization 9
CHAPTER 2: LITERATURE REVIEW 11
2.1 Introduction 11
2.2 Introduction to Conceptual Design Process 11
2.3 AI in Design 14
2.4 Knowledge-Based System 17

2.4.1 The knowledge base ... 18
2.4.2 The inference engine ... 20
2.5 Function-Based Design: A Survey ... 21
2.5.1 Definition of Function ... 21
2.5.2 Functional Representation ... 22
2.5.3 Functional Classifications ... 24
2.6 Function-to-Form Mapping ... 31
2.7 Concept Evaluation Process ... 37
2.7.1 Pugh’s Evaluation Method ... 38
2.7.2 Weighted decision matrix ... 38
2.7.3 Analytical hierarchy process (AHP) ... 42
2.8 Computer Aided Conceptual Design (CACD) Tools: A Survey ... 45
2.9 Summary ... 55
CHAPTER 3: METHODOLOGY ... 56
3.1 Introduction ... 56
3.2 Conceptual Design Process Model ... 57
3.2.1 Functional Modeling ... 60
3.2.2 Concept Generation ... 62
3.2.3 Concept Combination ... 65
3.2.4 Concept Evaluation ... 67
3.3 Model Construction ... 68
3.3.1 Software Design Specifications and Requirements ... 69
3.3.2 Programming Environment ... 72
3.4 Summary ... 73
CHAPTER 4: DEVELOPMENT OF CONCEPTUAL DESIGN SUPPORT TOOL (CDST) ... 75

4.1 Introduction 75

4.2 Development of the Software.. 76

4.2.1 Basic Programming Elements and Knowledge Representation Formalism in CLIPS 76

4.2.2 The Functional Modeling Module 80

4.2.3 Concept Generation Module 83

4.2.4 Concept Combination Module 89

4.2.5 Concept Evaluation Module 92

4.2.6 The GUI Development and its Integration with CLIPS......................... 92

4.3 Integration of the Modules and Initial Testing... 93

4.4 Summary 94

CHAPTER 5: CASE STUDIES.. 96

5.1 Introduction 96

5.2 Conceptual Design of Oil and Gas Separator using CDST........................... 96

5.2.1 Problem Description and Functional Modeling..................................... 96

5.2.2 Concept Generation and Combination.. 101

5.2.3 Concept Evaluation.. 106

5.3 CDST for Subsea Process Equipment Design... 112

CHAPTER 6: VERIFICATION AND VALIDATION OF CDST 119

6.1 Verification of CDST 119

6.2 Validation Test 119

6.2.1 Selection of Evaluators 120

6.2.2 Evaluation Methods and Procedures.. 120

6.2.3 Validation Test Results 121

6.3 Summary 127

CHAPTER 7: CONCLUSION AND RECOMMENDATIONS 128

7.1 Contributions of the Research ... 128

7.2 Critique of the Research ... 131

7.3 Recommendation for Future Research ... 133

REFERENCES ... 135

Appendix A: Source Code and Support Documents ... 144

A.1 Contents of the Attached CD ... 144

A.2 CDST in Different File Formats ... 144

A.3 Source Code ... 145

Appendix B: List of Software used to Develop CDST ... 147

Appendix C: Evaluation questionnaire ... 148

LIST OF TABLES

Table 2.1 Hundal's primary categories and sub-categories of basic functions (Hundal, 1990) 25

Table 2.2 Basic function groups and their extension (Kirschman and Fadel, 1998) ... 26

Table 2.3 Functional basis reconciled function set (adapted from (Hirtz et al., 2002)) 30

Table 2.4 Functional basis reconciled flow set (adapted from (Hirtz et al., 2002)) 31

Table 2.5 Fundamental scale of absolute numbers (Saaty, 1994)........ 40

Table 2.6 Weighted decision matrix 41

Table 2.7 A 5-point and 11-point scale for concept evaluation (Dieter, 2000). 42

Table 2.8 A hypothetical example to demonstrate AHP with three alternative concepts and two selection criteria 44

Table 2.9 Summary of CACD tools........ 53

Table 6.1 Overview of the number of concepts generated by two of the evaluators compared with CDST 122

Table B.1 List of software used to develop CDST 147

Table C.1 CDST evaluation questionnaire 148

LIST OF FIGURES

Figure 1.1 Phases of design process (adapted from (Pahl and Beitz, 1996))................3

Figure 1.2 Impact of decision and availability of computer tools during the design process (Wang et al., 2002)..5

Figure 1.3 Schematic view of the general methodology in developing the CDST........7

Figure 1.4 An example of subsea processing (Sapihie, 2007).....................................9

Figure 2.1 Architectural components of knowledge-based system 18

Figure 2.2 Decomposing overall function into subfunctions (adapted from (Pahl and Beitz, 1996)) 32

Figure 2.3 Functional decomposition and mapping in axiomatic design approach.....34

Figure 2.4 The general form of function logic diagram (adapted from (Sturges et al., 1996))..35

Figure 3.1 Steps in conceptual design process...58

Figure 3.2 The proposed conceptual design model 60

Figure 3.3 Functional structure of hand-held nailer...62

Figure 3.4 Pseudo code to create theoretically possible concept variants 66

Figure 3.5 The waterfall model of software development (adapted from (Fisher, 1991))..69

Figure 3.6 Flow chart of the CDST 71

Figure 4.1 The structure of a knowledge-based system and programming environments used 76

Figure 4.2 A deftemplate construct to define function in CLIPS 80

Figure 4.3 Screenshot of the function library 82

Figure 4.4 A deftemplate construct to define alternative concepts in CLIPS 84

Figure 4.5 Sample alternative concepts representation in the database 85

Figure 4.6 A general syntax for defrule construct 86

Figure 4.7 Sample mapping rule 87

Figure 4.8 Mapping rule to prevent repeated rule firing 87

Figure 4.9 A mapping rule when there is no alterative concept in the database 88

Figure 4.10 Screenshot of alternative concepts input window 89

Figure 4.11 Theoretically possible concept combination rule for 3 subfunctions 91

Figure 4.12 Flow compatible concept combination rule for 3 subfunctions 92

Figure 4.13 Screenshot of CDST welcoming window 93

Figure 5.1 Functional model for three-phase oil and gas separator 97

Figure 5.2 CDST welcoming window 98

Figure 5.3 Screenshot of the function library 99

Figure 5.4 Screen shot of pull down menu to load input functional structure 100

Figure 5.5 Function type selector for concept generation 102

Figure 5.6 Design summary window 103

Figure 5.7 Morphology chart 104

Figure 5.8 Concept variants represented textually 106

Figure 5.9 Schematic representation of the concepts 106

Figure 5.10 Concept selection process main window 107

Figure 5.11 Evaluation criteria input window 107

Figure 5.12 Absolute comparison window with list of concepts for concept variant 3 108

Figure 5.13 Screenshot of concept screening window 109

Figure 5.14 Weight assigning method selection popup window 110

Figure 5.15 Pairwise comparison matrix 110

Figure 5.16 Final concept evaluation using weighted decision matrix 111

Figure 5.17 Schematic view of the selected concept variant 111

Figure 5.18 Flow chart for CDSTsped 113

Figure 5.19 Screenshot of CDSTsped initial window 115

Figure 5.20 Selection of pump type 115

Figure 5.21 Kinetic pump design window 116

Figure 5.22 Viewing alternative concepts for a given subfunction 117

Figure 5.23 Separator design window 118

Figure 6.1 Concepts generated by the evaluators to separate liquid-gas mixture 122

Figure 6.2 Percentage rating for functional testing 123

Figure 6.3 Evaluators' rating for organization of the user interface 124

Figure 6.4 Startup/shutdown and positive test results 124

Figure 6.5 Summary of evaluators rating on general use of CDST 125

Figure 6.6 Evaluators rating on the overall achievement of the program 125

Figure 6.7 Excerpt from the saved design history 126

Figure A.1 An excerpt from CDST source code in CLIPS 145

Figure A.2 An excerpt from the source code of GUI in Python environment 146

CHAPTER 1: INTRODUCTION

1.1 Overview of Engineering Design

Engineering design is the process of devising a system, component, or process to meet perceived needs. Even though humans have been designing products for thousands of years, design research is still going on. This is because of the current competitive market, developing more efficient and new approaches, and dynamic customer requirement for new, cost effective and high quality products. In order to make the design process more effective and efficient, design research aims at developing means to understand the design and develop a support system to enable design process in getting more successful products (Blessing and Chakrabarti, 2009). Product quality, cost and time to market are the key measures for the effectiveness of design process (Ullman, 2003). Careful and detail exploration of alternative options may result in low cost and quality products. However, this requires more time and the process is also knowledge intensive. Thus, the designer needs to be supported with efficient tools to compete in the market.

Although it may be difficult to always border line between different phases, design process can generally be classified into four phases as shown in Figure 1.1 (Pahl and Beitz, 1996, French, 1998). Planning and task clarification is the first phase in which the designer identifies customer needs, collects information about the requirements and come up with requirements list or design specification as an output. The second phase of design is conceptual design, which takes the requirement list as an input and come up with one or more concept variants that can satisfy the requirements. This requires abstracting the essential problems, establishing functional structure, searching for alternative concepts, combining those concepts to form concept variants, and evaluating those concept variants. The selected concept variant is further developed in the embodiment design phase, where preliminary form design, material selection and calculations are done which results in determining the construction structure or overall layout. The last phase of design is the detail design phase in which details of production and operation documents are prepared. Among these phases, conceptual design is considered to be the most critical phase of design. This is because conceptual design is:

i. the most demanding phase from the designer's point of view to generate new solution or make remarkable improvements on the design;

ii. the stage where product's fundamental features are determined with imprecise and incomplete information; and

iii. the stage where large proportion of the product's lifecycle cost is committed and other major decisions are made.

Depending on its originality or innovativeness, a design activity can be classified as original, adaptive, or variant design (Pahl and Beitz, 1996). In original design, new solution principles are invented by selecting and combining known principles and technology, or by inventing completely new technology. When existing or slightly changed tasks are solved using new solution principles, it can also be considered as original design. Original designs usually proceed through all the design phases. In adaptive design, known and established solution principles are adapted to changed requirements, whereas in variant design the sizes and arrangements of parts and assemblies are varied within the limits set by previously designed product structures. In practice, it is often not possible to define precisely the boundaries between the three types of design and the majority of design problems are adaptation and variation of existing designs.

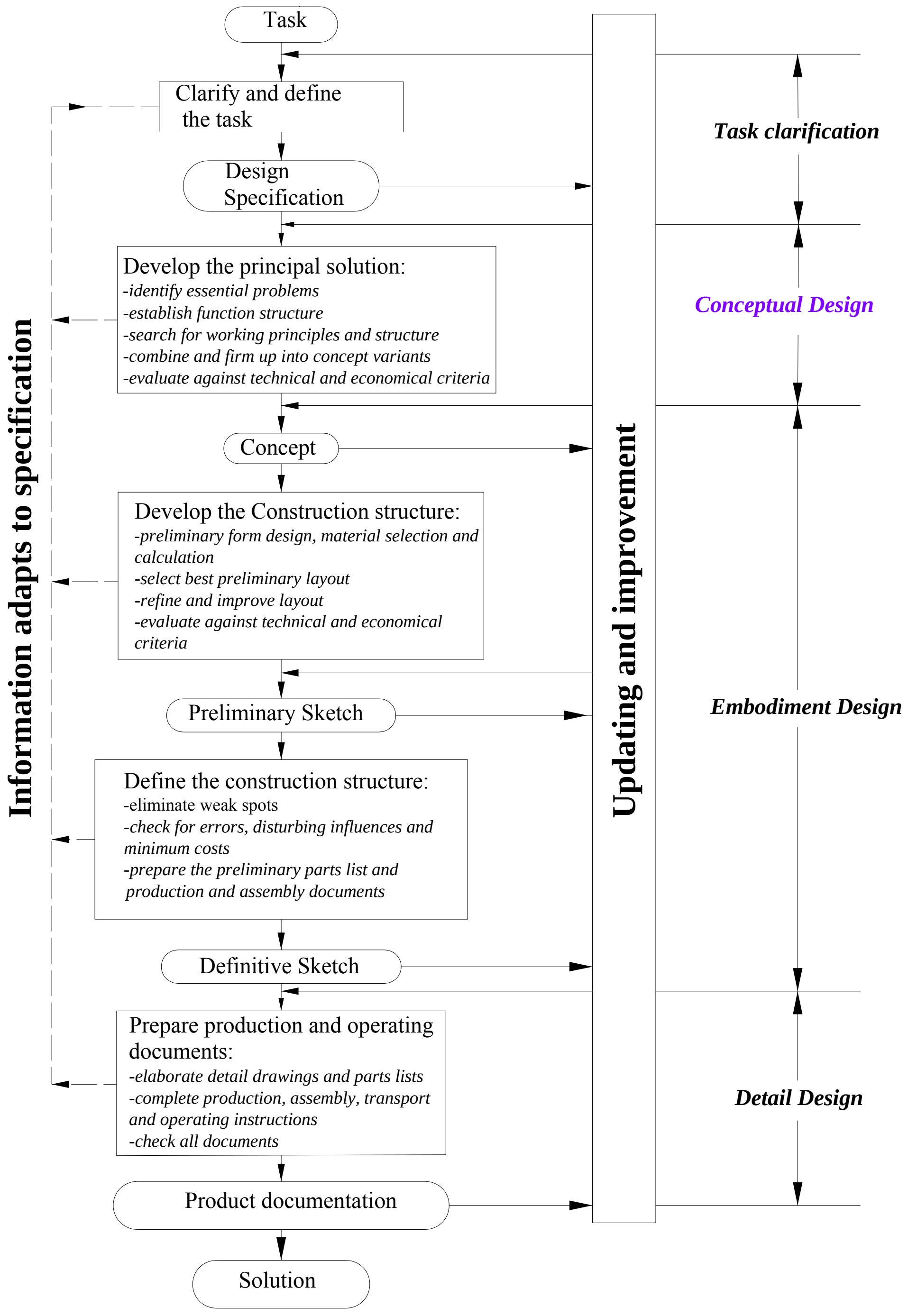

Task
Clarify and define the task
Design Specification
Develop the principal solution:
-identify essential problems
-establish function structure
-search for working principles and structure
-combine and firm up into concept variants
-evaluate against technical and economical criteria
Concept
Develop the Construction structure:
-preliminary form design, material selection and calculation
-select best preliminary layout
-refine and improve layout
-evaluate against technical and economical criteria
Preliminary Sketch
Define the construction structure:
-eliminate weak spots
-check for errors, disturbing influences and minimum costs
-prepare the preliminary parts list and production and assembly documents
Definitive Sketch
Prepare production and operating documents:
-elaborate detail drawings and parts lists
-complete production, assembly, transport and operating instructions
-check all documents
Product documentation
Solution
Information adapts to specification
Updating and improvement
Task clarification
Conceptual Design
Embodiment Design
Detail Design

Figure 1.1 Phases of design process (adapted from (Pahl and Beitz, 1996))

1.2 Problem Statement

Conceptual design is considered as the most important and critical phase of any product design process. It is the stage where the product's fundamental features are determined, 70-85% of the life-cycle costs of the product are committed (Sieger and Salmi, 1997, Zuo and Director, 2000, Hsu and Liu, 2000, Rao et al., 1999) and other major decisions are made with incomplete and imprecise information. Decisions made early at this stage have a significant impact on other aspects of the product's life cycle such as quality, cost, and manufacturability. It is usually difficult and even impossible to compensate a poorly conceived concept with good detail design process (Pahl and Beitz, 1996, Hsu and Woon, 1998, Rao et al., 1999, Hsu and Liu, 2000, Wang et al., 2002). Therefore, conceptual design requires special attention in order to get successful design.

Currently, there are several mechanical computer aided design (CAD) tools in the market to reduce the workload of human designer and product development time. However, most of these tools are used in the later phases of design such as drafting, geometric modeling, and computer aided engineering (CAE), which are mostly based on geometric information. These tools do not deal with the aspects of conceptual design process such as functional modeling, concept generation, combination and evaluation, which are function based and important during the conceptual design phase. Hence, the strength and use of the currently available CAD tools lies more at the detail design phase than the conceptual phase (Robertson and Radcliffe, 2009). Currently, there is no known commercial CAD tool that can be used for the whole conceptual design process.

In designing a product, knowledge about the product is gained as the design process progress from conceptual design to the detail design phase, but the impact of decision declines. This is because decisions made at the earlier stages become constraints for the later stage. Hence, the later stage of design is mostly done within the limits set during the conceptual phase. Figure 1.2 shows the impact of decision and the availability of computer tools during the different phases of the product design process. This indicates that there is a greater opportunity to enhance the design process during the conceptual design phase if computer support tools are employed. Furthermore, as the knowledge about the design is gained during the design process, the design requirements may change or evolve to new requirement, which has not been recognized at the beginning. This makes the design process iterative through which some of the design activities are repeated for refinement and improvement. Possible design options should be explored exhaustively and carefully to make necessary changes early in the design process. Changes made at the later stage of design are more costly and results in delay of the final product release (Qiu et al., 2002). The iterative and repetitive design tasks can be computer assisted in order to reduce product development time and improve the design process.

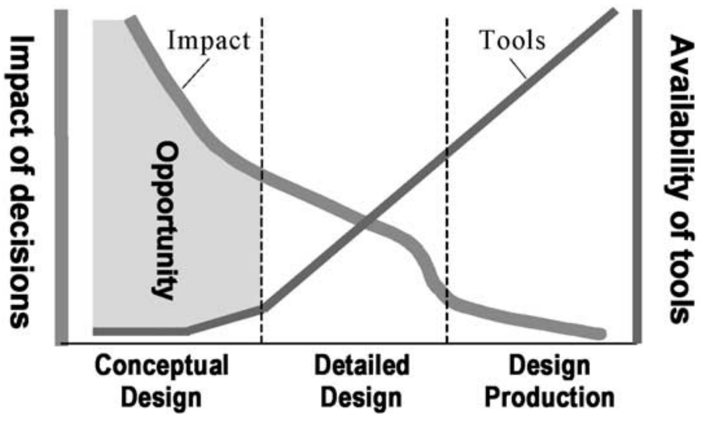


Figure 1.2 Impact of decision and availability of computer tools during the design process (Wang et al., 2002)

Conceptual design process is knowledge intensive, and requires collaboration of expertise from different disciplines as it needs large amount of diverse information. Furthermore, these large amount of data needs to be explored (i.e., processed) carefully to get better design. However, humans can only process seven plus or minus two information at a time (Miller, 1956). Because of this limitation, it is difficult to explore all the design space manually within a given time and make sound judgment. On the other hand, computers are capable of handling and processing large data though they are not creative like human being. The hypothesis of this research is that, by combining human creativity with computer capabilities it is possible to perform the conceptual design process more effectively than solely manual design.

1.3 Research Objective

The objectives of the research work reported in this thesis are:

1. To investigate the problems associated with function-based conceptual design process.
2. To propose a model or framework that can be used in functional design at the conceptual design stage to assist designers.
3. To develop a computer based design support tool to test and validate the proposed model.

The objectives of this research are realized through the development of the following research modules:

- *Functional modeling*: This includes defining functionality and representation of functions in such a way that it is both understood by the machine and human being.
- *Concept generation*: This includes representing concepts, generating concepts from database using domain independent production rules, assisting designers in conducting concept generation process manually, and displaying the generated concepts on morphology chart.
- *Concept combination*: This includes domain independent production rules to combine generated concepts to create concept variants.
- *Concept evaluation*: This includes assisting designers to define selection criteria for a given design problem and evaluating concept variants using different evaluation techniques.

In this research, a methodology integrating systematic design approach with a knowledge based system is proposed to develop the conceptual design process model as shown in Figure 1.3. The proposed model is implemented into a computer program known as conceptual design support tool (CDST) to assist designers during the conceptual design process and improve the design process. The detail methodology and development of CDST are discussed in Chapter 3 and Chapter 4 respectively.

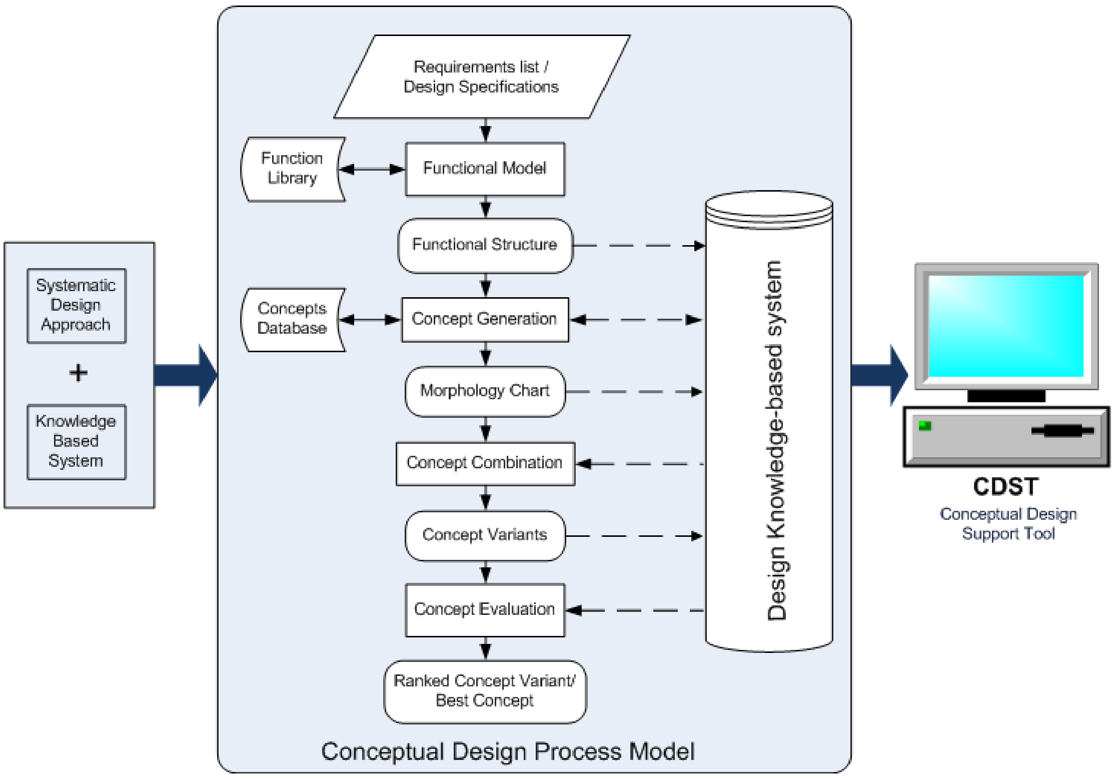

Systematic Design Approach
+
Knowledge Based System
Requirements list / Design Specifications
Function Library
Functional Model
Functional Structure
Concepts Database
Concept Generation
Morphology Chart
Concept Combination
Concept Variants
Concept Evaluation
Ranked Concept Variant/ Best Concept
Conceptual Design Process Model
Design Knowledge-based system
CDST
Conceptual Design Support Tool

Figure 1.3 Schematic view of the general methodology in developing the CDST

1.4 Scope of the Research

The objective of this research is not meant to fully computerize or automate the entire conceptual design process and substitute human designer with the computer tool. However, the aim is to integrate human creativity with computational and data handling capabilities of computers which results in hybrid conceptual design process.

The design knowledge in the computer support tool is developed based on design reuse philosophy. Design reuse plays central role in the conceptual design stage especially in concept generation process. Conceptual design knowledge can be obtained from experts or extracted from existing products and saved in the design knowledge base. Knowledge from current design process can also be used for future designs. To archive design knowledge in a computer, the use of standard method of representing mechanical functions and alternative concepts is important. This fosters reuse of the design knowledge for other similar problems in the future. The alternative concepts at this early phase of design are at higher level of abstraction with no detail geometric or material information. Hence, only the concept's functionality and input/output flows are captured.

The computer support system will be used in generating alternative concepts for given functions, creating morphology chart, combining compatible alternative concepts, and evaluate the concept variants in the mechanical engineering domain. In general, the tool will assist human designer in the conceptual design process:

- by providing design knowledge from past experiences;
- handling some of the monotonous and time consuming tasks which gives the designer more time to concentrate on the creative part of design where humans are better than computers; and
- capturing the new concepts generated during the current design process for future reuse.

The production rules in the knowledge-based system are generic to be used for any mechanical conceptual design process. However, the domain of application for the current research is concerned with conceptual design knowledge of subsea process equipment design in oil and gas industry. There is no known conceptual design support tool so far to address this domain. In recent years because of high global oil demand, depletion of old onshore fields and technological advancement, operators are moving to deepwater field development. The produced fluid from subsea wells which is mostly a multiphase mixture of oil, water and gas is transported to a platform or floating production storage and offloading (FPSO) deck located many kilometers away for processing. Because of back pressure imposed by production risers and long tie-backs there is a growing interest in processing the produced fluids on the seafloor (i.e., subsea processing) (Scott et al., 2004). Subsea processing mainly comprises of subsea separation and boosting. Separating fluids on the sea floor will avoid lifting large volumes of water to the surface for processing and disposal. Furthermore, subsea processing provides lesser susceptibility to hydrate formation since all the processing to final saleable crude can be done at the seabed. In general subsea processing provides reduced load requirement on the platform, and improved recoveries and greater efficiencies (Lyons and Plisga, 2005). However, subsea processing is an emerging technology which has not yet been fully utilized and there is also resistance from operators to use this new approach. Figure 1.4 shows a typical subsea processing consisting of separator module and multiphase pump module.

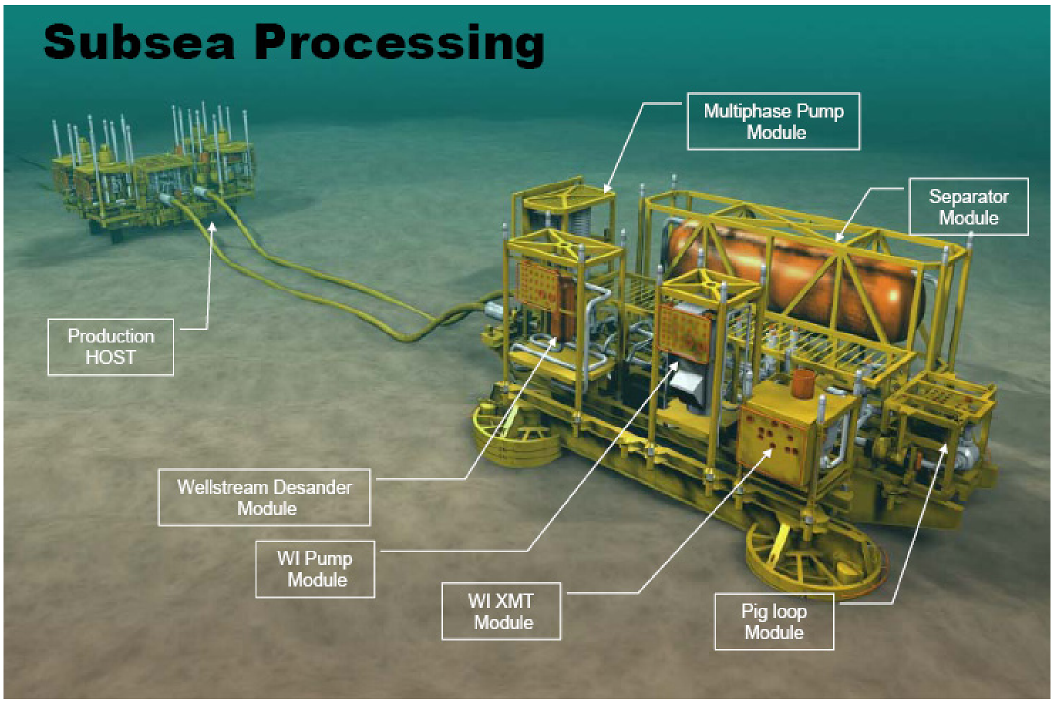


Figure 1.4 An example of subsea processing (Sapihie, 2007)

1.5 Thesis Organization

This thesis is composed of seven chapters. Chapter 2 presents the literature review of researches related to this work. This includes review of conceptual design process, support of artificial intelligence systems in design, and a survey of function-based design and computer aided conceptual design tools so far developed as bench mark. Chapter 3 presents the research methodology used to achieve the objective of this research. The proposed conceptual design model and the integrated knowledge-based system together with the programming environments used to implement the model are discussed. Chapter 4 presents the development of the conceptual design support tool (CDST) in detail. The applicability and implementation of CDST is illustrated with case studies in Chapter 5. CDST is demonstrated first with conceptual design of three phase separator and then with a general conceptual design support tool for subsea process equipment (CDSTsped). Chapter 6 describes about the verification and validation tests conducted by reviewers after using CDST. Finally Chapter 7 concludes this thesis by discussing the contributions and limitation of the work presented together with recommendation for future research.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter reviews the state of the art in the conceptual design research and areas in the computer support tool development for conceptual design. In particular, different conceptual design process models and the integration of artificial intelligence with design in developing computer aided conceptual design tools are reviewed. Function-based design approach together with the definition, classifications, representations, and decomposition of functions is presented. Finally, a review of existing computer aided conceptual design tools that support designers in performing conceptual design is presented.

2.2 Introduction to Conceptual Design Process

Conceptual design has been defined as that phase of design process where the designer identifies the essential problem through abstraction, establish functional structure, search for suitable working principles (concepts) and combine these into a working structure (concept variants), and evaluate the concept variants against technical and economical criteria to come up with one or two concept variants for further development (Pahl and Beitz, 1996). It is the phase where most important decisions are made by combining the knowledge from engineering science, practical experience, production methods, and marketing (French, 1998). The goal of conceptual design phase is to explore the design problem and field of solutions, together with finding the best solution that is feasible for further development (Bonnema and Houten, 2004).There is greatest demand from the designer to make remarkable improvements on the design at this stage. Hence, conceptual design is a complex process where the designer needs to make wise decisions by considering several parameters.

In order to support the conceptual design activity with computers, the product to be designed and the knowledge from which the design is developed needs to be modeled correctly. But this modeling process is considered as one of the most difficult issues to address (Hsu and Woon, 1998). Feature-based, knowledge-based and function-based design models are some of the modeling approaches used during the conceptual design process. These modeling approaches in relation to conceptual design support system will be reviewed next.

Feature-based design

When a product model is built using design features, it is known as design by features or feature-based design approach. There are varieties of definitions for the term "feature" in design literature indicating no consensus among the researchers (Hashim et al., 1994, Allada, 2001). Considering its purpose Hashim (Hashim et al., 1994) defined feature as a geometric entity having geometric attributes (e.g., holes, protrusions, bosses etc) that either provides or accepts a function. Features can be used to convey information used to model the relationships between the requirements, functional description, and physical solutions of a product (Brunettia and Golob, 2000). Computer tools developed based on feature-based design approach allows the designer to use mechanical features stored in feature library to build the product (Kamrani and Vijayan, 2006). Furthermore, this design approach helps the designer to consider the manufacturability and assembly process early in the conceptual stage. Feature based design approach suffers from the following limitations (Allada, 2001):

1. Feature validation needs to be performed every time a new feature is added to ensure the new feature is placed in correct position or it does not affects the existing features.

2. The determination of what features must be present in the feature library. A feature library with limited number of feature primitives may be difficult to satisfy various design needs, i.e., represent various design problems (Hsu and Woon, 1998). On the other hand a feature library with too many predefined features becomes cumbersome for the designer.

Knowledge-based design

Conceptual design is a knowledge intensive process where diversified knowledge and several years of experience are required to design quality, cost effective, and innovative product. Knowledge of experienced designers should be acquired and kept for future reuse or to train novice designers before the experienced designers retire or leave the company. Knowledge-based design is a computer based design approach which relies on knowledge acquired from experienced designers, analyzing existing products, handbooks, patents etc. to automatically perform design process or to support designers. The acquired knowledge is represented in the form of facts and production rules in the knowledge-based system. A knowledge-based system is an artificial intelligence (AI) system, consisting of domain knowledge in the knowledge base, a controlling mechanism (an inference engine), and interface to the outside world through user interface. Knowledge-based system use symbolic representation of knowledge which can easily be understood both by human designer and the machine. Furthermore, since the domain knowledge is separated from the controlling mechanism, it is easy to add new knowledge during the program development or later (Hopgood, 2001). A number of researchers used knowledge-based design approach for conceptual design of products to generate design solutions from existing design knowledge (Tong and Gomory, 1993, Bracewell and Sharpe, 1996, Sieger and Salmi, 1997, Moulianitis et al., 1999, Zhang et al., 2001b).

Function-based design

The conceptual design stage, which starts with requirements list and results in concept variants satisfying those requirements, is function oriented, and the process is known as functional design. Every product has reason behind its existence which is its function. The main design focus at the conceptual design stage is to find design solutions that can achieve the required functions, hence conceptual design is considered as function driven and the process functional design. Tor et al (Tor et al., 1998) defined functional design as an approach for designing CAD software that incorporates the representation of functional information, as well as structural information, and its aim is to provide computer tools to link design functions with the physical embodiments used to realize the functions.

Representing functionality in a computer program in human and machine understandable form, and functional reasoning (i.e., the use and manipulation of functional knowledge in a form suitable for computer based environment) are the basis for computer support tools using function-based design approach. A number of researchers advocate function-based design approach (Suh, 1990, Pahl and Beitz, 1996, Dieter, 2000, Ullman, 2003). In function-based design, the customer requirements are transformed into sets of functions (i.e., functional modeling) in a solution neutral form, which helps the designer not to stick to specific solutions too early in the design process. Functional models of products/devices provide a high-level representational framework in which activities such as design, diagnosis, verification, and modification can be performed without reference to the actual structure of the system (Erdena et al., 2008). Using the functional model, the designer generates wide-ranging alternative solutions and selects the most promising ones for further development. The function-based design approach will be discussed in detail in Sections 2.5 and 2.6.

In this thesis, a hybrid approach consisting of function-based and knowledge-based design approaches is used to develop the computer aided conceptual design tool which will be discussed further in Chapter 3.

2.3 AI in Design

Artificial intelligence (AI) has been defined as the simulation of human intelligence on a machine, so as to make the machine efficient to identify and use the right piece of knowledge at a given step of solving a problem (Konar, 2000). For the machine, to think intelligently, domain knowledge should be represented and stored together with means to reason about the knowledge. Within AI, three main directions of reasoning can be distinguished (Rentema and Jansen, 2000):

- Reasoning by logic, e.g. Rule-based reasoning technique in expert systems where the domain knowledge can be formalized into simple rules.

- Reasoning by learning, e.g. Artificial Neural Networks (ANN). An ANN consists of a network of nodes (processing elements) connected via adjustable weights (connections). By training the network with a large set of input-output pairs, the system learns the functional relation between the input and the output space. ANN is good for classification tasks and for performing associative memory retrieval. Hence, many neural networks applications in engineering design are geared towards either classifying the designs into families of design problems, or to finding the nearest values for the design parameters (Hsu and Woon, 1998).

- Reasoning by analogy, e.g. case-based reasoning technique. Case-based reasoning is the general problem solving method where a given problem is solved by retrieving and adapting stored solution to a closely related problem (Goel and Chandrasekaran, 1990). When a new problem is presented, the system searches for cases with similar problem descriptions. Although the retrieved case usually does not completely fit the new problem, the retrieved solution may be a good starting point for further adaptation.

AI systems are suitable to solve non-deterministic and "ill-structured" problems. Design problems are widely recognized as being "ill-defined" or "ill-structured", as opposed to well-defined or well-structured problems which have clear goal, and often one clear answer (Cross, 2008). The characteristics of "ill-defined" or "ill-structured" problems are:

- There is no definitive formulation of the problem. When the problem is initially set, the goals are usually vague, and many constraints and criteria are unknown.

- Any problem formulation may embody inconsistencies. Mostly, many conflicts and inconsistencies emerge only in the process of problem solving, and these have to be resolved in the solution.

- Formulations of the problem are solution-dependent. It is difficult to formulate a problem statement without implicitly or explicitly referring to a solution concept.

- Proposing solutions is a means of understanding the problem.
- There is no definitive solution to the problem. The mapping between the problem and solution is not usually one-to-one, which makes designing non-deterministic process. Different solutions can be equally valid responses to the initial problem.

These non-deterministic and “ill-structured” natures of design make it suitable to be solved with AI systems. Integrating AI with design begins in the early 1980s. The goal of using AI systems in design at that time was to develop intelligent CAD system that could design products more or less automatically with minimum user inputs and interactions (Tomiyama, 2007). Design was considered in its broader sense including analysis, selection (of components or materials), parametric design, optimization, data integrity management (such as geometric constraint management), process planning, and synthesis. However, this objective has not been achieved. Because of this, in the past two decades, the research in this area focus on developing an integrated design support environment that can provide useful knowledge and guide the designer rather than automating the design process. Tomiyama (Tomiyama, 2007) pointed out two major concepts as requirements for intelligent CAD development:

- The intensive use of design knowledge to design artifacts in one way or another, and
- The intelligent CAD should exhibit knowledge management capabilities because design is mostly a knowledge generation process.

A number of researches have been done in integrating AI with design especially during the conceptual design stage. For example, EFDEX (Engineering Functional Design Expert) is a knowledge-based (rule-based) system for automating conceptual design for specific domain (Zhang et al., 2001b), and AIDA (Artificial Intelligence supported Design of Aircraft), integrates rule-based and case-based techniques for supporting designers in the conceptual design of aircraft (Rentema and Jansen, 2000). The general descriptions of these systems and other computer aided conceptual design tools will be discussed further in Section 2.8.

In this research from AI system, a knowledge-based system is used to achieve the objectives of the research. Next, the components and features of a knowledge based system are reviewed.

2.4 Knowledge-Based System

Knowledge-based system is an artificial intelligence system which uses stored knowledge to solve problems in a specific domain. The three essential components of a knowledge-based system are:

i. The knowledge base;

ii. The inference engine; and

iii. Interface to the outside world.

The separation of the domain knowledge (knowledge base) from the controlling mechanism (inference engine) makes knowledge-based systems different from conventional programs where domain knowledge is intimately intertwined with software for controlling the application of that knowledge (Hopgood, 2001). This separation makes it possible to represent knowledge in a more natural way in which humans describe their own problem solving techniques. Furthermore, the explicit separation of knowledge from control makes it easier to add new knowledge, both during the program development and in the program's life time by incorporating a knowledge acquisition module to the knowledge-based system. The architectural components of a typical knowledge based system with knowledge acquisition module are shown in Figure 2.1. In the following sections these components of the knowledge-based system are discussed further.

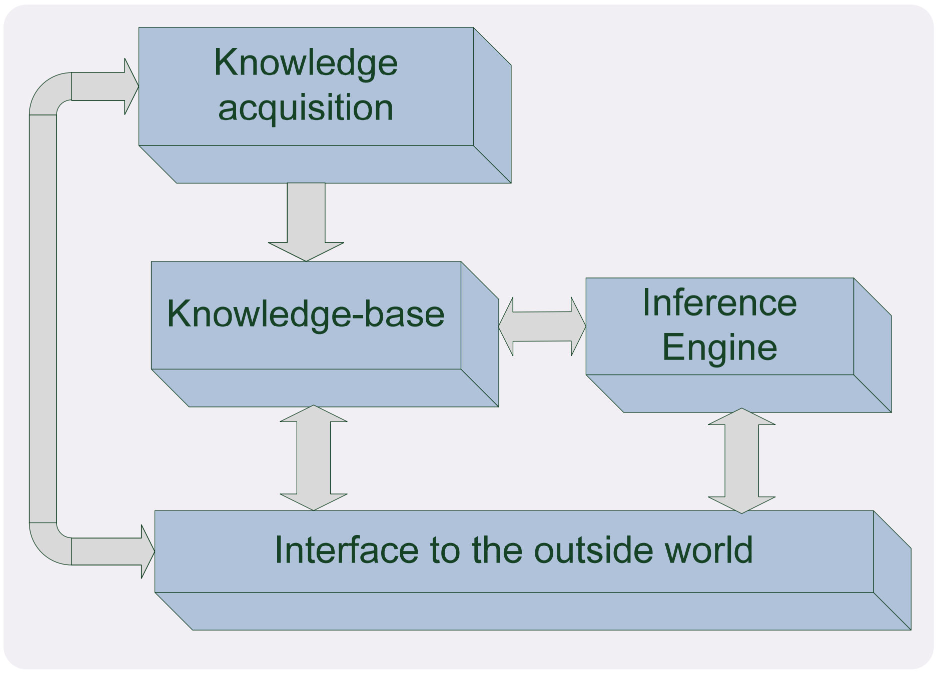


Figure 2.1 Architectural components of knowledge-based system

2.4.1 The knowledge base

The knowledge base contains the domain specific and control knowledge which is used to solve problems in the domain. This knowledge can be obtained from experts or published literatures such as handbooks, manuals, etc. The acquired knowledge should be represented following appropriate knowledge representation formalism and encoded so that it is amenable to computer manipulation. Knowledge can be represented and stored in the knowledge base in various forms. The main knowledge representation formalisms proposed in the literature includes (Nikolopoulos, 1997):

- *Rules (Production rules)* represent knowledge in the form of :

  If <condition> *then* <conclusion or action>
- *Semantic network* represents knowledge as a labeled directed graph with nodes (oval shaped) corresponding to objects, situations or concepts and arcs corresponding to relations or association between the nodes. The term semantic networks encompass a family of graph-based representations. These includes:
  - *Conceptual graph* represents knowledge using connected bipartite graph whose nodes represent either concepts (represented as box), or

conceptual relations (represented as ellipse). Conceptual graph does not use labeled arcs unlike semantic networks; instead the conceptual relation nodes represent relations between concepts (Luger and Stubblefield, 1998).

    - *Petri nets* represent knowledge using a directed bipartite graph having two types of nodes known by places (i.e., conditions) and transitions (i.e., discrete events that may occur), and directed arcs to connect nodes of different types describing which places are pre- and/or post conditions for which transition. A place is considered as an input to a transition if and only if there is a directed arc from the place to the transition and as an output to a transition if and only if there is an arc from the transition to the place. The place nodes are represented by circles and the transition by bars or boxes.

- *Frames* provide a means for organizing and representing knowledge as structured objects consisting of named slots with attached values. Frames can be connected through class-subclass relationships to form a frame system allowing data abstraction and inheritance i.e., a frame can inherit properties from its parent.
- *Object oriented paradigm* (OOP) provides a means to represent knowledge in a structured manner including data abstraction, inheritance, encapsulation (or information hiding), and dynamic binding (or late binding) (Hopgood, 2001). Because of this, knowledge representation with this scheme requires a programming language which supports these capabilities.
- *Hybrid representation* combines multiple representation paradigms into a single integrated programming environment. The fact that different sections of knowledge base may be encoded more efficiently using different formalisms, reveals the importance of hybrid systems.

The selection of knowledge representation scheme among these varieties of approaches depends on the knowledge to be represented and the capability of programming environment used.

2.4.2 The inference engine

The inference engine is the controlling mechanism which contains general algorithms, which are able to manipulate the knowledge stored in the knowledge base. In knowledge-based systems the inference mechanism compares the data/facts in the fact base with the information in the knowledge base and decides which information in the knowledge base applies to the data/fact to deduce results in an organized manner. The inference engine works based on an inference rule and a search strategy. There are two types of inference engines: forward-chaining or data-driven and backward-chaining or goal-driven.

2.4.2.1 *Forward Chaining*

In forward chaining or data-driven method, rules are selected and applied in response to the current fact base. The fact base comprises all facts known by the system, which are either derived by rules or supplied directly. In this method, the information in the fact base is compared with the IF part of the rules in the knowledge base. If a rule is found whose IF part matches the information in the fact base, then the rule fires, i.e., the rule's THEN part is added to the fact base. The procedure repeats until all possible conclusions are drawn.

2.4.2.2 *Backward-chaining*

Backward-chaining is an inference strategy that assumes the existence of a goal that needs to be established or disproved. In backward-chaining, the system forms a hypothesis that corresponds to the THEN part of a rule or set of rules in the knowledge base and then attempts to justify it by searching the fact base to establish the facts appearing in the IF part of the rule or rules. If successful, the hypothesis is established and the system reports its results; otherwise, another hypothesis is formed and the inference mechanism repeats the procedure.

The selection of the inference mechanism used for a given problem depends on the knowledge representation chosen, since each knowledge representation scheme has its own associated inference mechanism, i.e., different knowledge representation techniques support different types of inference processes. In addition to this, the programming environment chosen also affects the inferencing mechanism (for example Prolog uses backward-chaining inferencing while CLIPS uses forward-chaining mechanism).

2.5 Function-Based Design: A Survey

In this section, function-based design approach is reviewed from different perspectives in the literature. Various approaches used to classify and represent functional knowledge with their merits and demerits are presented together with the approaches used in the current research.

2.5.1 Definition of Function

Even though function is a critical aspect of design, especially during the conceptual design stage, there is no clear, uniform, objective, and widely accepted definition of function (Umeda and Tomiyama, 1997). Function represents the designer's intent about the expected product's basic characteristics. Some of the definitions available in the literature are presented next.

From design problem solving point of view, Pahl and Beitz defined function as the general input/output relationship of a system whose purpose is to perform a task and it should be represented independent of any particular solution (Pahl and Beitz, 1996). Function has also been defined from performance point of view by Cole (Cole, 1998) as the actions a system must perform in response to its environment in order to achieve the mission or goals given to it. Considering the way the design problems and their solutions should be described, Chakrabarti and Bligh (Chakrabarti and Bligh, 2001) defined function as a description of the action or effect required by a design problem, or that supplied by a solution. With regards to designer's intention in defining/describing a design problem, function has been defined as purpose or intended use (Hashim et al., 1994). From design goal (i.e., device/artifact) point of view, Stone and Wood (Stone and Wood, 2000) defined function as a description of an operation to be performed by a device or artifact. According to Sturges et al (Sturges et al., 1993, Sturges et al., 1996), function is defined as the domain-independent characteristics or behavior of elements or groups of elements. Umeda et al (Umeda et al., 1996) argue that it is difficult to distinguish function clearly from human behavior from which it is abstracted and defined function as a description of behavior abstracted by the human through recognition of the behavior in order to utilize the behavior.

Apart from variations in defining function, all the researchers agree that function plays central role in product design and development process especially at the conceptual design stage. They defined function from different perspectives. It is believed that these definitions are acceptable provided that they are capable to express the designer's intent and describe the effect provided by the product or device unambiguously.

2.5.2 Functional Representation

Conceptual design process can be considered as transformation of design specification which is given as requirements list (i.e., functional requirements) into one or more concepts that can satisfy these requirements for further development. In order to develop computational methods to support this synthesis process, formal conceptual design process model is required, where design problem and solutions can be described and represented in terms of their functions. A formal functional representation technique is important for functional modeling and functional reasoning among others. The representation scheme should support easy definition, modification, and retrieval of functions by the designer for specific design problem.

Traditionally there have been three approaches to represent functions in design (Chakrabarti and Blessing, 1996, Chiang et al., 2001). These are:

i. Representing function in the form of verb-noun pairs – an example would be the function of a shaft, i.e., "transmit torque";

ii. Input-output flow transformations, where the inputs and outputs can be energy, materials, or information, i.e., flow-based representation; and

iii. Transformation between input-output situations and states, i.e., state-based representation

However, the last two approaches can be grouped together and functional representation can be generalized in two approaches as proposed by Chakrabarti and Bligh (Chakrabarti and Bligh, 2001):

i. Linguistic (Natural-language-like) representation of function, i.e., verb-noun pairs. This kind or representation is close to the way humans express their ideas, however it is difficult to formalize in a generalized way for computer application.

ii. Mathematical representation of function, where function is expressed as a transformation between input and output. Input/output transformations can be represented by mathematical functions in situations where a function involves the process of flow of energy, material or signal, or physical quantities. This representation can be easily formalized for computer application but it needs translation to designer's natural language.

From these two approaches, it is preferable to represent function qualitatively, using a linguistic description since mathematical representation of design in the early stage is not always feasible with limited information available. Thus, in this research, a hybrid functional representation approach is used consisting of linguistic approach by verb-noun pair together with the flows of energy, material and signals where applicable. The relationship between inputs and outputs is expressed independently of the solutions. This hybrid representation minimizes the limitation of flow based representation and gives more flexibility to the designer. For insistence, flow based representation can not sufficiently describe a function which is not transformation between input and output (e.g. function of a bolt). In such cases function is represented by verb + noun pair (connect/fix solid material for bolt). Similarly, if the function consists of transformations of flows as in the case of motor for example, the function is represented with verb + noun together with flow description (e.g., convert electrical energy to mechanical energy for motor)

2.5.3 Functional Classifications

Classification (taxonomy) of mechanical functions has been one of the research areas in the design community to develop common classification scheme. Functional classification of mechanical functions is required to:

i. Provide standardized common design language to represent a product and eliminate semantic confusion;

ii. Assist in developing computational tools for function-based design approach; and

iii. Assist designers in developing functional modeling process indicating clear stopping point for functional decomposition in a repeatable manner.

The work done by Collins et al (Collins et al., 1976) can be considered as the first attempt to list mechanical functions. After studying the failure mode and occurrence of helicopter system, they described each part of a helicopter in terms of its function(s). Based on this, they propose classification consisting of 46 keywords and 40 antecedent adjectives from which they identified 105 elemental mechanical functions. According to their work elemental mechanical function is defined as a distinctive generic characterization of the basic function of a machine part without reference to the specific application for which it is used. Their classification has the following limitations:

i. Despite the fact that helicopter is a complex machine, their classification may not be exhaustive enough to cover all mechanical functions.

ii. The elemental mechanical functions are not grouped or organized logically, thus it may be considered as collection of mechanical functions rather than classification, and

iii. There are several functions which seem to be repeated and can be grouped together, for example switching and gas switching, signal transmitting and information transmitting, pumping and pumping oil are considered to be different functions.

In the early eighties, Pahl and Beitz (Pahl and Beitz, 1996) proposed five generally valid functions namely: change, vary, connect, channel, and store which are derived respectively from type, magnitude, number, place and time characteristics. They also defined three types of flows: flow of matter, energy, and signal. However the generally valid functions are at higher level of abstraction, which may sometimes hinder the direct search for solutions.

Extending the functional classifications of Pahl and Beitz, Hundal (Hundal, 1990) classified primary categories of basic functional classes into six as channel, store/supply, connect, branch, change magnitude, and convert. He further classified each of these according to the quantities handled (material, energy, signal), the input and/or output, their physical forms and other necessary descriptors and proposed thirty nine sub-categories of these basic functions shown in Table 2.1.

Table 2.1 Hundal's primary categories and sub-categories of basic functions (Hundal, 1990)

Primary categories of basic functions	**Sub-categories**
channel	transmit, transport, move, stop
store/supply	store, supply
connect	connect, compare, mark, valve, switch, pack, mix, add, subtract, multiply, divide, AND, OR
branch	cut, branch, count, display, separate
change magnitude	process, crush, form, coalesce, change
convert	liquefy, solidify, evaporate, condense, integrate, differentiate, NOT, display, sense, convert

Kirschman and Fadel (Kirschman and Fadel, 1998) proposed taxonomy of elemental mechanical functions after analyzing Collin’s work (Collins et al., 1976) and considering consumer products. Accordingly they proposed four basic mechanical function groups which are related to the concepts of Motion, Power / Matter, Control, and Enclosure. This classification system was extended to cover more descriptive mechanical functions, as shown in Table 2.2. To increase the information content of the function which is normally established by combining verb-adjective, they include directions and convert to sentence form that leads to about 150 combinations of elemental mechanical functions. Though their taxonomy is more structured and includes functions of consumer products in addition to Collin's work, it does not attempt to cover all functions used in mechanical design. Furthermore, there functional representation which is formed by verb-adjective varies from the commonly used verb-noun representation adapted from value engineering in the early sixties.

Table 2.2 Basic function groups and their extension (Kirschman and Fadel, 1998)

motion	• rotary, linear, oscillatory, other • create, convert, modify, dissipate, transmit • flexible, rigid
control	• power, motion, information • continuous, discreet • modification, indication • user-supplied, internal feedback
power/matter	• store, intake, expel, modify, transmit, dissipate • electrical, mechanical, other
enclose	• cover, view, protect • removal, permanent • support, attach, connect, guide, limit

Deng et al (Deng et al., 1998) argue that it is not possible to classify all mechanical functions because of the diversity in mechanical components. They defined fundamental mechanical functions as the lowest level embodiment functions. Fundamental mechanical functions are functions which are associated with fundamental physical structures. Accordingly, they classified fundamental mechanical functions into four categories as:

1. Functions relating to supplying or storing energy or material, e.g. the functions of electric motor, spring, flying wheel, oil tank, etc.
2. Functions relating to transmitting energy or material. This category can be further classified as:
   - Transmitting motion, e.g. the functions of shaft, gear, belt, chain;
   - Transmitting force or moment
   - Transmitting material, e.g. the function of pipe.
3. Functions relating to converging or branching energy or material, e.g. the functions of switch, valve, gear train, etc.
4. Functions relating to changing form or magnitude of energy or material, or physical quantities relating to energy. This category can be further classified as:
   - Changing form of energy, or changing form of physical quantities relating to energy, or changing form of material,
   - Changing magnitude of physical quantities relating to energy, or flow of material.

These categories neither lay ground for common vocabulary to perform functional modeling in a repeatable manner nor clearly define stopping point for functional decomposition in creating functional structure.

Stone and Wood (Stone and Wood, 2000), proposed a common design language termed as “functional basis”, which allows designers to describe a product’s overall function as a set of simpler sub-functions. They defined functional basis as a standard set of functions and flows capable of describing the mechanical design space. With this definition, they proposed a group of eight classes of mechanical functions: branch, channel, connect, control magnitude, convert, provision, signal, and support. These classes are extended to include twenty four basic functions and eight flow restricted functions. Functional basis also defines three classes of flows: material, signal and energy; with nineteen basic and eleven sub-basic flows. In functional basis, functions (both overall and sub-) must be expressed as a verb-object pair where the basis functions fill the verb spot and the basis flows provide the object. They claim that functional basis subsumes previous taxonomies and offers a more complete and consistent set of functions and flows that is non-redundant, for electromechanical domain.

Szykman et al (Szykman et al., 1999) from the National Institute of Standards and Technology (NIST), United States, developed a taxonomy of about 120 functions and over 100 flows, by extracting and distilling from extensive review of literature related to function and flows terminologies. There are several similarities between their taxonomy and the functional basis of Stone and Wood. Because of this, researchers from NIST taxonomy and functional basis reconcile the two functional vocabularies following a three step algorithm consisting of review, union and reconcile steps, and come up with reconciled functional basis (Hirtz et al., 2002). They claim that the reconciled functional basis completely describe the electromechanical design space. In the reconciled functional basis, functions are classified into eight classes (primary): branch, channel, connect, control magnitude, convert, provision, signal and support; which further classified into forty five secondary and tertiary classes of action verbs as shown in Table 2.3. Similarly, the reconciled flow set consists of three basic classes (primary) flows: material, energy and signal, which also have forty two secondary and tertiary flows as shown in Table 2.4. Note that in both tables, the column labeled as “correspondents” is provided as aid for mapping from terms that are not in the reconciled functional basis to the terms that are.

After thoroughly studying those functional classifications and taxonomies, the requirement of standard functional representation which should be exhaustive enough to cover most of mechanical design domain, and accepted by the design community remains central to be addressed. There are two options to tackle this problem: to adopt one of those taxonomies with some modification or to come up with new taxonomy of mechanical functions. The latter option seems to be reinventing the wheel as several classifications have been done in the past, and needs time to make it universal language. Thus, among those classifications discussed in this section, the reconciled functional basis proposed by Hirtz et al (Hirtz et al., 2002), is adopted in this research. The rationale behind this selection is that, this classification subsumes most of the previous classifications and includes most of the action verbs for mechanical design.

Table 2.3 Functional basis reconciled function set (adapted from (Hirtz et al., 2002))

Class(Primary)	**Secondary**	**Tertiary**	**Correspondent**
Branch	Separate		Isolate, sever, disjoin
		Divide	Detach, isolate, release, sort, split, disconnect, subtract
		Extract	Refine, filter, purify, percolate, strain, clear
		Remove	Cut, drill, lathe, polish, sand
	Distribute		Diffuse, dispel, disperse, dissipate, diverge, scatter
Channel	Import		Form entrance, allow, input, capture
	Export		Dispose, eject, emit, empty, remove, destroy, eliminate
	Transfer		Carry, deliver
		Transport	Advance, lift, move
		Transmit	Conduct, convey
	Guide		Direct, shift, steer, straighten, switch
		Translate	Move, relocate
		Rotate	Spin, turn
		Allow DOF	Constrain, unfasten, unlock
Connect	Couple		Associate, connect
		Join	Assemble, fasten
		Link	Attach
	Mix		Add, blend, coalesce, combine, pack
Control Magnitude	Actuate		Enable, initiate, start, turn-on
	Regulate		Control, equalize, limit, maintain
		Increase	Allow, open
		Decrease	Close, delay, interrupt
	Change		Adjust, modulate, clear, demodulate, invert, normalize, rectify, reset
			scale, vary, modify
		Increment	Amplify, enhance, magnify, multiply
		Decrement	Attenuate, dampen, reduce
		Shape	Compact, compress, crush, pierce, deform, form
		Condition	Prepare, adapt, treat
	Stop		End, halt, pause, interrupt, restrain
		Prevent	Disable, turn-off
		Inhibit	Shield, insulate, protect, resist
Convert	Convert		Condense, create, decode, differentiate, digitize, encode, evaporate, generate, integrate, liquefy, process, solidify, transform
Provision	Store		Accumulate
		Contain	Capture, enclose
		Collect	Absorb, consume, fill, reserve
	Supply		Provide, replenish, retrieve
Signal	Sense		Feel, determine
		Detect	Discern, perceive, recognize
		Measure	Identify, locate
	Indicate		Announce, show, denote, record, register
		Track	Mark, time
		Display	Emit, expose, select
	Process		Compare, calculate, check
Support	Stabilize		Steady
	Secure		Constrain, hold, place, fix
	Position		Align, locate, orient

Table 2.4 Functional basis reconciled flow set (adapted from (Hirtz et al., 2002))

Class (Primary)	**Secondary**	**Tertiary**	**Correspondents**
Material	Human		Hand, foot, head
	Gas		Homogeneous
	Liquid		Incompressible, compressible, homogeneous
	Solid	Object	Rigid-body, elastic-body, widget
		Particulate	
		Composite	
	Plasma		
	Mixture	Gas–gas	
		Liquid–liquid	Aggregate
		Solid–solid	
		Solid–liquid	
		Liquid–gas	
		Solid–gas	
		Solid–liquid–gas	
		Colloidal	Aerosol
Signal	Status	Auditory	Tone, word
		Olfactory	
		Tactile	Temperature, pressure, roughness
		Taste	
		Visual	Position, displacement
	Control	Analog	Oscillatory
		Discrete	Binary
Energy	Human		
	Acoustic		
	Biological		
	Chemical		
	Electrical		
	Electromagnetic	Optical	
		Solar	
	Hydraulic		
	Hydraulic		
	Magnetic		
	Mechanical	Rotational	
		Translational	
	Pneumatic		
	Radioactive/Nuclear		
	Thermal		

2.6 Function-to-Form Mapping

In this section, the process of mapping functions to structures and different approaches of functional decomposition principles are discussed. The task of the designer during the conceptual design process is to find suitable concept that can satisfy the given requirements and constraints. The requirements at this phase of design describe the overall function of the product. This high level abstract formulation of design should be decomposed into less complex subfunctions, before the form (structure) that can perform the function is sought. This increases the innovative capability of the designer by reducing the cognitive effort required in finding solutions. There are different approaches by different researchers for functional decomposition. Some of these approaches are discussed next.

In their systematic design approach, Pahl and Beitz (Pahl and Beitz, 1996) proposed functional decomposition using the technique based on the flow of energy, material, and signal. The overall function of a given complex design problem is first defined in terms of the inputs and outputs of all the quantities involved. This overall function is then broken down into identifiable subfunctions which follow the flow of energy, material and signal as shown in Figure 2.2. The decomposition process continues until solution to each subfunctions can easily be found resulting in functional structure for the given design problem. Finally the alternative design solutions for each subfunction are generated and mapped to each subfunction.

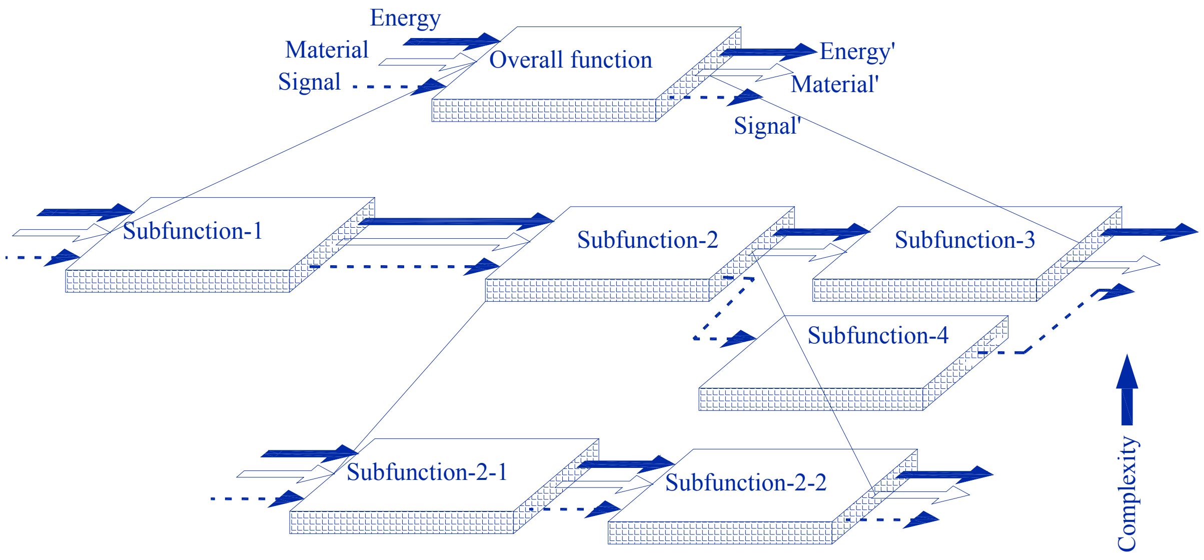


Figure 2.2 Decomposing overall function into subfunctions (adapted from (Pahl and Beitz, 1996))

The process of analyzing the requirements list and decomposing the overall function into subfunctions and creating functional structure is known as functional modeling. For flow based functions Kurfman et al (Kurfman et al., 2003) proposed five steps to derive functional modeling. These steps are:

i. Identify the input and output flows (material, energy and signal) that address customer needs,

ii. Generate a black box model of the product to be designed,

iii. Create function chains for each input flow,

iv. Aggregate function chains into a functional model, and

v. Verify the functional model with customer needs i.e., check that the collective effect of all the subfunctions in the functional model can satisfy the customer's needs.

The overall function is satisfied by combining the generated solution for each subfunction in bottom-up approach. The main problem with this type of functional decomposition is at what point should the decomposition stop and start mapping. As stated in Section 2.5.3, the decompositions should stop when each subfunction is expressed in terms of elemental mechanical functions provided that the overall function can be achieved by combining the subfunctions in the functional structure.

For each function in the functional structure the designer generate alternative concepts satisfying those subfunctions. There are a number of manual concept generation methods that improve creativity. Among these, the most recommended ones are:

i. Conventional concept generation methods such as brainstorming, 6-3-5 method, Delphi method, and Gallery method (Pahl and Beitz, 1996, Ullman, 2003).

ii. Logical concept generation method such as TRIZ (McMunigal et al., 2006).

iii. Design by analogy. For example, using biomimetic design which uses biological phenomena to inspire solutions to engineering problems (Mak and Shu, 2008, Chakrabarti et al., 2005).

iv. Combination of these methods.

Suh (Suh, 1990) proposed hierarchical decomposition in his axiomatic design approach. According to his approach, the first step in the design process is to establish hierarchical functional requirements (FRs) from the needs that the final product must satisfy. These hierarchical functional requirements in functional domain are directly mapped to design parameters (DPs) in physical domain, which are also hierarchical. However, FRs at the ith level cannot be decomposed into the next level of the FRs hierarchy without first going over to the physical domain and developing a solution that satisfies the ith level FRs with all the corresponding DPs (Figure 2.3). This zigzag process continues until the FRs can no longer be decomposed. Though, this can be considered as stopping point for functional decomposition, Suh did not propose vocabulary of standard functional requirements.

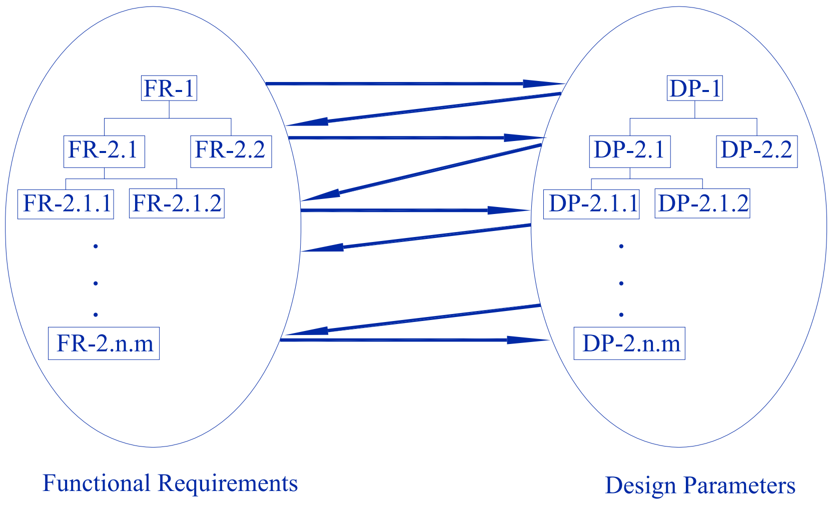


Figure 2.3 Functional decomposition and mapping in axiomatic design approach

Functional block diagram (FBD) or functional logic diagram, which is drawn using the rules of functional logic developed by Charles Bytheway in the early sixties, can also be used for functional decomposition and function to form mapping (Sturges et al., 1996). In FBD, the overall design problem is identified and represented at higher level of abstraction by basic function which is decomposed by the design team into several functions. These secondary functions are then translated into components or recursively decomposed. The function decomposition process continues until one can map each function into a component or system that will accomplish it. The general form of the FBD, shown in Figure 2.4, represent the function block (or node) consisting of the function name (what is done) in verb + noun format. The nodes to the left of a given function node represent the reason why a function is included with a higher level function. The nodes to the right are functions describing how the function is performed with lower level functions. Each higher level function is connected to the lower level function preserving this how/why relationship. They did not describe any standardized vocabulary of functions used to create FBD, and it is practically rare to find a one-to-one correspondence between functions and components.

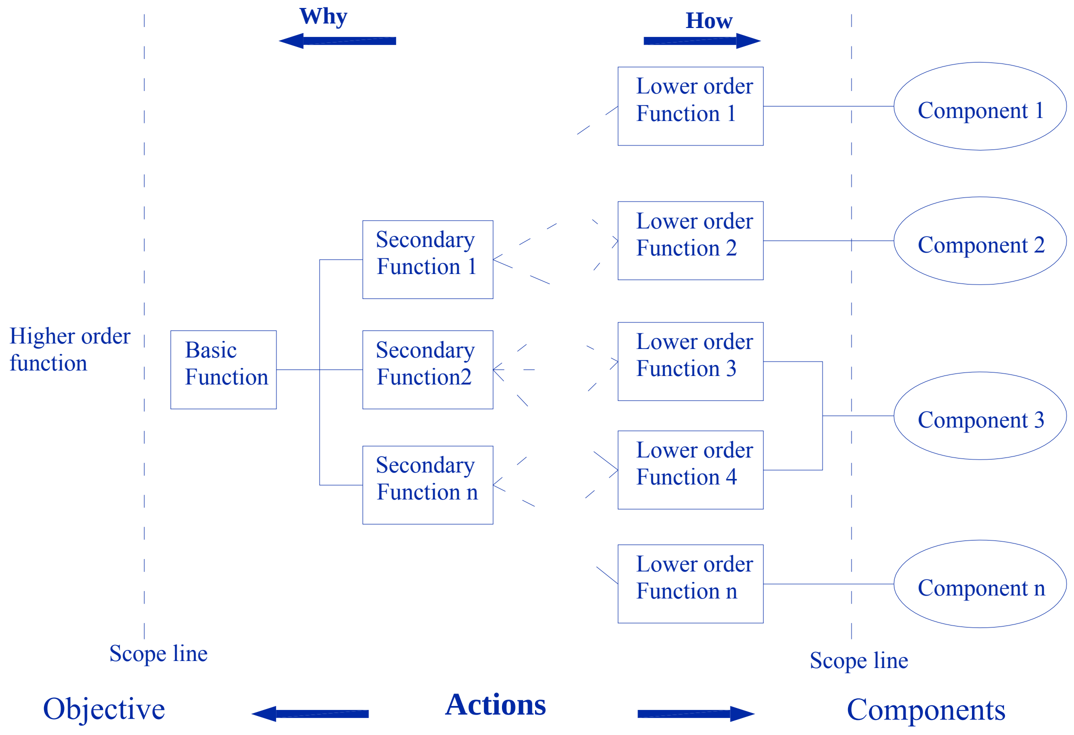


Figure 2.4 The general form of function logic diagram (adapted from (Sturges et al., 1996))

Zhang et al (Zhang et al., 2001a) define construction rules for function decomposition and mapping. In their knowledge-based conceptual synthesizer (KBCS), they used predefined decomposition rules to guide the decomposition and mapping process. In KBCS, the design process starts by defining the goal function i.e., the overall function to be achieved in the working memory. The system searches the behavior base to find the behavior whose functional output matches with the given goal function. If there is no match in the behavior base, function decomposition rule is fired and the goal function decomposed. However, if there is matching behavior in the behavior base, mapping between the function and behavior is done, and the matching behavior will be retrieved into the working memory, where its deriving input is taken to be the new design goal. This process continues recursively until the deriving input of the matched behavior is provided by the working environment. The system prescribe automating the design process but it works in a closed world system, i.e., the system works only for those functions whose decomposition rules are in the knowledge base, requiring new domain knowledge for each product.

An example of the function decomposition rule in the KBCS is given here:

Rule specific_Decompose 1

IF a desired function is Insert terminal into housing

THEN decompose it into Clamp housing after locating it

AND Insert terminal after holding it.

In general from those research works reviewed in this section, it can be summarized that functional decomposition is a subjective process which depends on the designer performing it. Two designers may not come up with the same functional decomposition for the same design problem. Therefore, standardized taxonomies of mechanical functions can be used to indicate at what point to stop functional decomposition. Accordingly, functional decomposition should continue until all the subfunctions can be represented with these standard mechanical functions. This is also important to perform functional design in a repeatable manner.

2.7 Concept Evaluation Process

Conceptual design process can be considered as an activity consisting problem definition, generation of concepts, firm up the generated concepts into concept variants followed by evaluation to decide the best concept for further development. All the subsequent design activities depend on the decision made during the concept evaluation process; therefore, care must be taken not to overlook better design options. At the early stage of design, product concepts always need refinement and are subject to change. However, changes made later in the design stage are costly. To reduce design iteration, and the cost incurred due to this, designers must select product concepts with better performance. Ulrich and Eppinger (Ulrich and Eppinger, 2004), defined concept selection as the process of evaluating concepts with respect to customer needs and other criteria, comparing the relative strengths and weakness of the concepts and selecting one or more concepts for further investigation, testing, or development. Unfortunately, this decision is made at the stage where the designers have incomplete, uncertain, and evolving information about the concepts.

The first task in the concept evaluation process is identifying the concept evaluation criteria including technical and economical characteristics of the concept based on the customer requirement. The criteria should help at least to distinguish one of the alternative concepts from the others. In other words, if all the alternative concepts have same value for a given criterion, that criterion should be eliminated as it has no contribution in making decision.

The most common formal and systematic methods of concept selection methods include Pugh's evaluation method (concept screening method as modified by Ulrich and Eppinger (Ulrich and Eppinger, 2004)) , weighted decision matrix, and analytical hierarchy process (AHP). In addition to these, there are less structured methods used in industry such as concept review meetings, checklists or expert assessment (based on personal preference and expertise), voting on concept variants, and intuitive selection of concepts. The study conducted by Salonen and Perttula (Salonen and Perttula, 2005) revealed that the degree of utilization of formal and systematic concept selection methods in industry is relatively low. On the other hand, the less structured methods possess a higher degree of utilization in the industry than the formal and systematic methods. However, their finding also concludes a higher degree of satisfaction in those companies using one or more formal and systematic concept selection methods compared to those companies who do not use.

An overview of the systematic methods of concept selection approaches is presented next.

2.7.1 Pugh’s Evaluation Method

Pugh’s concept selection method, proposed by Stuart Pugh (Pugh, 1990), is the most widely known and referred concept selection methodology (Pahl and Beitz, 1996, Dieter, 2000, Ullman, 2003, Ulrich and Eppinger, 2004). To use Pugh’s concept selection method, it is required first to choose the evaluation criteria and prepare a selection matrix with the selection criteria on the first column and the alternative concepts on the first row of the matrix. Then, one of the concept variants is selected as a datum concept or a competitive products concept is added as a datum next to the last alternative concept. The evaluation process is performed by comparing each concept variant and the datum concept with respect to each criterion and giving values: “+” if the concept variant is better than the datum, “-” if the concept variant is worse than the datum, or “0” or “S” if the concept variant is same as the datum concept. A score pattern for each concept variant is calculated as the number of pluses, minuses and zeros or S’s. Even though this method can be used to eliminate infeasible concepts it assumes all the criteria are equally important and it did not indicate how much better or worse the concept is compared to the datum. Concept screening method (Ulrich and Eppinger, 2004) is a modified version of Pugh’s concept selection method where the net score of the concept variants is calculated as a sum of pluses and minuses allowing ranking of the concept variants. However, this method also inherits the limitation of Pugh’s concept selection method.

2.7.2 Weighted decision matrix

A decision matrix is a method of evaluating competing alternative concepts by ranking the selection criteria with weighting factors and rating the degree to which each concept variant meets the criterion. In this method, a relative weight is assigned to each criterion since in reality the evaluation criteria markedly differ in terms of importance. The relative weight can be assigned by using either of the following methods (Sen and Yang, 1998):

i. *The direct assignment technique*: where the decision maker assign weights based on his/her experience using certain evaluation standards.

ii. *Pairwise comparison*: In this method each criterion is compared with all other criteria one at a time, and rated in comparison matrix. Comparison matrix is n by n matrix, where the row and column headings are the criteria. A given criterion is rated as '0' if it is less important than the other and as '1' if it is more important. The normalized row value is taken as weight for each criterion. The drawback of this approach is the difficulty in handling the number of comparisons as the number of criterion increases. The other drawback of this approach is that, since there is no intermediate value between '0' and '1', the comparison become coarse and difficult to differentiate.

iii. *Analytical Hierarchy Process (AHP)*: AHP can be used to assign weights to each criterion by comparing with each other like pairwise comparison method. However, instead of using '0' and '1' to compare the criteria, a 9 point scale known as fundamental scale of AHP (Saaty, 1994) shown in Table 2.5 is used to make the comparison finer.

Table 2.5 Fundamental scale of absolute numbers (Saaty, 1994)

Intensity of Importance	**Definition**	**Explanation**
1	Equal importance	Two activities contribute equally to the objective
3	Moderate importance	Experience and judgment slightly favor one activity over another
5	Strong importance	Experience and judgment strongly favor one activity over another
7	Very strong importance	An activity is favored very strongly over another; its dominance demonstrated in practice
9	Absolute/extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation
2,4,6,8	Immediate values between above scale values	Sometimes one needs to interpolate a compromise judgment numerically because there is no good word to describe it.
Reciprocals of above	If element i has one of the above non-zero numbers assigned on it when compared with activity j, then j has the reciprocal value when compared to i	A comparison mandated by choosing the smaller element as the unit to estimate the larger one as a multiple of that unit.

The decision matrix is prepared with the selection criteria and their respective weights in the first and second column of the matrix respectively; whereas the alternative concepts to be evaluated are displayed on the top of the matrix as shown in Table 2.6. Each concept variant is rated with respect to each criterion by using a 5-point scale (Table 2.7) when the knowledge about the criteria is not detailed and an 11-point scale when the information is more complete (Dieter, 2000).

After all the concept variants are rated with respect to each criterion, regardless of the used scale the total score for each concept variant is calculated as the weighted sum of the concept variant's rating given by:

$$R_j = \sum_{i=1}^{n} R_{i,j} = \sum_{i=1}^{n} w_i r_{i,j}$$

Where:

$R_{i,j}$ = weighted score of concept j for the i^{th} criterion

w_i = weighting factor for i^{th} criterion

$r_{i,j}$ = row rating of concept j for the i^{th} criterion

n = number of criteria

R_j = the total score of concept j

Table 2.6 Weighted decision matrix

		Concept Variants (CV)							
		CV -1		CV -2		.		CV -j	
Selection Criteria	**weight**	Rating	Weighted score	Rating	Weighted score	.	.	Rating	Weighted score
C -1	w_1	r_{11}	R_{11}	r_{12}	R_{12}	.	.	r_{1j}	R_{1j}
C -2	w_2	r_{21}	R_{21}	r_{22}	R_{22}	.	.	r_{2j}	R_{2j}
.	.	.	.	.	.	.	.	.	.
C –n	w_n	r_{n1}	R_{n1}	r_{n2}	R_{n3}	.	.	r_{nj}	R_{nj}
Total score		R_1		R_2		.		R_j	

Table 2.7 A 5-point and 11-point scale for concept evaluation (Dieter, 2000).

11-point scale	**Description**	**5-point scale**	**Description**
0	Totally useless solution	0	Inadequate
1	Very inadequate solution		
2	Weak solution	1	Weak
3	Poor solution		
4	Tolerable solution	2	satisfactory
5	Satisfactory solution		
6	Good solution with a few drawbacks		
7	Good solution	3	Good
8	Very good solution		
9	Excellent (exceeds the requirement)	4	Excellent
10	Ideal solution		

2.7.3 Analytical hierarchy process (AHP)

Analytical hierarchy process, developed by Thomas L. Saaty, is a structured multi-criteria decision making framework well suited for evaluation problems whose criteria have a hierarchical structure (Saaty, 1994). In using AHP, both the criteria and the alternative concepts are pair wisely compared as follows:

Step 1:

For each criterion, prepare a square matrix (comparison matrix) in which the set of alternative concept is compared with itself. Each judgment represents the dominance of an alternative concept in the column on the left over an alternative concept in the row on top. It reflects the answers to two questions: which of the two concepts is more important with respect to the criterion under consideration, and how strongly, using the fundamental scale of absolute numbers shown in Table 2.5, for the alternative concept on the left over the alternative concept at the top of the matrix. If the element on the left is less important than that on the top of the matrix, we enter the reciprocal value in the corresponding position in the matrix.

Step 2:

From all the paired comparisons calculate the local priorities (weight) and display them on the right of the matrix. To calculate the local priorities:

i. Normalize the weight by computing the sum of each column and then divide each column by the corresponding sum.
ii. Compute the average values of each row which is the local priority(weight)

Step 3:

Similarly, prepare a pairwise comparison matrix for the criteria, evaluate using fundamental scale of absolute numbers, and calculate the priorities (weights).

Step 4:

Prepare decision matrix, with the local priorities and calculate the final (global) priorities for each alternative concepts.

These steps are demonstrated by taking a hypothetical example with three alternative concepts: A, B and D, and two criteria: C1 and C2 in Table 2.8.

Table 2.8 A hypothetical example to demonstrate AHP with three alternative concepts and two selection criteria

Step 1:

C1: Criteria1

	A	B	D
A	1	3	5
B	1/3	1	1/5
D	1/5	5	1

C2: Criteria2

	A	B	D
A	1	5	7
B	1/5	1	3
D	1/7	1/3	1

Step 2:

C1: Criteria1

	A	B	D	Priority
A	0.65	0.33	0.81	0.6
B	0.21	0.11	0.03	0.12
D	0.13	0.56	0.16	0.28

C2: Criteria2

	A	B	D	Priority
A	0.75	0.79	0.64	0.73
B	0.15	0.16	0.27	0.19
D	0.1	0.05	0.09	0.08

Step 3:

Criteria

i.

	C1	C2
C1	1	5
C2	1/5	1

ii.

	C1	C2	Priority
C1	0.83	0.83	0.83
C2	0.17	0.17	0.17

Step 4:

Decision matrix

	weight	A	B	D
C1	0.83	0.6	0.12	0.28
C2	0.17	0.73	0.19	0.08
Global priority		***0.62***	***0.13***	***0.25***

AHP provides a diagnostic tool for assessing the consistency of the preference and reduces the bias on the decision maker. However, it is a relatively complex method with long decision process, especially when the number of alternative concepts is large with increased number of criteria requiring each alternative concept to be compared with all others for each criterion.

A combination of these concept evaluation methods should be used iteratively in order to select the best concept for further development. In addition some of the concept may be combined to improve their performance and the design process repeated.

2.8 Computer Aided Conceptual Design (CACD) Tools: A Survey

Several commercial computer aided design tools have been developed in the past to reduce the workload of human designer, reduce duration of the product development time, and simplify the design process. Most of these tools are geometry based rather than function-based, concentrating on the later phases of the design process. Thus, the contribution of computers in the conceptual design phase is at its infancy compared to embodiment and detail design phases. The support of computers in the conceptual design phase is lagging behind because:

- The knowledge of design requirements and constraints during this early phase is usually imprecise and incomplete making it difficult to implement (Hsu and Woon, 1998).
- Currently available conventional CAD system does not have built in intelligent system to perform functional reasoning (Zhang et al., 2001b).

Currently, there is no known commercial computer aided conceptual design tool that can be used to design all products in the market. However, there are a number of prototype tools developed in research centers such as MODESSA (Kersten, 1995), Schemebuilder (Bracewell and Sharpe, 1996), web based morphological chart (Huang and Mak, 1999), AIDA (Rentema and Jansen, 2000), EFDEX (Zhang et al., 2001b), 2nd-CAD (Vargas-Hernandez and Shah, 2004), IDEA-INSPIRE (Chakrabarti et al., 2005), and Concept Generator (Bryant et al., 2005). These tools use varieties of approaches for representation and categorization of knowledge. The working principles and the main features of these tools are reviewed next.

MODESSA

Kersten (Kersten, 1995) developed a computer based conceptual design support system known as MODESSA (which is an acronym for MOrphological DESign Support Aid). MODESSA consists of morphological chart, information sheet regarding functions and alternative concepts and weighting table for concept evaluation. Function is described using two words: the action that should be done and material to be handled (e.g. fill case). The user of MODESSA manually selects functions, requirements, and alternative concepts from a database or creates a new one. The design process is performed by selecting one subfunction at a time, and perform manual search for alternatives followed by evaluation of alternatives for the same subfunction. The alternatives of the next subfunction are selected based on the selection of the first alternative and the process continues until the last subfunction. There is no intelligent system to retrieve the design knowledge from the database and most of the actions are dependent on the user. Furthermore, the databases of MODESSA are limited in their coverage.

Schemebuilder

Schemebuilder is a knowledge-based design support tool to assist designers in the conceptual and embodiment design of mechatronic products (Bracewell and Sharpe, 1996). In using Schemebuilder as conceptual design tool, a function used to describe a given design problem must be a member the pre-defined functional embodiment knowledge-base where functions are hierarchically classified as data function, energy function or mass transfer function. The function will gradually decompose and embodied using bond graph based functional decomposition principle. The generated alternative solutions for those functions are represented in the information structure of FEST-ER (Functional Embodiment STructure-Extended Recursively) which is an extension of a traditional function–means tree structure. Unlike traditional function-means tree structure, FEST-ER supports referencing to already embodied functions in case they appear more than once, and embodiment of more than one functions by single means. The designer can terminate those branches in the structure that seems to be infeasible. The generated schemes can be viewed in the integrated 3D modeler and simulated for design verification. Even though, the bond graph model of a design object can be constructed and simulated in Schemebuilder, it has a difficulty to represent functions that are not represented as power flow, inheriting the disadvantage of bond graph technique.

Web based morphological chart

Huang and Mak (Huang and Mak, 1999) developed a prototype web based morphological chart, which consists of five web based modules namely: concept browser/editor, concept base, function analyzer, concept generator and concept assessor. The concept browser/editor allows the user to enter new concepts into the concept base and/or explore its contents. Functional analysis system technique (FAST) is used to create function means tree in the system. The concept generator performs the task of creating the morphology chart from concept base, short listing conceivable means based on feasibility, and combining short-listed feasible means. Among these, short listing feasible means and concept combination process are done manually. The combined concept variants are evaluated using web based Pugh's concept selection chart. The concept base consists of generic functional requirements expressed as goals, potential solution principles expressed as means, and their relationships. However, the concept base has limitations in its coverage, where only eight assembly functions and forty means are stored in its database.

AIDA

AIDA (Artificial Intelligence supported Design of Aircraft), is computer aided conceptual design tool developed by Rentema and Jansen for aircraft design (Rentema and Jansen, 2000). The AIDA system consists of three separate modules and user interface. The first module is case-based reasoning module where case-based reasoning techniques from AI system is used to first retrieve 'a best matching' case from case-based database of previous successful designs. The retrieved case is reused after adaptation for the current design problem. However, the adapted case should first be evaluated before other adaptations can be applied which will be done in the next module i.e., functional module. Functional module supports the execution of parameter studies which includes network of numerical relations using rule-based reasoning technique to evaluate and modify adapted cases produced by the previous module. The rules link functional parameter (from specification) to structural parameters (from design objects) which will be taken as input to the last module, i.e., the geometrical module. Successfully adapted cases will be saved in the database for future reuse. The geometrical module uses constrained aided geometrical modeling technique to display solid model of the suggested aircraft for visualization and to deduce some geometrical information such as volume and area. This module is implemented in Pro/Engineer; a commercial feature based modeling software. The domain of application of AIDA is limited to aircraft design and it is more used in "routine design" type rather than "creative design".

EFDEX

EFDEX (Engineering Functional Design Expert) is a knowledge-based conceptual design tool developed by Zhang et al (Zhang et al., 2001b). They proposed an extended Function-Environment-Behavior-Structure (FEBS) modeling framework as a reasoning strategy in EFDEX. This modeling framework consists of three layers: function layer where the overall functional requirement is decomposed gradually and hierarchically, behavior layer where a set of behaviors are interconnected with each other, and environment layer where the working environment enable the functional output to achieve the requirement in the behavior layer. EFDEX integrates rule-based and object oriented knowledge representation scheme to represent function related design characteristics. They developed 255 domain specific rules to perform functional design of an automatic assembly system for manufacturing electronic connectors. In addition to these, there are general rules that are used to solve general problems such as anti-looping rules to prevent recursive firing and rules to terminate searching branch.

EFDEX uses both backward and forward chaining inferencing mechanisms. The inferencing strategy starts when the user gives as inputs the overall functional requirement. The system first scans the behavior base to find behavior whose functional output can match with the given functional requirement and satisfies the functional constraint. If there is no match found, then the inference engine scans the rule base to search for domain specific production rules to decompose the function, and continue searching for matching behavior for those subfunctions. If there is matching behavior, the behavior is retrieved into the working memory, and its driving input is taken as new functional requirement. The general rule to terminate the search process will be fired at this stage to check if the new functional requirement is available in the working environment. If it is found in the working environment, the branch will be terminated, otherwise the system scans the behavior base and the process continues recursively. Finally potential concept variants produced by the run will be listed and the concept variants evaluated manually. The tool is limited to conceptual design of automatic assembly system for manufacturing systems and to use the tool for other products it requires building domain knowledge specific to that product. Reusing the design knowledge to design products other than its initial intended use is limited. The tool is towards automating the conceptual design process except the concept evaluation is done manually.

2nd-CAD

SECOND-CAD (Systems Engineering CONceptual Design-CAD) or 2nd-CAD is another computer aided conceptual design tool developed by Vergas-Hernandez and Shah for electromechanical systems (Vargas-Hernandez and Shah, 2004). They claim that 2nd-CAD supports conceptual design process specifically in functional design, behavior modeling, and component selection from standard industrial supply catalogs for mechanical, fluid, and electric engineering domains. Among the three main flows (material, energy, and signal), which are widely accepted in the design community, only the flow of energy is considered in the reported version of 2nd-CAD to represent function, behavior and component. They represent behavior using bond graph and preferred mathematical representation of functions rather than linguistic (grammatical) representation considering the computational manipulation. In representing function, behavior and component two options are available in 2nd-CAD for the user: define a new category or select from previously defined categories in the catalogue. To represent structures they propose three types of relationships:

1. Flow relationships to relate the output flow of an element to the input flow of another. The attributes compared depend on the element level, (e.g. for functions only the domain and power type are compared, for behaviors the effort and flow are compared in addition to those for functions, and for components the input/output specifications are compared in addition to those for behaviors).

2. Composition relationships relate parent elements to child elements defining in the process a subsystem hierarchy (e.g. ancestors and descendants).

3. Mapping relationship connects elements from different structures (e.g. function to behavior) to obtain interconnected structures.

The system interacts with the user through graphical user interface receiving inputs and providing outputs. The received input is converted into quires which will be sent into database management system that interacts with the catalogs and structure database. No case study is presented to demonstrate the application of this tool. 2nd-CAD is limited to products having behavior with power flow; this limitation is inherited from bond graph representation used. Furthermore, mathematical representation of functions used in 2nd-CAD limits its area of application; since representing all functions mathematically early at this stage is difficult specially when designing new device.

IDEA-INSPIRE

Chakrabarti et al (Chakrabarti et al., 2005) developed a computational tool known as IDEA-INSPIRE, which can be used for supporting designers in generating solutions for a given design problem based on analogical reasoning methodology mainly for mechanisms design. The software tool consists of two databases: database of natural systems with about 100 entries from plant and animal domain, and database of artificial systems. Each entry in these databases is constructed using the proposed SAPPhIRE (which stands for *State-Action-Part-Phenomenon-lnput-oRgan-Effect*) model of causality in human-understandable form and a computer-understandable form. The user of IDEA-INSPIRE software either browse the entities in the natural and artificial systems database or search for solutions for a given problem through the provided graphical user interface. In searching for solution in the database, the designer first analyze the given design problem and give the action described using a verb, noun, and adjective (VNA) as an input to the software. The program takes these as “input” variables and searches the computer-understandable form of the entries for these variables. If a direct match with the variables is not found, it would search for synonyms of each variable in the entries and give a corresponding weight. The output of the software is a list of matched entries sorted in a descending order of importance to solve the problem. These solutions, in addition to matching with the input, have potential to inspire new solutions. Thus, there are three types of solutions expected from the software for the designer:

1. *Exact solution:* when all the constructs in an entry match with that of the given inputs and it is accepted as the potential solution for the given problem.
2. *Partial solution:* when some of the constructs in an entry match with that of the given inputs and it is accepted as potential partial solution for the given problem.
3. *Inspirational solution:* when an entry with an exact or partial match with the given inputs triggers generation of new solution.

The designer may redefine the problem using different VNA words and repeat the process until satisfactory solutions are found. The tool is limited to concept generation process, i.e., concept combination to create concept variants and their evaluation is done manually.

Concept Generator

Bryant et al (Bryant et al., 2005) proposed a computational based method of concept generation that quickly produces concept variants. Over the course of several years, they have developed a web-based design repository to store design knowledge of about 50 consumer products with collaboration between two universities (University of Missouri–Rolla and University of Texas at Austin). The design knowledge in the repository is collected by reverse engineering process, i.e., dissecting products and recording the product information such as functionality, bills of materials, and design structure matrix (DSM) i.e., the component-component compatibility. The conceptual functional model developed based on functional basis is given by the designer as input to the system together with function component matrix (FCM), i.e., the function-component relationship extracted from the web-based design repository. Then, the set of concept variants are computed using the proposed matrix manipulation on the FCM, and the functional model represented as connectivity matrix. The component-component compatibility is defined by extracting from the design repository in the form of DCM to prune incompatible concept variants. Finally, the system provides ranked list of concept variants based on compatibility of the components. The concept generation program with graphical user interface to automate the concept generation process is created based on the above principle. However, their concept generation program is limited to single non-branching flow chains, thus branching functional model with multiple flows (material, energy, and/or signal) should be divided into single non-branching flow chains with the total number of chains limited to 5. Furthermore, only a maximum of 10 subfunctions is accepted by the software in each chain. Concept evaluation process and compiling the concept variants of each chain to obtain the complete solution are done manually.

Summary of CACD tools

As can be seen from the computer aided conceptual design tools reviewed in this section, there is no known tool that can be used to design all mechanical products. Some of the tools are domain dependent, while others support only part of the conceptual design process like concept generation. Among the tools, MODESSA and web based morphological chart covers the entire conceptual design process (functional analysis, concept generation and evaluation) and have certain similarity with the conceptual design support tool (CDST) developed in this research. The computer aided conceptual design tools reviewed in this section are summarized in Table 2.9.

Table 2.9 Summary of CACD tools

Tool	Capabilities	Main features	Domain	Limitations
MODESSA (Kersten, 1995)	A morphological design support system for functional design, concept generation and evaluation	• GUI containing “morphological overview”, “info sheet” and “weighting table” • contains functional, requirements, design alternatives and previous projects databases. • the designer manually selects functions, requirements, and alternative concepts from databases or create new one.	flexible filling and case packing	• limited in its coverage; • almost every action depends on the users; • has no intelligent system for concept combination and retrievals
Schemebuilder (Bracewell and Sharpe, 1996)	A conceptual and embodiment design tool	• knowledge-based system • bond graph based functional decomposition • FESTER to represent generated concepts • integrated 3D modeler for simulation	mechatronic products	difficult to represent functions not represented as power flow, (limitation on bond graph representation)
Web based morphological chart (Huang and Mak, 1999)	A web based conceptual design support system for collaborative product development	• consists of 5 web-based modules: concept browser/editor, concept base, functional analyzer, concept generator, and concept assessor. • FAST technique to create function means tree, Morphology chart to represent generated concepts • Pugh’s concept selection method	assembly system for electrical plug	• limited in its coverage (only 8 assembly functions and 40 means) • manual concept combination, and selection of conceivable means, • lacks means for documentation of design history for later retrieval and reuse.
AIDA (Rentema and Jansen, 2000)	A design assistant tool for conceptual design of aircraft, using generate and test strategy	• case based reasoning techniques to propose and adapt initial concepts , • rule based reasoning techniques to analyze and evaluate the concept , and • constrained based geometric modeling techniques to model and visualize the proposal in Pro E.	air craft	• limited to aircraft design • used mainly for routine design • did not follow the common conceptual design steps (function-means/concept)

Table 2.9 (continued)

EFDEX (Zhang et al., 2001b)	An expert system for functional design of engineering system	• knowledge-based system • functional decomposition and concept generation using FEBS modeling to create concept variants • automation of conceptual design	automated assembly system for electronic connectors	• extension of the tool to other products requires writing new production rules • concept evaluation is done manually.
2nd-CAD (Vargas-Hernandez and Shah, 2004)	A tool for conceptual systems design in electromechanical domain	• functional design, where functions are represented mathematically, • behavior modeling using bond graph, • component selection from standard industrial supply catalogs, • GUI to receive input and provide output	electro-mechanical	• difficulty in representing non power behaviors (limitation of bond graph representation) • material and signal flows are not included • difficult to represent all functions mathematically.
IDEA-INSPIRE (Chakrabarti et al., 2005)	An inspirational tool which uses design by analogy to generate concepts	• consists of databases of natural and artificial systems, • functions are represented in VNA form • concepts are generated by searching for analogical similarity in the database	mechanism design	covers only the concept generation part of conceptual design process, the remaining parts are done manually
Concept Generator (Bryant et al., 2005)	A concept generation tool utilizing functional basis and design repository	• functional basis for functional representation • web-based design repository of consumer product obtained via reverse engineering method, • computational based concept generation algorithm to generate and rank concept variants.	consumer product	limited to single flow non-branching functional structure

2.9 Summary

In this chapter, research works related with conceptual design process have been reviewed. Literatures covering the entire conceptual design process from functional analysis to concept evaluation were reviewed. In relation with modeling the conceptual design process feature-based, knowledge-based and function-based design approaches have been discussed with their pros and cons. Accordingly a hybrid design approach consisting of function-based design with knowledge-based design approach is selected as a basis for developing the framework of conceptual design process. Research works related to the integration of AI systems, particularly knowledge-based systems, with conceptual design process have been reviewed. In addition different knowledge representation formalisms have been discussed.

Researches related with function-based design approach have been reviewed thoroughly to address the definition and representation of function, and functional classifications (taxonomy of mechanical functions). After analyzing the pros and cons of existing methods; linguistic approach for functional representation and reconciled functional basis for classification of mechanical functions have been adopted.

This chapter also reviews the existing prototype computer aided conceptual design support tools developed so far in research centers. The features and capabilities of those tools have been discussed together with their limitations. These tools vary in their domain of applications, knowledge representation formalisms and coverage. Features and capabilities of the conceptual design support tool developed in this research to address some of the limitations of the existing tools together with major contributions of research are presented in Chapter 7.

In the following chapter, the methodology to achieve the objectives of this research is discussed. Some of the design approaches adopted in this chapter are utilized in devising the methodology.

CHAPTER 3: METHODOLOGY

3.1 Introduction

The problem statement and objectives of the research reported in this thesis have been outlined in Chapter 1. The states of the art and researches in the area of the conceptual design process and computer support tools for conceptual design were reviewed in Chapter 2 together with some of the approaches adopted in this research. In this chapter, the methodology used to develop an appropriate conceptual design model that can be used to build a computer based tool for supporting designers during the conceptual design process, is presented.

After examining the way human designer performs conceptual design process, a conceptual design model is proposed using a systematic design approach together with a knowledge-based system. This model is later used to develop the computer aided conceptual design support tool. Here, a model is considered as a simplified representation of a complex system with the goal of providing predictions of the system's behavior or performance measures (metrics) of interest (Altiok and Melamed, 2007). Models reflect certain features of a real system, i.e., only those aspects intended to be relevant to the characteristic under study. The following steps were used in developing and implementing the conceptual design model:

i. *Problem analysis and information collection*. An extensive literature survey is done to analyze the conceptual design process. The steps in the conceptual design process are examined and areas where human designer needs computer support identified. Furthermore, representations of the collected information are dealt with.
ii. *Data collection*. This includes collecting the necessary domain knowledge about the products to be designed from handbooks, patents, existing products, and experts. In addition, the necessary tools/equipments (in this case programming languages used) for model construction are prepared.

iii. *Model construction*. The collected data are organized and the model implemented into a computer program.

iv. *Model verification*. This to make sure that the model is constructed correctly, and does what it is supposed to do according to its specification.

v. *Model validation*. This is to confirm that the results of the model are acceptable by taking practical case studies and comparing with the models counterpart or perform validation tests.

The first two steps are discussed in the following sections of this chapter, while the remaining steps: model construction, verification and validations are discussed in the next chapters.

3.2 Conceptual Design Process Model

Conceptual design process can be considered as the transformation of design specification which is given as requirement list into one or more concepts that can satisfy these requirements for further development. Careful and extensive exploration of the design space helps not to overlook better design solutions. This is because in most cases there is more than one solution that can satisfy the given requirement. In order to model this process, first how a human designer performs conceptual design following a systematic design approach which is widely used by designers and included in a number of design textbooks (Pahl and Beitz, 1996, Dieter, 2000, Ullman, 2003, Cross, 2008), is examined. In conducting manual conceptual design process using a systematic design approach, the human designer:

i. Analyzes the requirements or customer needs and converts these requirements into the overall function.

ii. Depending on the complexity of the problem, decompose the overall function into less complex subfunctions.

iii. Generates a set of alternative concepts for each function/subfunction by applying knowledge which is in the designer's area of expertise. These generated concepts are posted on a morphology chart.

iv. Synthesize (i.e., combine) the alternative concepts for each subfunction to get a set of concept variants that can satisfy the requirements or customer needs.

v. Evaluate these concept variants based on technical and economical criteria and selects one or two concept variants for further development.

These steps are shown schematically in Figure 3.1.

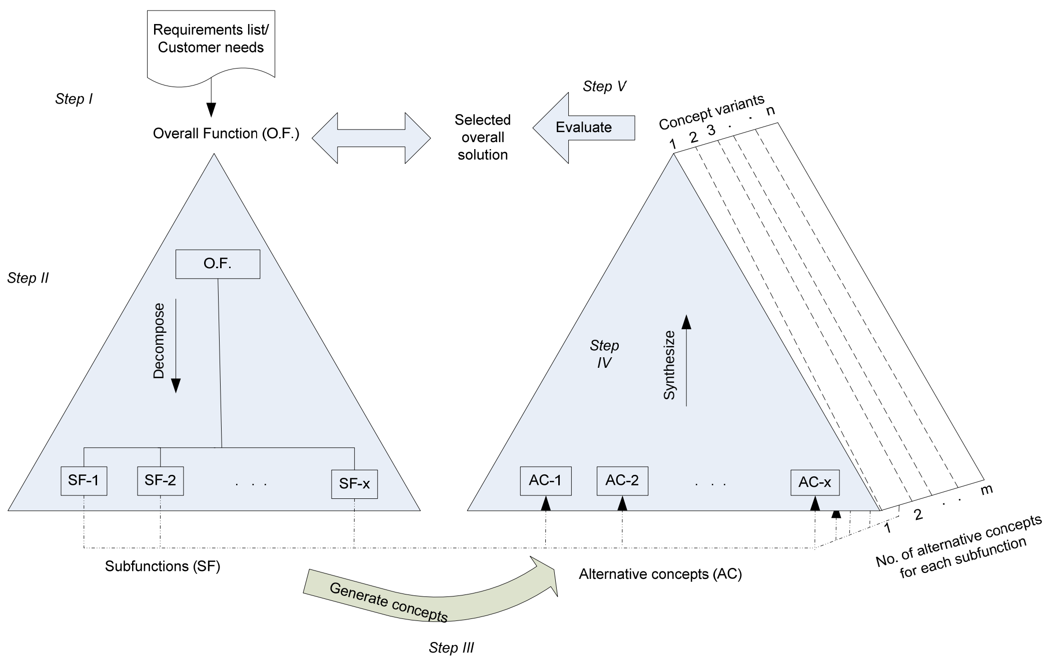


Figure 3.1 Steps in conceptual design process

From the steps in manual conceptual design process described above, the following key issues can be pointed out regarding computer support tools:

- Conceptual design is knowledge intensive process requiring diverse knowledge and its modeling should include a means to store and present design knowledge upon request to augment the knowledge of the designer outside his area of specialization/ scope.
- Concept generation process can be automated provided that the necessary knowledge is acquired and stored in the computer system.
- Some of the tasks like concept combination process and representing the generated concepts on morphology chart are repetitive and time consuming which may be computer supported.

To address these issues, the systematic design approach needs to be integrated with a knowledge-based system to support designers with computer tools. Accordingly, a conceptual design process model, shown in Figure 3.2 is proposed. This model consists of the four major steps in conceptual design: identifying and clarifying the functions required, generating alternative concepts, combining those alternative concepts into concept variants, and evaluation of the concept variants integrated with knowledge-based system.

From this model, it can be seen that the conceptual design process can be considered as a series of activities and achievements. The activities are: functional modeling, concept generation, concept combination and concept evaluation. These activities are done combining the human designer's knowledge and/or the design knowledge-base system. The achievements from a given activity are displayed to the user and given as input to the design knowledge-base system to perform the next activity. The dashed line indicates the information flow between the activities/achievements and the design knowledge-based system. This model is taken as base for the development of the conceptual design support tool (CDST) described in Chapter 4. The detailed descriptions of each activity in the conceptual design model and how the knowledge-based system is incorporated will be discussed in the following sections.

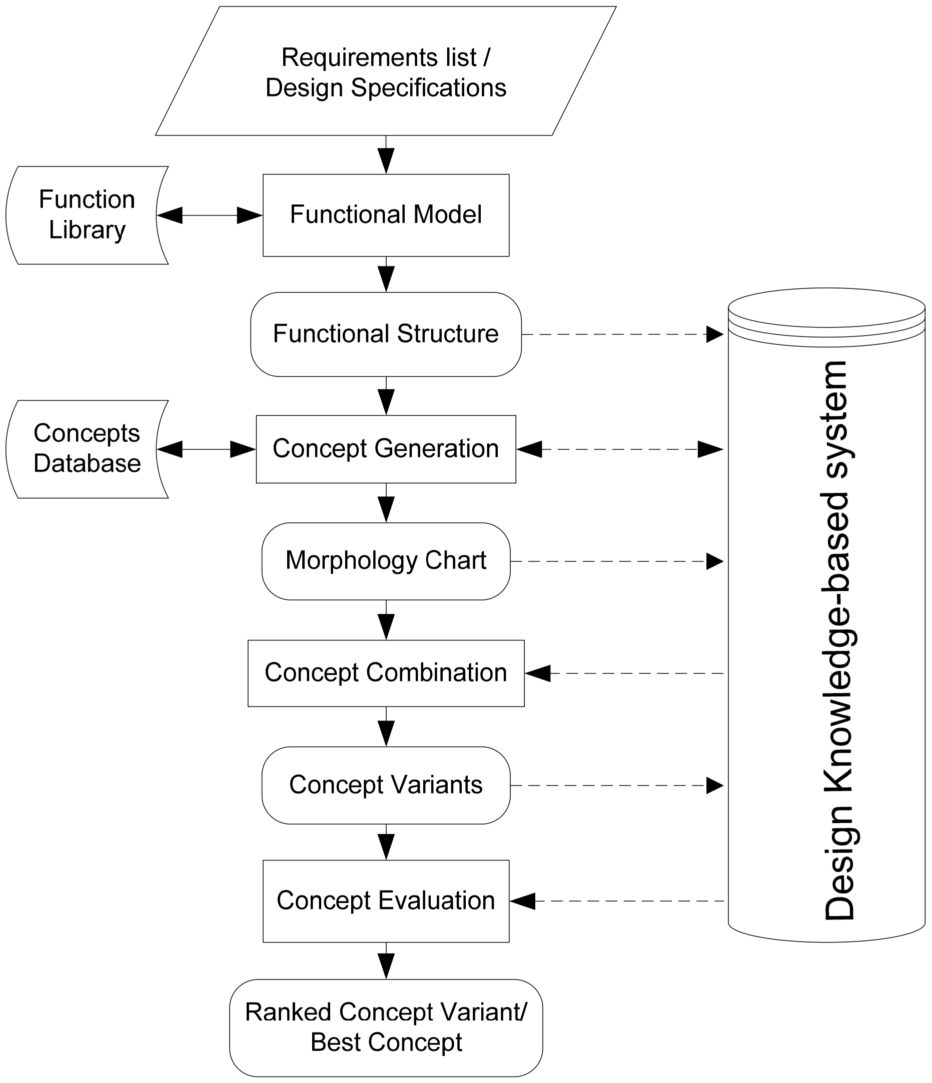


Figure 3.2 The proposed conceptual design model

3.2.1 Functional Modeling

Functional modeling is a process of analyzing the requirement list or design specification to come up with the overall function of the design problem and decomposing this into discrete easily solvable subfunctions to establish functional structure. Functional modeling provides an abstract method for representing and documenting a given design task (Kuttig, 1993). Furthermore, functional modeling permits the designer the ability to view the complete design at the earliest stage. It is well known that form follows functions, and every product has some reason for its existence which is its function.

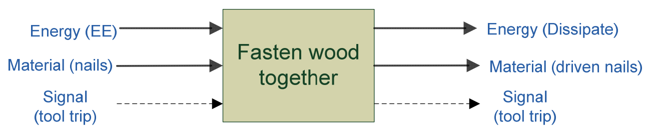
During functional modeling process, the requirement list or the design specification is described in terms of the overall function by the designer. The designer then decompose overall function based on its complexity into subfunctions and the functional structure constructed guided by the function library provided. Function library is constructed systematically from primary, secondary and tertiary function and flow classifications adopted from the reconciled functional basis (Hirtz et al., 2002) as discussed previously in Section 2.5.3. The designer stops decomposing the functions when all the subfunctions in the functional structure can be represented by the functions in the function library. Function is represented using the following attributes/slots:

Function

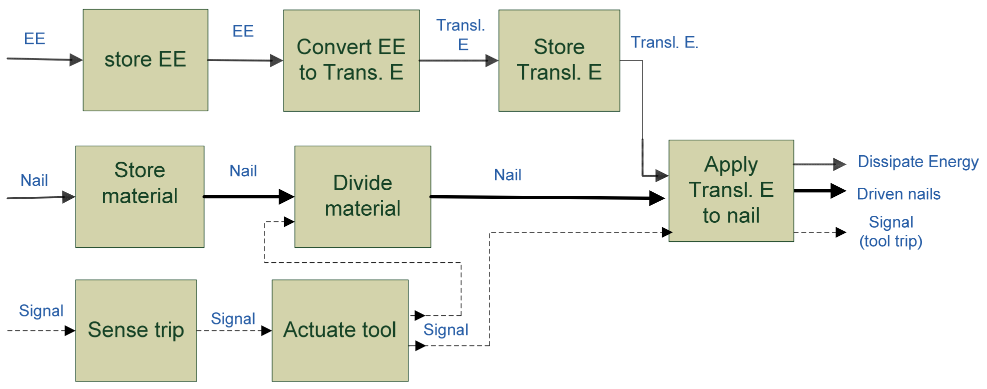
Name:	*verb + Noun*
Complement:	*additional information*
Input:	*{input flows}*
Output:	*{output flows}*
Matched:	*{yes/no}*

In this representation, the name slot is used to describe the function using action verb + noun, the functional class takes the verb position and flow corresponds to noun. The complement slot is used to describe additional information about the function and to represent transformation functions. The input and output slots are used to describe the flows. The matched slot, whose default value is "no", is included to facilitate rule firing during concept generation process.

The process of creating functional structure for a given design problem may not be done in single step. The designer may need to iteratively construct the functional structure until the collective effect of all the subfunctions included in the functional structure represents the required overall function. As an example, a flow based functional structure of hand-held nailer (Ulrich and Eppinger, 2004) represented in black box is shown in Figure 3.3. In constructing this functional structure, it was assumed that electrical energy is given as source of energy in the design specification. If other form of energy was used, a different functional structure would have been obtained. Even for the same assumption taken here, a different functional structure may be obtained if the accumulation of energy is not considered to be in the mechanical domain. This shows the subjective nature of functional modeling, i.e., it is rare to obtain identical functional structure by two designers for the same design problem. Once the functional structure is done manually guided by the function library, the next step is to generate as much alternative solutions as possible that can satisfy each subfunction in the functional structure.



(a) Overall function represented in black box



(b) Functional structure after decomposition

Figure 3.3 Functional structure of hand-held nailer

3.2.2 Concept Generation

Concept generation is the creative and most demanding part of the design process where the designer generates a set of alternative concepts for each function/subfunction by applying a knowledge in his/her area of expertise. A concept is an idea/principle, a component, or an assembly that can be used as a means to provide certain function(s). Concepts can be represented as verbal or textual descriptions, sketches or any other form that gives an indication of how the function can be achieved. The goal of the concept generation step in the conceptual design process model is to generate many concepts quickly and early in the design process by making use of existing design knowledge in the concepts database. This can help to supplement the designer's knowledge by providing concepts outside the scope or areas of expertise of the designer. Furthermore, the generated concept may stimulate the designer to generate new ideas.

An alternative concepts database is developed from which the system searches and generates concepts for a given function. In this research, conceptual design knowledge about subsea process equipments has been obtained from text books and handbooks (Arnold and Stewart, 1999b, Arnold and Stewart, 1999a, Karassik et al., 2001, Lyons and Plisga, 2005, Robert O. Parmely, 2005), patents (Boley, 1985, Saruwatari, 1988, Filho, 1992, Massinon, 1992, Jager, 1994, Hatton, 1998, Ditria and Hadfield, 2001, Nilsen and Wolff, 2005, Lush et al., 2007), manufacturers catalogues, and personal experience and stored in the alternative concepts database. Design knowledge refers to concepts saved in the design knowledge base as facts and production rules used to derive the conceptual design process.

One of the requirements of the computer assistant tool is to accept design knowledge from the user in the course of designing in addition to providing those saved in the database. Thus, when new ideas or technologies are invented or if there is a need to generate new concepts not included in the database manual concept generation methods can be used to expand the alternative concepts database throughout the life of the tool.

In manual design process, the generated concepts are drawn on a morphology chart and posted. Morphology chart or morphological matrix, which was first proposed by Zwicky (Zwicky, 1967), is a matrix consisting of all functions on the first column and alternative concepts on the columns adjacent to each function. Morphology chart represents a methodology for organizing alternative solutions for each subfunctions and combining them to generate a great number of concept variants each of which can potentially satisfy the overall requirement. It is used as a means to record information about the solutions for the relevant functions and aid in the cognitive process of generating design solutions.

Similar to the manual design process, the computer tool should display all the available concepts from the database on morphology chart so that the designers may add some more concepts if they have new idea not included in the morphology chart. Novice designers can also learn from the experience of experts by exploring these concepts or may generate new ideas inspired by the automatically generated concepts. An electronic version of morphology chart that displays all the generated concepts is developed.

3.2.2.1 *Conceptual Knowledge Representation*

Similar to functional representation, the alternative concepts need to be represented in computable manner to facilitate automated concept generation. In principle, the representation of concepts should consist of functionality, behavior, assembly, and technical specifications such as dimensions and material. However, these detail information may not be available at this early stage with the conventional concept generation methods, and if all these are to be fulfilled, it may result in early rejection of ideas because of incompleteness. Thus, the concepts are assumed to be at higher level of abstraction with no dimension or material information. Therefore, the information captured during this study for each concept is only the functionality and input and output flows. Accordingly, alternative concepts are represented in the database with the following attributes/slots:

Alternative-Concept

Name:	*String*
Schematic-representation:	*Sketch*
Input:	*{input flows}*
Output:	*{output flows}*
Primary-Function:	*{function}*
Secondary-Function:	*{function/Nil}*
Other effects:	*{side-effects/Nil}*

Here, the name slot is used to describe the name of the concept and the schematic-representation to include sketches of the concept. The input and output slots are used to indicate material, energy, and/or signal flows of the concept. For concepts where there is no flow, the input and output slots will have nil value. For concepts that can perform more than one function, the secondary-function slot is included. The default value for this slot is nil, but each concept must have at least primary function slot value. Other effects slot is added to include if there is any side effect the concept can cause other than the desired purpose. While the tool is in use, if the designer wants to add new concepts to the database, a simple graphical user interface for knowledge acquisition will be provided where those slot values can be given by the designer as input.

3.2.2.2 *Automating the Concept Generation Process*

The concept generation process is done using predefined domain independent production rules which will be discussed in detail in Chapter 4. The rules are domain independent in the sense that, the rules are represented in terms of variables to generate all concepts regardless of the product to be designed. This is achieved by systematically representing the conceptual design knowledge in computable form together with the pattern matching properties and inferencing mechanisms used. Two types of rules are used for concept generation. The first one is mapping rule, where for all subfunctions in the functional structure from functional modeling process and given as input to the system, the system searches for concepts whose primary or secondary function matches with the subfunction. If there are alternative concepts in the concept base, the subfunction and all the alternative concepts will be included in the morphology chart. However, if there is no alternative concept for one or more of the subfunctions in the database, the second type of rule will be fired. In this case the user will be asked to perform concept generation manually using conventional concept generation methods discussed earlier and give as input the alternative concepts that will be saved in the database for future use. Finally, the subfunctions and their corresponding alternative concepts are displayed on the morphology chart. The next step is to combine those alternative concepts to get concept variants.

3.2.3 Concept Combination

After alternative concepts are generated for each subfunction in the functional structure, the overall function is achieved by combining the concepts. Combinatorial explosion is the main problem in concept combination process. For instance if there are five subfunctions having six, three, six, four, and five alternative concepts respectively for those subfunctions; the total number of concept variants will be 6 x 3 x 6 x 4x 5 = 2160. Practically, it is difficult to evaluate all these concept variants, because of time and resource limitation. One of the means to reduce the number of concept variants is to include compatibility rule, which may be flow compatibility or geometric compatibility. Applying geometric compatibility is not possible at this stage because the concepts are at higher level of abstraction with no geometric information in the concepts base. Instead, flow compatibility rules can be included in the knowledge base since the input and output flows of each concept are captured in the concepts base.

Two types of rules are used to combine the concepts in the knowledge base:

- *General rule to create theoretically possible concept variants*: in this case, the concept variants are created by taking one concept at a time for each subfunction in the morphology chart. Assume that there are z subfunctions in the morphology chart, each having different number of alternative concepts say r, s...t. The pseudo code to generate theoretically possible concept variants is shown in Figure 3.4.

```
Subfunction-1 = { Alt.concept-1, Alt.concept-2, …, Alt.concept-r};
Subfunction-2 = { Alt.concept-1, Alt.concept-2, …, Alt.concept-s};
            .
            .
Subfunction-z = { Alt.concept-1, Alt.concept-2, …, Alt.concept-t};
Concept-variant = 0;
        foreach (x1 in Subfunction-1)
        {
            foreach (x2 in Subfunction-2)
            {
                .
                 .
                  {
                     foreach (xz in Subfunction-z)
                     {
                        Write (Concept-variant + ":" " x1 +", " x2 + ", " . + ", ". +", " xz ");
                        Concept-variant += 1;
                     }
                   . .
                  }
              }
          }
```
Figure 3.4 Pseudo code to create theoretically possible concept variants

- *Flow constrained rule to create flow compatible concept variants*: the synthesis process is the same as the general rule, but flow compatibility constraint is added. Concept variants are considered compatible, if and only if the output flow of the preceding concept is the same as the input flow of the succeeding concept in the morphology chart. This rule is limited to single flow non-branching functional structure. If there is branching functional structure, it needs to be decomposed manually into single flow non branching functional structure before it is given as input to the system.

The concept variants can be displayed both textually and schematically showing all the component concepts that make up the concept variant. The details of the rules used to perform concept combination process will be presented in the next chapter.

3.2.4 Concept Evaluation

Even though, the concept variants obtained during the concept combination process may satisfy customer requirements, it is difficult and impossible to develop all the concepts because of time and cost constraints. All the subsequent design activities depend on the decision made during the concept evaluation process; therefore, care must be taken not to overlook better design options. At the early stage of design, product concepts always need refinement and are subject to change. However, changes made later in the design stage are costly. To reduce design iteration, and the cost incurred due to this, designers must select product concepts with better performance.

From the concept evaluation methods discussed Section 2.7 absolute comparison, concept screening, and weighted decision matrix are used to evaluate the concept variants. Accordingly, the concept variants are first evaluated using absolute comparison method, where concepts are directly compared with set of requirements. This consists of:

- Feasibility judgment i.e., based on the comparison made to prior experience.

- Evaluation based on assessment of technological readiness (state-of-the-art capabilities).

- Evaluation based on go/no-go screening of customer requirements

Some of the infeasible concept variants will be eliminated using the absolute comparison method. The remaining concept variants will be evaluated using concept screening method iteratively taking competitive product or one of the concept variants as a datum. If the competitive product is to be taken as datum concept, it should be reduced to the same level of abstraction as other concept variants. This process will reduce further the number of concept variants. The remaining concept variants will finally be evaluated using weighted decision matrix method. In this method, relative weight is assigned to each criteria based on their importance using either direct assignment technique or pairwise comparison using analytical hierarchy process. The result of the weighted decision matrix is ranked concept variants, from which one or more concepts will be selected for further development or combine some of the concept variants to obtain better performance and repeat the conceptual design process.

3.3 Model Construction

Model construction is the process of converting the conceptual design process model proposed in Section 3.2, into a computer program (CDST) to test and verify the proposed model. The development of CDST follows the “waterfall” model of software development cycle (Figure 3.5), which was adopted by most software professionals (Fisher, 1991, Rakitin, 2001). In the “waterfall” model, the software development cycle starts with the requirement analysis phase which consists of analyzing the basic designer’s requirement in the conceptual design process where computers can support the repetitive and time consuming tasks to get the software requirement or software design specification. This process have been explained in the previous chapters in developing the conceptual design model and pointing out the main areas where computer can better support human designer in terms of functionality requirement and user interface designs. The next phase is design specification, where the software blueprint in the form of general flowchart together with module decompositions is dealt with. The third phase is implementation of the model which consists of coding, testing and debugging each module designed in the design specification. Each built module should be tested individually and integrated into single program structure. The integrated program (software) then verified and validated by testing before the software is being released. Fixing any bugs or problems encountered by users during usage is dealt and maintained in the later phase.

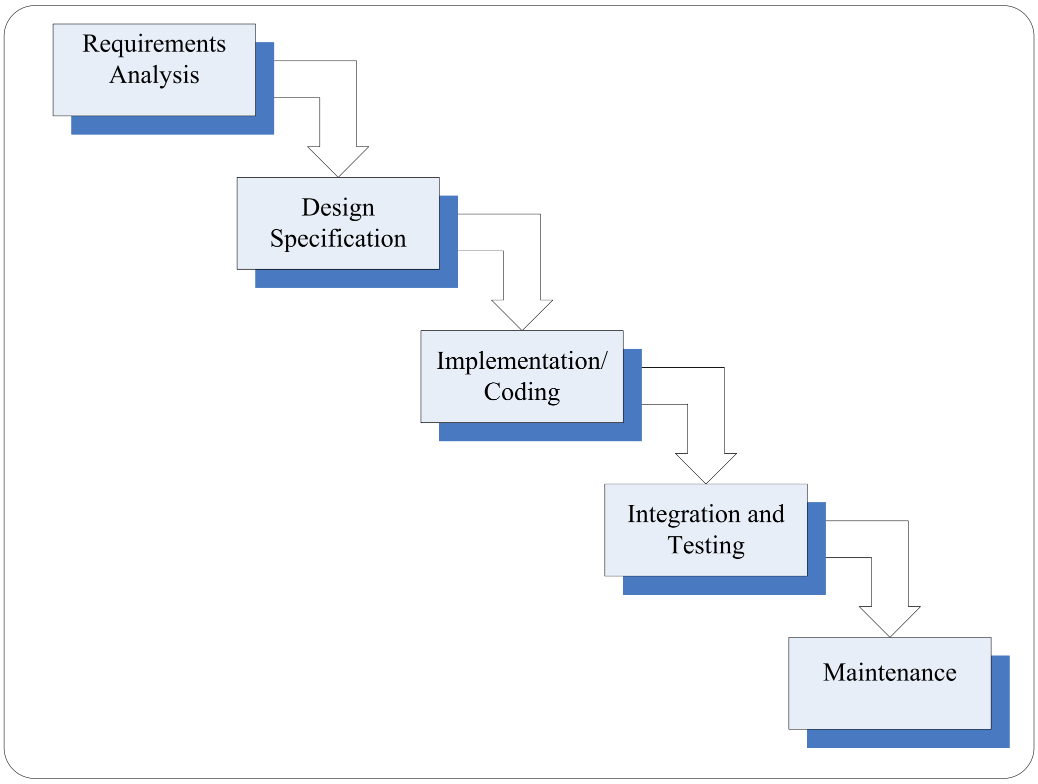


Figure 3.5 The waterfall model of software development (adapted from (Fisher, 1991))

3.3.1 Software Design Specifications and Requirements

It has been pointed out in the previous sections that the main requirement (i.e., the designer's requirement) is to get computer support in performing conceptual design process. Since the requirement is not to automate the conceptual design, the software should provide information to the designer for decision making, and also accept input from the designer. Furthermore, the software needs to accept new knowledge from the current design process and retain it for future use, allowing the knowledge-base to be built incrementally. The software needs to have a graphical user interface (GUI) to interact with the user which can easily be controlled by mouse having windows, icons, and menus giving visual feedback about the actions being performed.

For ease of implementation in coding the software, the conceptual design process is decomposed into four modules: functional modeling, concept generation, concept combination, and concept evaluation. Each module is built based on the proposed framework separately but in a compatible manner where the outputs and inputs of the consecutive parts are matched. In addition to these modules, there is a central graphical user interface which helps to hide all the programming details from the user and link all the modules in seamless way. The system interacts with the user, the hardware, and other systems such as database and software through its interface. The user interface may employ questions and answers, menu driven system, or graphical interface. The user interface simplifies communications and hides much of the systems complexity. The main requirements for the user interface in this research are: it should be easy to use and give visual feedback about actions performed such as displaying the schematic representation of each alternative concept. Thus, a graphical user interface (GUI) which can easily be controlled by mouse having windows, icons, and menus is developed for the user to interact with the computer.

The other requirement from the proposed framework is that, the system acquires knowledge from the current design process in addition to providing existing concepts during the concept generation process. Knowledge acquisition is the process of collecting the knowledge necessary for problem solving and encoding it in the form suitable for computer manipulation. This part is included to ease the addition of new alternative concepts throughout the life of the tool with mouse and keyboard driven graphical user interface dedicated for this purpose. From the human understandable form in the graphical user interface, the inputs are converted into machine understandable form by the system and saved in the database.

The overall flow diagram (blueprint) of the software (i.e., CDST) is shown in Figure 3.6. This flow diagram will be used to build the software in Chapter 4 and as a verification document in Chapter 6.

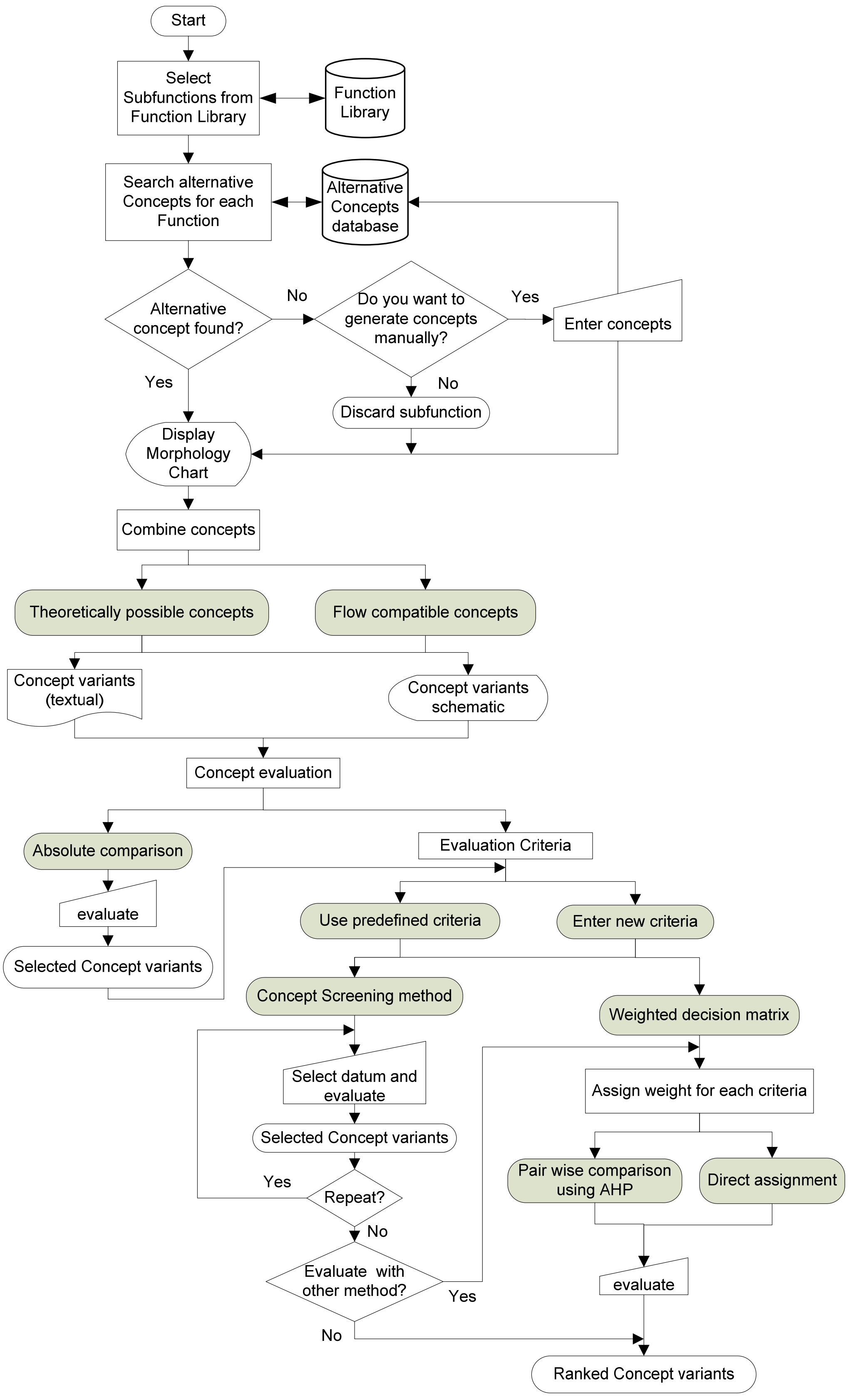

Start
Select Subfunctions from Function Library
Function Library
Search alternative Concepts for each Function
Alternative Concepts database
Alternative concept found?
No
Do you want to generate concepts manually?
Yes
Enter concepts
Yes
No
Discard subfunction
Display Morphology Chart
Combine concepts
Theoretically possible concepts
Flow compatible concepts
Concept variants (textual)
Concept variants schematic
Concept evaluation
Absolute comparison
Evaluation Criteria
evaluate
Use predefined criteria
Enter new criteria
Selected Concept variants
Concept Screening method
Weighted decision matrix
Select datum and evaluate
Assign weight for each criteria
Selected Concept variants
Yes
Repeat?
Pair wise comparison using AHP
Direct assignment
No
Evaluate with other method?
Yes
evaluate
No
Ranked Concept variants

Figure 3.6 Flow chart of the CDST

3.3.2 Programming Environment

The selection of the programming environment is done after analyzing the requirements and specifications set in the previous section. The primary objective is to select programming language and construct the representation and control structures required for performing conceptual design process. A number of programming environments are available in the market for knowledge-based system development. The availability, cost and capabilities in handling the type of knowledge (i.e., symbolic and schematic) were the factors used to select the languages.

Prolog, one of the artificial intelligence programming languages in earlier days, was first considered. Prolog is a declarative language (i.e., a language that expresses the logic of a computation without describing its control flow), which uses backward chaining inference mechanism. An expert system shell based on prolog known as Flex Expert Systems Toolkit from Logic programming Associates Ltd (LPA, n.d.) was obtained as a free trial version to experiment on it. Because of the nature of the knowledge to be represented (i.e., symbolic and schematic), Prolog needs to be integrated with other programming language, since it is purely a symbolic language and cannot accept sketches. Furthermore, it incurs additional cost to purchase as it is not readily available.

The other programming environment considered was CLIPS (C Language Integrated Production System). CLIPS is a public domain expert system shell (Riley, 2008), which was initially developed by the Software Technology Branch (STB), NASA/Lyndon B. Johnson Space Center. CLIPS is a forward-chaining, rule-based production-system language, based on the RETE algorithm for pattern-matching (Giarratano and Riley, 1998). CLIPS allow hybrid knowledge representation including rule based, user defined functions and object-oriented programming in one. CLIPS is also a symbolic language, which cannot support graphical representation of knowledge such as sketches. For this application, wxCLIPS which is an extension of CLIPS to develop knowledge-based system applications with graphical interface was readily available as public domain software (Smart, 1997). An initial study was conducted to develop a prototype conceptual design tool for subsea process equipments using wxCLIPS (Woldemichael and Hashim, 2007). The result was promising even though there were some limitations such as flexibility in terms of creating different GUI components and compatibility (i.e., wxCLIPS was developed based on CLIPS 5.1, and not updated to be compatible with the current version of CLIPS 6.30). Because of these reasons another alternative programming environment compatible with the current version of CLIPS was explored.

After thoroughly investigating possible options, wxPython is considered as graphical user interface (GUI) development environment. wxPython is a public domain cross-platform wrapper for the GUI application programming interface (API) wxWidgets for the Python programming language (Dunn). The interface between Python and CLIPS can be done by PyCLIPS which is also public domain open source software (Garosi, 2008a). Thus all the programming environments selected are freely available under public domain license.

3.4 Summary

In this chapter the methodology used to achieve the objectives of this research was presented. Specifically a conceptual design process model has been proposed after analyzing manual conceptual design process and identifying areas where computer support can be introduced. Accordingly, the framework of function-based conceptual design process integrating systematic design approach with knowledge-based system is presented. In this framework, the entire conceptual design process is divided into four modules representing the major activities in performing conceptual design. These modules are: functional modeling, concept generation, concept combination and concept evaluation. In each module the knowledge representation formalisms, their inputs and outputs together with the processes to achieve these, and the activities done by the designer and the computer were identified.

To verify the proposed methodology, a roadmap to convert the proposed conceptual design process model into a computer program in the form of flowchart is also presented in this chapter. In addition the following public domain open source programming environments to build CDST were selected:

- CLIPS to build the knowledge-base,
- wxPython to develop GUI and represent schematic knowledge, and
- PyCLIPS to integrate the CLIPS with Python.

In the next chapter, the selected programming environments are used to develop the conceptual design support tool based on the methodology proposed in this chapter.

CHAPTER 4: DEVELOPMENT OF CONCEPTUAL DESIGN SUPPORT TOOL (CDST)

4.1 Introduction

In this chapter, the conceptual design process model proposed in Chapter 3 is converted into computer program using CLIPS, wxPython, and PyCLIPS as programming languages. Figure 4.1 shows the programming languages used to develop each architectural components of the knowledge-based system. Knowledge in the form of rules, facts, and functions (i.e., in the form of procedural program) are represented in CLIPS. CLIPS has built in forward chaining inference engine to control the knowledge. The graphical user interface is built using wxPython which is a Python module itself. PyClips is used to embed CLIPS in Python and write production rules within Python programming environment (Garosi, 2008b). In general, CLIPS is used for the knowledge-base, the inference engine, the knowledge acquisition, and the interface. On the other hand, the GUI which includes the knowledge acquisition and displaying the output of the program both textually and schematically is built using wxPython.

The main programming language (i.e., CLIPS) to develop CDST is introduced first to familiarize the reader with knowledge representation formalism. The software development process is divided into different modules for ease of implementation and testing. Sample CLIPS codes for the constructs and rules used in each module are presented and explained. However, the detail construction of the GUI is not explained in this chapter. The source code for the entire program consists of thousands of lines and several files. Considering the number of pages it takes, only screenshot of the excerpt from the source code is presented in Appendix A. The complete source codes and other file formats are found in the attached CD.

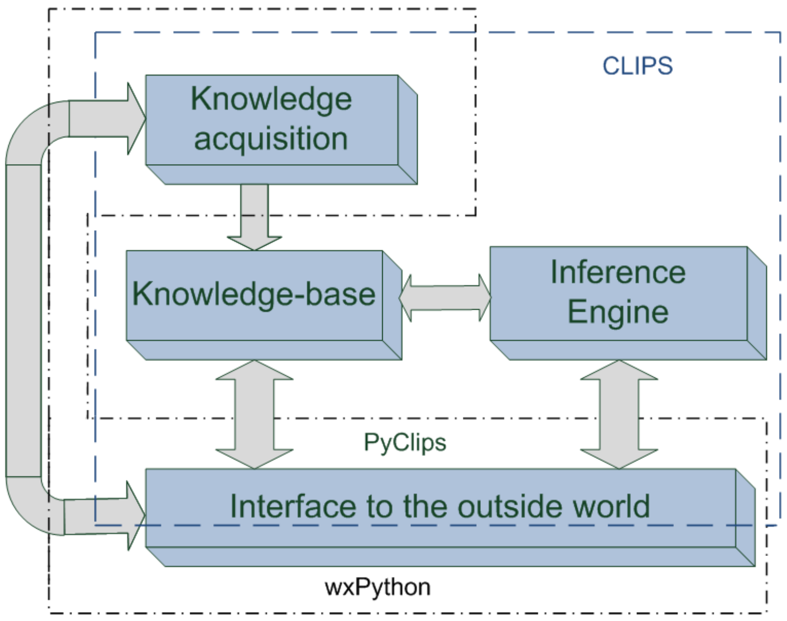


Figure 4.1 The structure of a knowledge-based system and programming environments used

4.2 Development of the Software

Converting the software requirement or design specification into code is the most demanding and time consuming phase of the software development cycle. In this section, detail description of each module in the software development with regards to the knowledge-based system and GUI is presented. To make the implementation easier to understand, and familiarize the reader with CLIPS programming used in this thesis for knowledge-based development, the basic programming elements are introduced next. The GUI is built using wxPython and its codes are not discussed here, except some screenshots to display the result.

4.2.1 Basic Programming Elements and Knowledge Representation Formalism in CLIPS

CLIPS provides three basic elements for writing programs: primitive data types for representing symbolic and numeric information, functions for manipulating data, and constructs for adding to a knowledge-base (Giarratano, 2007). These basic elements are used in representing knowledge in rule-based and procedural programming within CLIPS.

These programming paradigms work based on the information represented and saved in the system in the form of facts and/or global variables. Facts are pieces of information/data required by the CLIPS program to reason out in solving a given problem. Facts consist of a relation name followed by zero (if it is ordered facts) or more slots (if it is non-ordered facts), and their associated values. Facts may be added, removed, modified or duplicated in the system (fact list) using assert, retract, modify, or duplicate commands respectively.

Function in CLIPS is a piece of executable code identified by a specific name which returns a useful value or performs a useful side effect (such as displaying information). Even though CLIPS supports both user defined functions, (i.e., functions written externally with other languages and linked with CLIPS), and system defined functions, (i.e., functions that have been defined internally by the CLIPS environment) only the system defined functions were used here.

From the several defining constructs provided by CLIPS to add information to the knowledge-base, the followings were used:

- *Deftemplate:* a construct used to create a template which can then be used by non-ordered facts to access fields of the fact by name.
- *Deffacts:* a construct that allows a list of facts to be defined which are automatically asserted whenever the reset command is used.
- *Defglobal:* a construct that allows variables to be defined which are global in scope throughout the CLIPS environment.
- *Deffunction:* a construct that allows the user to define new functions (i.e., system defined function) in CLIPS directly.
- *Defrule:* a construct that allows defining rules.
- *Defmodule:* a construct which allows knowledge-base to be partitioned.

All constructs in CLIPS are surrounded by parentheses. The construct opens with a left parenthesis and closes with a right parenthesis. Comments can be added to the CLIPS code to make it easier to understand. All constructs (with the exception of defglobal) allow a comment directly following the construct name. Comments can also be placed within CLIPS code by using a semicolon (;). Everything from the semicolon until the next return character will be ignored by CLIPS. If the semicolon is the first character in the line, the entire line will be treated as a comment.

The rule-based programming paradigm in CLIPS provides a means to represent knowledge in the form of rules. Rules are used to represent heuristics, or "rules of thumb", which specify a set of actions to be performed for a given situation. A rule is composed of an antecedent (the if portion or the left-hand side (LHS) of the rule) and a consequent (the then portion or the right-hand side (RHS) of the rule).

The antecedent of a rule is a set of conditions (or conditional elements) which must be satisfied for the rule to be applicable. In CLIPS, the conditions of a rule are satisfied based on the existence or non-existence of specified facts in the fact list. One type of condition which can be specified is a pattern. Patterns consist of a set of restrictions which are used to determine which facts satisfy the condition specified by the pattern. The process of matching facts to patterns is known as pattern-matching. The inference engine of CLIPS provides a mechanism, which automatically matches patterns against the current state of the fact list and determines which rules are applicable.

The consequent of a rule is the set of actions to be executed when the rule is applicable. The actions of applicable rules are executed when the CLIPS inference engine is instructed to begin execution of applicable rules. The preferred mechanisms in CLIPS for ordering the execution of rules are salience and modules.

Salience allows for explicitly specifying one rule to be executed before another. If more than one rule is applicable, the inference engine uses a conflict resolution strategy to select which rule should have its actions executed first. A conflict resolution strategy is an implicit mechanism for specifying the order in which rules of equal salience should be executed. CLIPS provides seven conflict resolution strategies; among these the most common ones are:

- *Depth strategy*: - newly activated rules are placed above all rules of the same salience.

- *Breadth strategy*:- newly activated rules are placed below all rules of the same salience.

- *Simplicity strategy*: - among rules of the same salience, newly activated rules are placed above all activations of rules with equal or higher specificity. The specificity of a rule is determined by the number of comparisons that must be performed on the LHS of the rule.

- *Complexity strategy*: - among rules of the same salience, newly activated rules are placed above all activations of rules with equal or lower specificity.

The default strategy is depth, but new strategy can be set by using the “set-strategy” command, which will reorder the agenda based upon the new strategy. Agenda is the lists of rules whose conditions are satisfied and have not yet been executed. Each module has its own agenda. The agenda acts similar to a stack where the top rule on the agenda is the first one to be executed. When a rule is newly activated, its placement on the agenda is based on the following factors:

a. Newly activated rules are placed above all rules of lower salience and below all rules of higher salience.

b. Among rules of equal salience, the current conflict resolution strategy is used to determine the placement among the other rules of equal salience.

c. If a rule is activated (along with several other rules) by the same assertion or retraction of a fact, and steps (a) and (b) are unable to specify an ordering, then the rule is arbitrarily ordered in relation to the other rules with which it was activated.

Modules allow to explicitly specify that all of the rules in a particular group (module) should be executed before all of the rules in a different group. In addition, CLIPS modules also allow a set of constructs to be grouped together such that explicit control can be maintained over restricting the access of the constructs by other modules. Two defmodule constructs (AUXILIARY and MAIN) are created in CDST to provide rule execution control.

The syntax of the CLIPS constructs used in each module are presented in the following sections.

4.2.2 The Functional Modeling Module

Functional modeling is the first part responsible for creating the functional structure in CDST. This consists of the function library and the necessary attributes required to define and represent a given function. Function is represented in CLIPS by creating a deftemplate construct as shown in Figure 4.2[1], which is the implementation of function representation explained in Section 3.2.1. The first line in this figure (Line 1) describes the module (i.e. AUXILIARY in this case) in which the template named “function” is defined. Lines 2-7 define the slots, slot name, and default value (if any). Slots can also be constrained by value, type and numeric range. A slot can hold either a single-field value (defined as slot) or multi-field value (defined as multislot). Thus, when the keyword slot is specified, the slot can hold one value, where as when the keyword multislot is specified, the slot can hold a multifield value comprised of zero or more fields.

```
1          (deftemplate AUXILIARY::function
2          (multislot verb)
3          (multislot noun)
4          (multislot complement)
5          (multislot input)
6          (multislot output)
7          (slot matched (default no)))
```

Figure 4.2 A deftemplate construct to define function in CLIPS

Once the deftemplate is defined, any function can be defined using the construct specified. For example a function transmit rotational energy can be defined as:

[1] In this thesis, examples of CLIPS code are presented annotated with line numbers on the left. Please note that these line numbers are not part of the CLIPS program, instead introduced in this thesis to facilitate easy reference to particular line in the CLIPS program.

```
(function
        (verb           transmit)
        (noun           rotational energy)
        (input          rotational energy)
        (output         rotational energy))
```

In this case, those slots defined in the template and have no value during definition will be considered as nil, unless they have default value. Therefore, the actual definition of this function in the system is:

```
(function
        (verb           transmit)
        (noun           rotational energy)
        (complement     nil)
        (input          rotational energy)
        (output         rotational energy)
        (matched        no))
```

Based on the deftemplate defined in Figure 4.2 a function library from which the designer selects elementary functions is built. The function library is database of all functions and flows from the reconciled functional basis (Section 2.5.3). Using the developed GUI (Figure 4.3) the user selects the subfunctions in the functional structure from the function library.

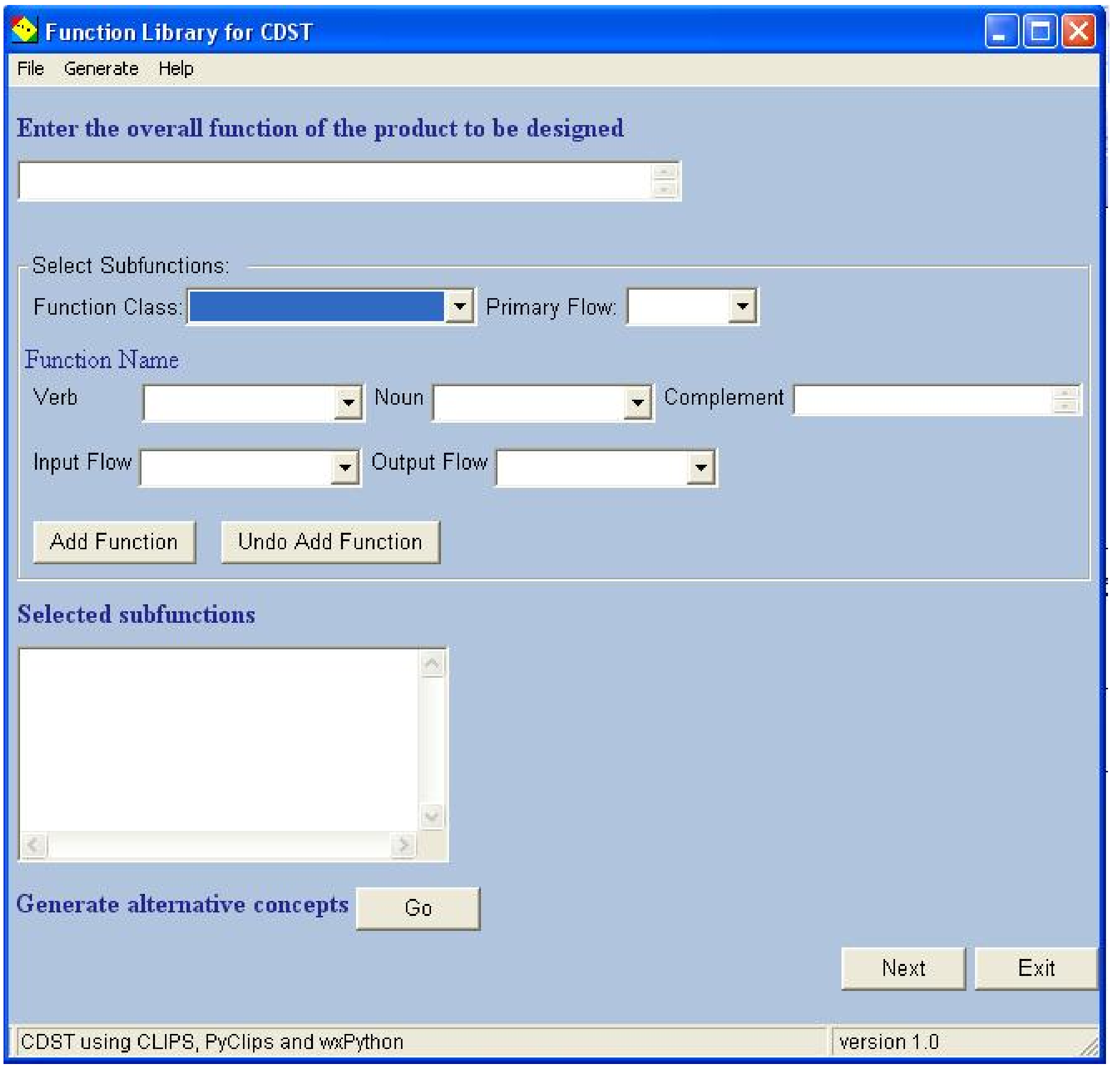


Figure 4.3 Screenshot of the function library

The functional modeling process which consists of decomposing the overall function into subfunctions is basically done manually by the designer. The program assists the designer in defining those subfunctions by providing the function library.

The function library shown in Figure 4.3 starts by accepting input from the designer textually about the overall function of the product to be designed. Then, each subfunction in functional structure is selected from the library and added to the system (fact list). To define a subfunction, first the user selects the functional class, representing the primary category (class) in the reconciled functional basis, having eight choices: branch, channel, connect, control magnitude, convert, provision, signal, and support. Each functional class brings the corresponding secondary and tertiary categories to populate the choices of the "verb" field of the function name. Therefore, when one functional class is selected, the corresponding secondary and tertiary categories will be available as verb choice items in the function name, which was initially empty. Similarly, the user selects one option from the primary flow class which has three choices: energy, material, and signal. This selection will automatically populate the noun field of the function name and the input and output flow choices, with secondary and tertiary categories of flow sets in reconciled function basis. If the definition of the subfunction requires additional information, the user can give the information textually in the complement text field. After all the choice items are selected, the "Add Function" button is used to add (assert) the subfunction to current system (fact list), which will display the added subfunction in the "Selected subfunctions" window. All the subfunctions in the functional structure are added to the system following the same procedure. The next step is to generate alternative concepts for those subfunctions. This will be discussed in the next module.

4.2.3 Concept Generation Module

The input to the concept generation module is the set of subfunctions from the functional modeling in the form of functional structure and its output is set of alternative concepts displayed on morphology chart that can satisfy those subfunctions. In order to generate alternative concepts for functions, the system requires knowledge in terms of facts in the alternative concepts database, and the necessary rules to perform the matching.

Similar to functional representation, a deftemplate construct is required to represent and save alternative concepts in the alternative concepts database. The alternative concepts deftemplate construct shown in Figure 4.4, is constructed based on the methodology devised in Section 3.2.2. The first line in this figure (Line 8) describes the module (i.e. MAIN) in which the template named "alternative-concept" is defined. The remaining lines 9-19 define the slots. In the slot definition, three types of inputs and outputs are given (Lines 11-16); these are included to differentiate material (-m), energy (-e) and signal (-s) flows. The slots function-1 and function-2 stands for the primary function and secondary function respectively for each concept.

```
8     (deftemplate MAIN::alternative-concept
9     (slot name)
10    (slot schematic-representation)
11    (multislot input-m)
12    (multislot input-e)
13    (multislot input-s)
14    (multislot output-m)
15    (multislot output-e)
16    (multislot output-s)
17    (multislot function-1)
18    (multislot function-2)
19    (multislot other-effect))
```

Figure 4.4 A deftemplate construct to define alternative concepts in CLIPS

The alternative concepts database is built based on the deftemplate constructed in Figure 4.4. Known design knowledge in the form of facts can be collected and saved using deffacts construct in the database. For example, if there are known concepts in a given domain, then those facts can be grouped together with single name and saved in the database as shown in Figure 4.5. This defined list of facts can be loaded to the system and automatically asserted with a "reset" command in CLIPS. Line 20 in this figure represents the name of the deffacts construct (i.e., concepts-xx in this case). There are three concepts defined in this construct (electric motor, shaft, and linear actuator) which may be expanded by adding more concepts. The schematic representation of each concept can be sketched on CAD software or manual sketches may be scanned and saved in the database with the same file name as the alternative concepts name in wxPython supported file format. Some of the wxPython supported file formats includes: windows bitmap (BMP), joint photographic experts group (JPEG), graphic interchange format (GIF), interchange file format (IFF), tagged image file (TIF), portable network graphics (PNG), and windows icon format (ICO). The alternative concepts database contains similar deffacts construct which are defined during program development. New design knowledge may also be added while using the software using the knowledge acquisition provided.

```
(deffacts concepts-XX
  (alternative-concept  (name electric-motor)
             (input-e electrical energy)
             (output-e rotational energy)
             (function-1 supply rotational energy)
             (function-2 convert electrical energy to rotational energy))
  (alternative-concept  (name shaft)
             (input-e rotational energy)
             (output-e rotational energy)
             (function-1 transmit rotational energy))
  (alternative-concept  (name linear-actuator)
             (input-e hydraulic energy)
             (output-e translational energy)
             (function-1 supply translational energy)
             (function-2 convert hydraulic energy to translational energy))
)
```

Figure 4.5 Sample alternative concepts representation in the database

Once the alternative concepts database is built, the next step is to develop rules that can use the database to generate concepts. Rules are defined in CLIPS using defrule construct. The general syntax of defrule construct is shown in Figure 4.6. Each rule in CLIPS is identified with unique name (line 36); redefining another rule with the same name will overwrite the previous rule. Optional comments may be placed next to the rule name on line 36. Line 37 describes optional declaration of the rule property such as salience to guide the order of firing (executing) rules. The next part is the main part of the LHS of the rule which consists of a series of conditional elements that must be satisfied for the rule to be placed on the agenda. There are eight types of conditional elements: pattern, test, and, or, not, exists, for all, and logical conditional elements. An implicit "and" conditional element always surrounds all the patterns on the LHS. The pattern conditional element is the most basic and commonly used conditional element containing constraints which are used to determine if any pattern entities (facts) satisfy the pattern. The arrow on line 39 (=>) separates the LHS from the RHS. The RHS contains a list of actions to be performed when the LHS of the rule is satisfied. There is no limit to the number of conditional elements or actions a rule may have; other than the limitation placed by actual available memory. Actions are performed sequentially when all the conditional elements on the LHS are satisfied.

```
36      (defrule module:: <rule-name>
37      [<declaration>]                   ; Rule Properties (optional)
38      <conditional-elements>            ;  Left-Hand Side (LHS)
39      =>
40      <action>)                         ; Right-Hand Side (RHS)
```

Figure 4.6 A general syntax for defrule construct

The concept generation rules are built considering the pattern matching properties of the rules in CLIPS. The main objective in defining those rules is to provide generic rule that can be used to generate alternative concepts regardless of the product to be designed. This is achieved by systematically representing the design knowledge and the heuristic rules in the knowledge-base. Hence, the concept generation rules are represented in terms of variables, which provide the following advantages:

1. There is no need to write individual rules for each product to be designed by the tool. This reduces the total number of rules required.
2. The tool becomes domain independent, i.e., theoretically any product can be designed using the tool provided that the design knowledge is available in the database and represented in terms of the functions in the reconciled functional basis together with the knowledge representation formalism used in this thesis.
3. Future knowledge addition does not require the program to be altered.

The domain independent rule is meant to find and display all the alternative concepts for each subfunction in functional structure which are given as input to the current working memory. The first mapping rule which search the database for concepts whose primary function matches with the given subfunction is shown in Figure 4.7. In this rule a salience of 500 is declared (line 42), making the rule top priority. Line 43 restricts the rule to be fired only when the user choose to search for primary functions. Line 44 describes function in terms of variables which is not matched yet to be matched with the alternative concept (line 45) whose primary function is the same as the function given. Variable are represented using "?" and symbol if it is single field slot and "$?" and symbol if it is multifield value. If the condition on the LHS of this rule is met, then the function and the alternative concept are included in the morphology chart (line 47). The rule will fire repeatedly until all the alternative concepts in the database satisfying this condition are retrieved. Similarly, there is another rule that searches the alternative concepts database for secondary function changing the values of line 43 and 45.

```
41      (defrule MAIN::Mapping-rule-PrimaryFunction
42       (declare (salience 500))
43      (functiontype primary)
44      ?func <- (function (verb ?verb) (noun $?noun) (complement $?comp) (matched no))
45       ?alt <- (alternative-concept (name ?name) (function-1 ?verb $?noun $?comp))
46        =>
47       (assert (morphologyChart (functions ?verb ?noun ?comp) (alternative-concepts ?name) (concept ?alt))))
```

Figure 4.7 Sample mapping rule

To prevent the rules form repeated firing indefinitely, another type of rule with less salience is introduced to modify the property of the matched function from “no” to “yes” as shown in Figure 4.8, line 56. The matched slot value could have been changed in the previous rule by modifying the property of the function in the action part of the rule (Figure 4.7), had the mapping between function and alternative function been one to one. All the concept generation rules are activated by “Go” button next to the “Generate alternative concepts” text in the function library (Figure 4.3).

```
48      (defrule MAIN::Mapping-rule-Matched
49      (declare (salience 400))
50      (or (functiontype primary)
51         (functiontype secondary))
52       ?func <- (function (verb ?verb) (noun $?noun) (complement $?comp) (matched no))
53      (or  (alternative-concept (name ?name) (function-1 ?verb $?noun $?comp))
54         (alternative-concept (name ?name) (function-2 ?verb $?noun $?comp)))
55       =>
56      (modify ?func (matched yes)))
```

Figure 4.8 Mapping rule to prevent repeated rule firing

There may be one or more subfunction that is not matched yet with those concept generation rules, if the conditions are not met (i.e. if there is no alternative concept in the database whose either primary or secondary function matches with the subfunction). In such cases, another type of rule is required to handle this particular situation. The main purpose of this rule is to notify the user about the unavailability of concept in the database for that particular function (line 67 in Figure 4.9). When there is no alternative concept in the database, the rule first remove the function from the current fact list (line 65) and then introduce a new buffer function (line 66) whose alternative concept is going to be generated manually by the user. Notice that, the declared salience for this rule is 30 (line 58), indicating that this rule will be fired after the rules with higher priority search the database and fail to match.

```
57      (defrule MAIN::Mapping-rule-NoAlternativeConcept
58      (declare (salience 30))
59      (or (functiontype primary)
60         (functiontype secondary))
61      ?func <- (function (verb ?verb) (noun $?noun) (complement $?comp) (matched no))
62      (or  (alternative-concept (name ?name) (function-1 ~?verb ~$?noun))
63      (alternative-concept (name ?name) (function-2 ~?verb ~$?noun)))
64       =>
65      (retract ?func)
66      (assert (bufferfunction (verb1 ?verb) (noun1 ?noun) (complement1 ?comp)))
67      (printout t ?verb " " ?noun " " ?comp " has no alternative concept in the database" crlf))
```

Figure 4.9 A mapping rule when there is no alterative concept in the database

Furthermore, besides notifying the user when there is no alternative concept in the database, the system will also ask to perform concept generation manually for those subfunctions with no alternative concept in the database. A simple GUI shown in Figure 4.10 is built to accept manually generated concepts as input to the system and save in the alternative concepts database for future use.

The final output of the concept generation process is the morphology chart consisting of the subfunctions in the functional structure together with all the available alternative concepts (both generated from the database and by the user). The morphology chart is constructed using wxPython's grid element (Rappin and Dunn, 2006). The morphology chart can display as many alternative concepts as there are in the concept database without limitation. This has been tested with ideal case where there are more than 20 alternative options for single function, to be displayed on the morphology chart. The concepts in the morphology chart (especially those generated by the software) may stimulate/inspire the designer to add some more concepts not included in the morphology chart. In such cases, the designer can add new concepts to the database using the pull down menu in the function library (Figure 4.3) which launches the alternative concepts input window (Figure 4.10). The morphology chart shows the solutions for each subfunction separately. To obtain, the overall solution, those individual solutions need to be combined. This process is treated in the next module.

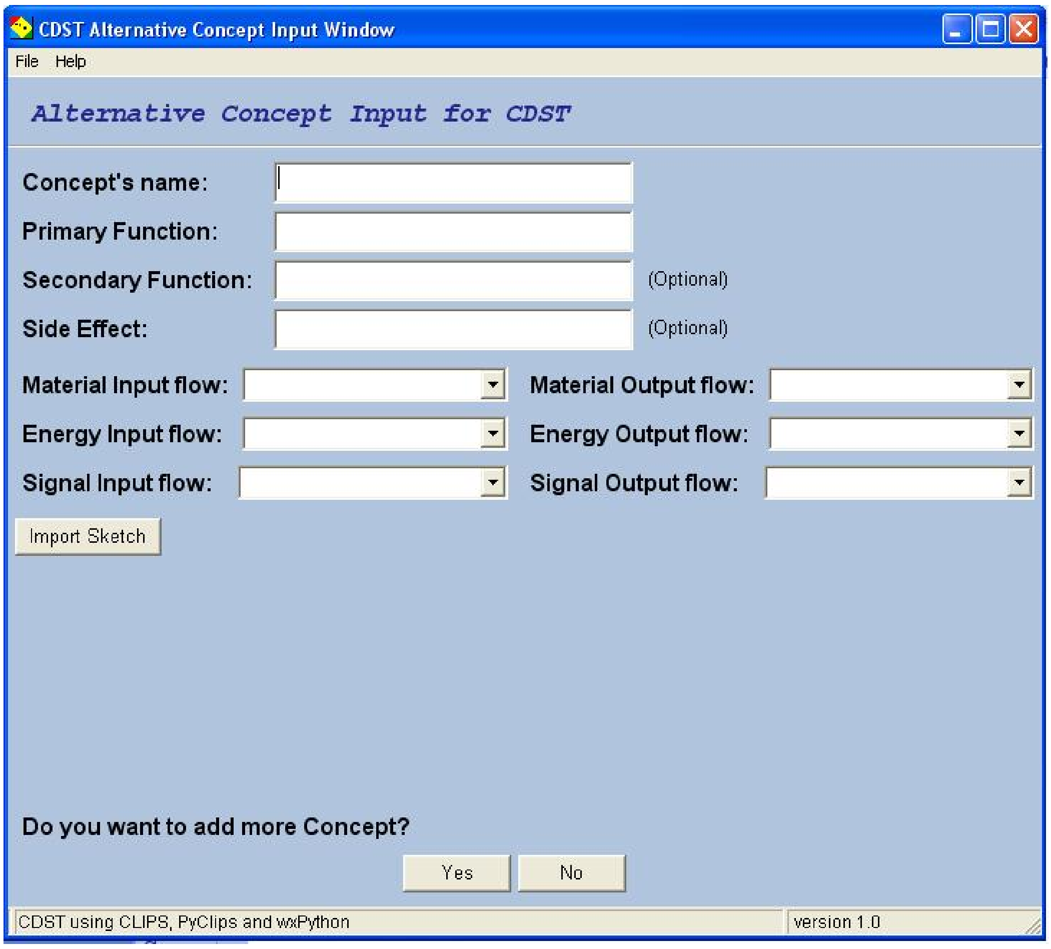


Figure 4.10 Screenshot of alternative concepts input window

4.2.4 Concept Combination Module

The concept combination process takes the morphology chart as an input and gives the combined concept variants as an output to be evaluated in the concept evaluation module. The implementation of the concept combination process is done based on the methodology described in Section 3.2.3. There are two options to be considered in concept combination: to combine all theoretically possible solutions, or to combine flow compatible solutions. In both cases the combined concepts are displayed both textually and schematically showing the components (concepts) that make up the concept variant.

The main objective in the implementation of the concept combination process is to develop generic concept combination rule that can combine concepts regardless of the number of subfunctions in the functional structure. This can be done either by writing all the rules for each case varying the number of subfunctions starting from two to specified number, or to develop a system that can automatically generate the rule depending on the number of subfunctions. The first option have been experimented in the initial phases of the development but the maximum number of subfunctions achieved was ten (Woldemichael and Hashim, 2008). This is because of the complexity in constraining each subfunction and its properties which is error prone while writing the rules manually. To overcome this limitation, the second option has been devised in which a procedural programming approach using deffunction construct is used to directly generate the necessary rule for that particular number of subfunctions. In this case, only the procedures to build the rules are available in the system during the program initiation, i.e., there is no predefined rule at the start of the program. Once the number of subfunction for that particular session is determined from the morphology chart and the concept combination process is invoked, then the system will generate the necessary rule for that particular number of subfunctions. However, this option works only for combining the theoretically possible concept variants. The combination of flow compatible concept variants is done using the first option which is limited to ten subfunctions for one session.

The general rule to combine the theoretically possible concept variants is principally based on the pseudo code in Section 3.2.3, where the concept variants are created by taking one concept at a time for each subfunction in the morphology chart. To compare the general rule with flow compatible concept combination rule, consider the case where there are three subfunctions in the morphology chart. The general rule to combine three subfunctions is shown in Figure 4.11. In this rule, the number of subfunctions (line 69) is the fact that is obtained automatically by the system from the morphology chart. A test constraining element (line 74) is used as part of the pattern matching process in the LHS of the rule to show each subfunction are different and prevent repeated firing of the rule.

```
68      (defrule MAIN::combination1-3
69       (subfunctions1 3)
70      (mainflow Energy)
71      ?f1 <- (morphologyChart (functions $?x1) (alternative-concepts $?y1) (concept ?z1))
72      ?f2 <- (morphologyChart (functions $?x2&~$?x1) (alternative-concepts $?y2) (concept ?z2))
73       ?f3 <- (morphologyChart (functions $?x3&~$?x1&~$?x2) (alternative-concepts $?y3) (concept ?z3))
74      (test (> (fact-index ?f1) (fact-index ?f2) (fact-index ?f3)))
75       =>
76      (bind ?*z* (+ ?*z* 1))
77      (printout t "variant No." ?*z* " : ")
78      (assert (conceptVariant (conceptVariantNumb ?*z*) (conceptVariants ?y1 ?y2 ?y3)))
79      (printout t ?y1 " + " ?y2 " + " ?y3 crlf))
```

Figure 4.11 Theoretically possible concept combination rule for 3 subfunctions

The corresponding flow compatible combination rule is shown in Figure 4.12. In this rule, the basic combination principle is the same as the theoretically possible concept variant rule shown in Figure 4.11. However, this rule has additional test constraining elements (lines 87 and 88) which require the subsequent alternative concepts for the subfunctions in the function structure to have the same input and output. These constraining elements are satisfied if and only if the output flow of the preceding concept is the same as the input flow of the succeeding concept in the morphology chart. Because of this constrain, the applicability of this rule is limited to single flow non-branching functional structure. If there is branching functional structure, it needs to be decomposed manually into single flow non-branching structure before it is given as input to the system. There are three similar concept combination rules in the knowledge-base for each flow type (by varying the value of main flow in line 82) with the same number of subfunctions.

```
80    (defrule MAIN::combination2e-3
81     (subfunctions2 3)
82    (mainflow Energy)
83    ?f1 <- (morphologyChart (functions $?x1) (alternative-concepts $?y1) (concept ?z1))
84    ?f2 <- (morphologyChart (functions $?x2&~$?x1) (alternative-concepts $?y2) (concept ?z2))
85     ?f3 <- (morphologyChart (functions $?x3&~$?x1&~$?x2) (alternative-concepts $?y3) (concept ?z3))
86    (test (> (fact-index ?f1) (fact-index ?f2) (fact-index ?f3)))
87    (test (eq (fact-slot-value ?z1 output-e) (fact-slot-value ?z2 input-e)))
88    (test (eq (fact-slot-value ?z2 output-e) (fact-slot-value ?z3 input-e)))
89     =>
90    (bind ?*y* (+ ?*y* 1))
91    (printout t "variant No." ?*y* " : ")
92    (assert (conceptVariant (conceptVariantNumb ?*y*) (conceptVariants ?y1 ?y2 ?y3)))
93    (printout t ?y1 " + " ?y2 " + " ?y3 crlf))
```

Figure 4.12 Flow compatible concept combination rule for 3 subfunctions

4.2.5 Concept Evaluation Module

The concept evaluation module takes the output of the concept combination phase (i.e., concept variants) and gives ranked concept variants as an output. Three concept evaluation methods (i.e., absolute comparison, concept screening, and weighted decision matrix) are implemented in this module as discussed in Section 3.2.4. The software implementation is to automatically prepare the selection matrix, provide GUI to accept selection criteria from the user as input, and perform simple arithmetic calculation while rating. The concept evaluation module retrieves the concept variants from the knowledge-base, i.e., the actions of the concept combination rules (line 78 or line 92), and selection criteria from the user to create the concept selection matrix. The concept selection matrix is built using wxPython's grid element. Once the user rate the concepts based on their merits, the software can calculate the net score and rank the concepts based on the result to assist the user in decision making.

4.2.6 The GUI Development and its Integration with CLIPS

The graphical user interface is built using wxPython toolkit, to facilitate easy communication between the user and the software by hiding all the programming details. The integration of the knowledge-based system developed in CLIPS with parent code of the GUI development environment (i.e., Python) is done using PyCLIPS module. PyCLIPS embeds full CLIPS functionality in Python applications allowing all the libraries and the knowledge-base to be called and used in Python environment.

wxPython provides standard toolkit to build GUI elements such as: windows, frames, graphical images, grids, menus, dialogs, buttons, texts (both editable and static), list box, choice items, and popup menu. Thus, a user friendly GUI has been built which can accept inputs from the users and display results of the actions being performed. There are more than seventeen GUI windows developed starting from the welcoming window (Figure 4.13) to windows addressing each activities in conceptual design process. Each window has either menus or buttons to control the user's action such as going to the next window, performing specific tasks or quitting the software (Ctrl-Q). In addition, help documentation is provided for users on each window either by using pull down menu "Help" or pressing "F1" key.

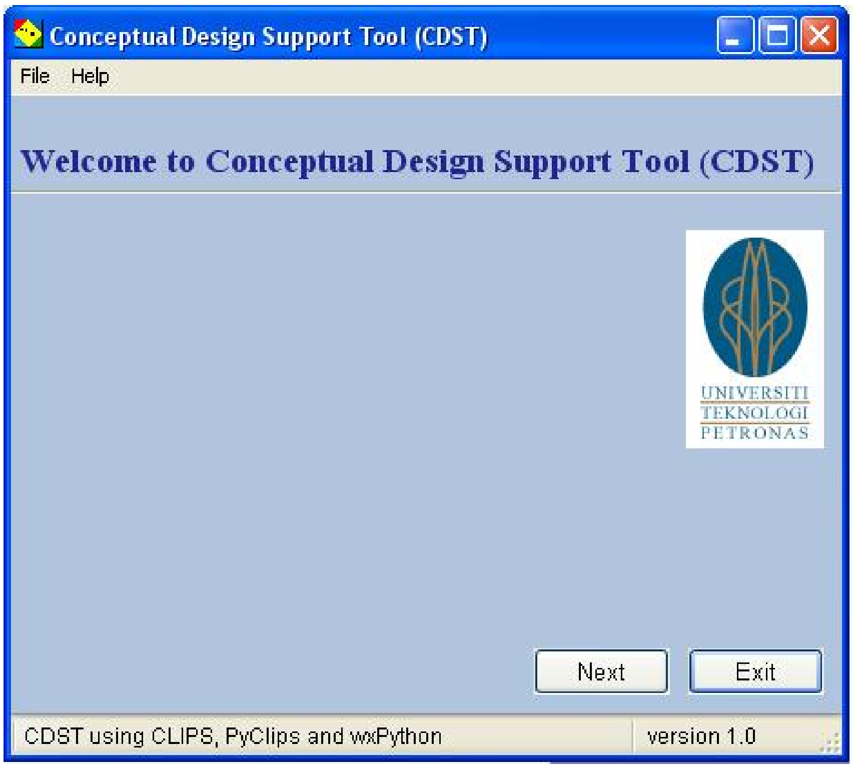

Conceptual Design Support Tool (CDST)
File
Help
Welcome to Conceptual Design Support Tool (CDST)
UNIVERSITI TEKNOLOGI PETRONAS
Next
Exit
CDST using CLIPS, PyClips and wxPython
version 1.0

Figure 4.13 Screenshot of CDST welcoming window

4.3 Integration of the Modules and Initial Testing

After completing the coding process of each module, the next step is to integrate those individual modules to form complete software. However, before integrating the component modules into one, each module should be tested individually. Testing is the process of executing programs with the intention of finding errors and checking whether the program is performing according to intended plan. Accordingly each module has been tested individually to find defects in logic, data, inputs, and outputs. After fixing bugs found during the unit (module) testing process, the modules have been integrated incrementally performing similar tests at each point.

The integrated software is converted into an executable file to make it portable and use the software without installing the programming environments used to build it. As an alternative option, the executable file is converted into windows self installable software where users can install on their machine. A user’s guide is also prepared to familiarize new users and guide on how to install and use CDST. All supporting documents and files are included in the attached CD. The description of these files is presented in Appendix A. The list of software and programming language used to build CDST are presented in Appendix B.

4.4 Summary

In this chapter, the development of CDST was presented as an implementation of the proposed conceptual design process model. The knowledge representation formalisms in CLIPS were first introduced, followed by how the facts and the production rules in each module (i.e., functional modeling, concept generation, concept combination and concept evaluation) were constructed. The conceptual design knowledge in terms of facts such as functions and alternative concepts were represented using template consisting of named slot and attached values. The control knowledge to solve specific problem was represented in terms of production rules. The production rules are composed of an antecedent (the LHS of the rule) consisting of the conditional elements to be met for the rule to be fired and a consequent (the RHS of the rule) consisting of actions to be executed. The production rules are fired based on the existence or non-existence of the specified facts in the working memory using pattern matching and given conditional elements. Accordingly, the production rules for concept generation and concept combination were developed. A knowledge acquisition module to accept new concepts was also developed. The modules were built, tested and integrated to form CDST.

In the next chapter, the features and capabilities of CDST will be demonstrated using case studies. In Chapter 6, the verification and validation tests conducted on CDST by evaluators will be presented.

CHAPTER 5: CASE STUDIES

5.1 Introduction

In this chapter the function-based conceptual design framework proposed in this Chapter 3 is validated by using the conceptual design support tool developed in Chapter 4. The main features and the functionalities of CDST are illustrated with case studies. First a conceptual design of three phase subsea separator design is used as a case study. This case study demonstrates how the designer would interact with the system when engaged in function-based design. The procedures in using the CDST are presented by using screenshots of the GUI together with explanations for each window. As a second case study, a conceptual design support tool for subsea process equipment design (CDSTsped) is introduced. CDSTsped is presented to demonstrate how the knowledge-base of CDST can be customized to specific products and used as design knowledge management system to train novice designers.

5.2 Conceptual Design of Oil and Gas Separator using CDST

This section presents a case study that demonstrates how the developed conceptual design support tool assists designers during the early phase of design. Next the steps in the conceptual design process are demonstrated using the screenshots from the interactions between the designer and the tool.

5.2.1 Problem Description and Functional Modeling

A designer is given a task to design a device that can be used to separate subsea oil well stream which is a mixture of gases and hydrocarbon liquids mixed with water flowing at high velocity into its components. The overall function can be deduced from this customer requirement as: to separate three-phase well fluid into oil, gas, and water for subsea application.

The first step is to translate the customer requirement into functional model manually by the designer. The functional modeling of the product to be designed is derived by decomposing the overall function into a set of subfunctions with the methods described in Section 3.2.1. The subfunction may be classified as main subfunctions and auxiliary subfunctions. Main subfunctions are those subfunctions that directly contribute to the overall function while auxiliary subfunctions are supporting subfunctions which contribute to the overall function indirectly (Pahl and Beitz, 1996). A thorough study of the principles through which the bond between the flowing fluids can be weakened and separated into its components together with the study of existing product results in the functional structure shown in Figure 5.1. Note that auxiliary subfunctions are not included in the functional modeling and material flow is considered as the primary/main flow in this particular design.

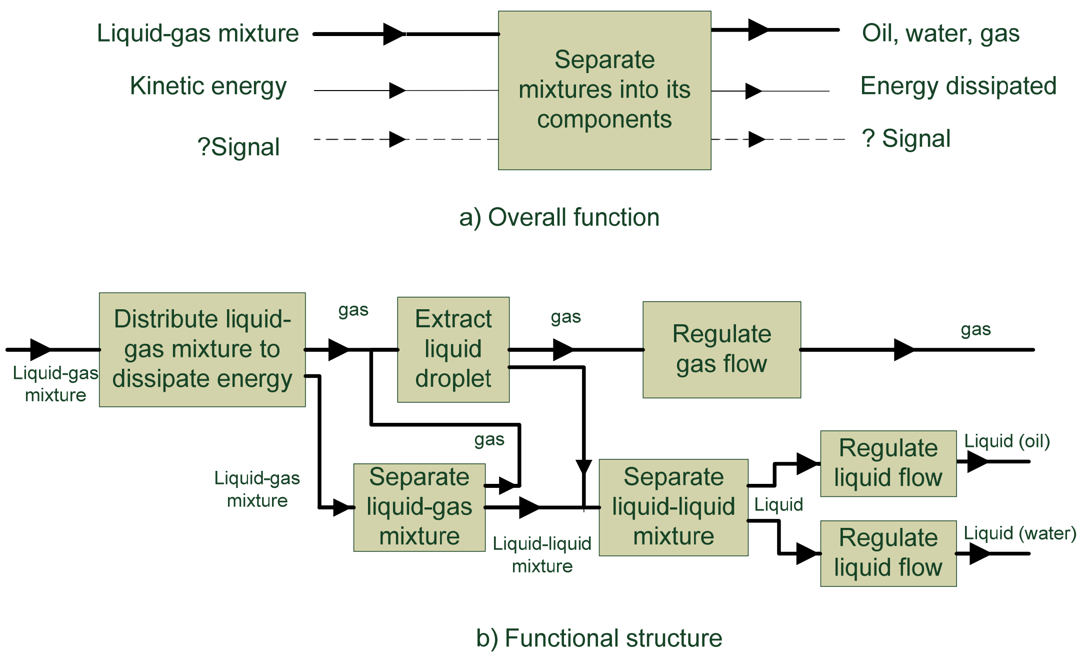


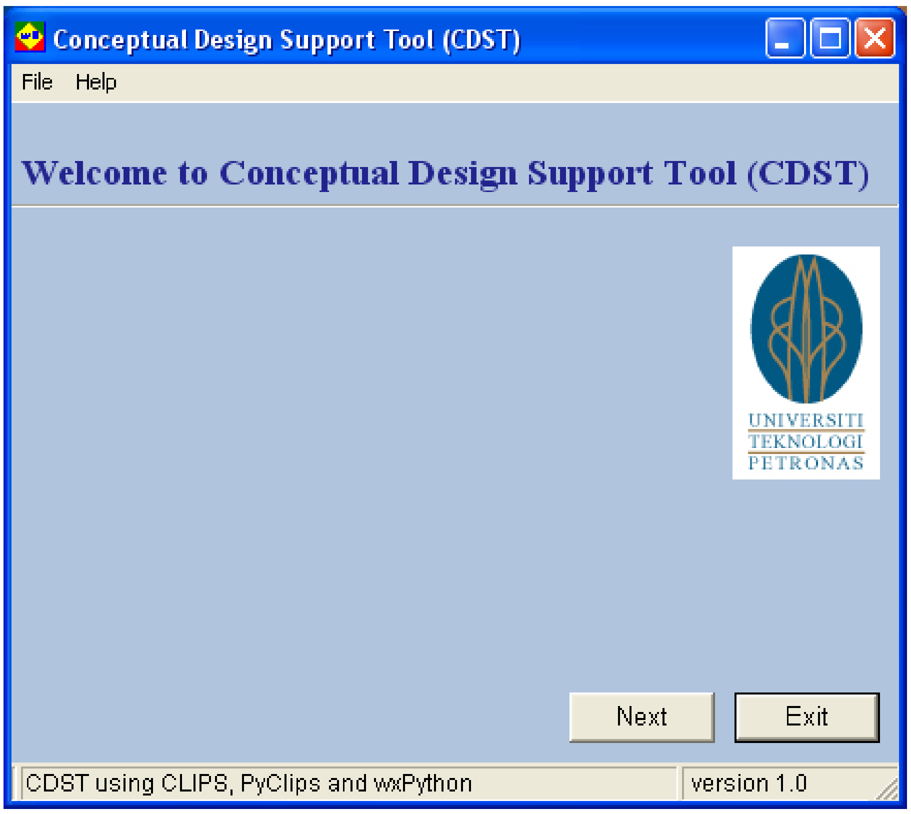
Figure 5.1 Functional model for three-phase oil and gas separator

The next step is to give those subfunctions as input to the CDST. The design session with CDST can be initiated using two approaches:

i. Double click on the CDST setup file in the executable folder of the software; or

ii. Go to start -> Programs -> Conceptual Design Support -> CDST, if the software is installed on the machine.

Both options bring the welcoming window shown in Figure 5.2.


Conceptual Design Support Tool (CDST)
File
Help
UNIVERSITI TEKNOLOGI PETRONAS
Next
Exit

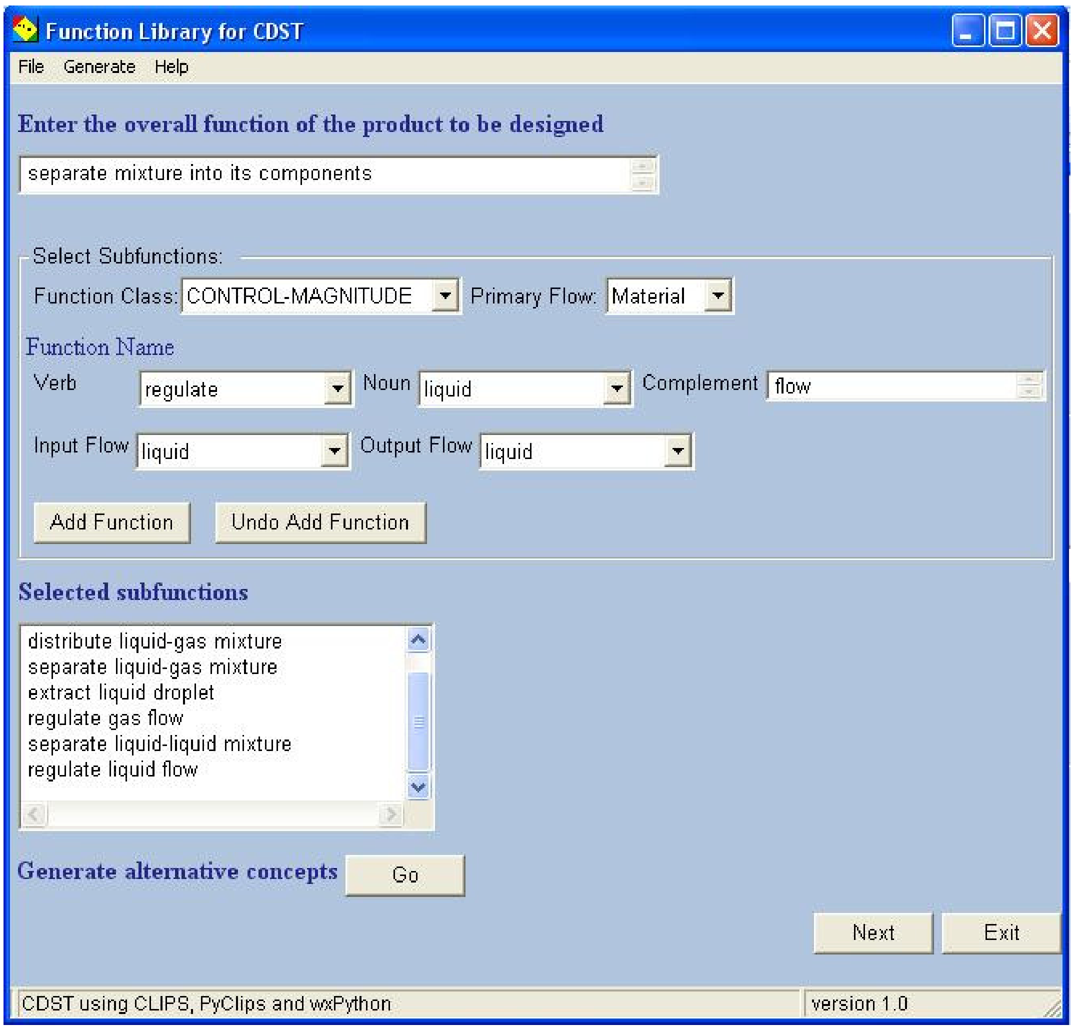
Welcome to Conceptual Design Support Tool (CDST)

CDST using CLIPS, PyClips and wxPython

version 1.0

Figure 5.2 CDST welcoming window

The “Next” button on this window brings the function library window (Figure 5.3), where the subfunctions in the functional structure are given as input to the system. In the function library, the user first types the overall functions on the space provided, and then give each subfunction as input to the system. Note that the subfunctions in the function structure are described in terms of the functions in the function library.



Enter the overall function of the product to be designed

separate mixture into its components

Select Subfunctions:

Function Class: CONTROL-MAGNITUDE

Primary Flow: Material

Function Name

Verb: regulate

Noun: liquid

Complement: flow

Input Flow: liquid

Output Flow: liquid

Selected subfunctions

distribute liquid-gas mixture
separate liquid-gas mixture
extract liquid droplet
regulate gas flow
separate liquid-liquid mixture
regulate liquid flow

CDST using CLIPS, PyClips and wxPython

version 1.0

Figure 5.3 Screenshot of the function library

There are two options for the user to give subfunction in the functional structure as input to CDST: load from text file or select from the function library.

To load from text file:

i. Prepare a text file with all the subfunctions following CLIPS input format as defined in function template (Figure 4.2) and save as a text file (i.e., write the following entry in any text editor and save as “filename.txt”).

(function (verb distribute) (noun liquid-gas mixture))

(function (verb separate) (noun liquid-gas mixture))

(function (verb extract) (noun liquid) (complement droplet))

(function (verb regulate) (noun gas) (complement flow))

(function (verb separate) (noun liquid-liquid mixture))

(function (verb regulate) (noun liquid) (complement flow))

ii. Use the “File” pull down menu on the function library window shown in Figure 5.4 and select “Load”, or press Ctrl+L key to load the saved text file.

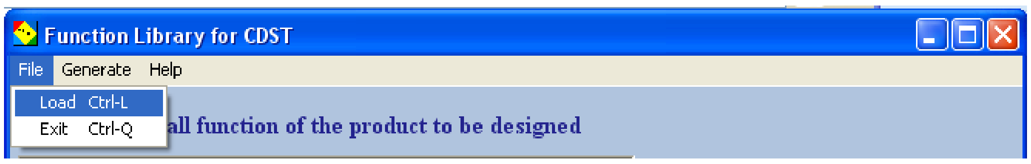

Function Library for CDST
File
Generate
Help
Load Ctrl-L
Exit Ctrl-Q
all function of the product to be designed

Figure 5.4 Screen shot of pull down menu to load input functional structure

iii. The loaded subfunctions will be displayed on the “Selected subfunctions” text window in the function library window (Figure 5.3).

To select the subfunctions from the function library the user goes through the following steps:

1. Use the functional class choice item and select from the eight primary or main classes of functions that corresponds to your subfunction. This will populate the “Verb” choice item with the secondary and tertiary functions in the Function Name. The user can view the help documents by pressing “F1” key to know which action verbs are under a given primary class.
2. Use the primary flow choice item and select the flow type corresponding to your subfunction. This will populate the “Noun”, “Input Flow”, and “Output Flow” choice items with the secondary and tertiary flows corresponding to the selected flow.
3. Select the verb and noun choices. Add textually the complement if the function cannot be described by verb + noun. Select the corresponding input and output flows. Note that the complement, the input flow and

output flow are optional, i.e., in cases where there is no input and output flows and the function can be described by only verb + noun, these attributes can be omitted.

4. Press the "Add Function" button to add the subfunction into the working memory. The added subfunction will be displayed in the "Selected subfunction" text field. You can remove the added function from the working memory using the "Undo Add Function" button one at a time.
5. Repeat steps 1-4 until all the subfunctions in the functional structure are added. Note that fact duplication is not enabled in the CLIPS, thus when there are duplicate subfunctions in the functional structure only one subfunction should be given.

5.2.2 Concept Generation and Combination

The "Add Function" button in the function library window (Figure 5.3) and the "Load" in the "File" pull down menu (Figure 5.4) instruct the software to add (assert) those subfunctions into the current working memory of the knowledge-based system. Once all the subfunctions are added to working memory, the concept generation process is initiated by using the "Go" button in the function library window next to "Generate alternative concepts" text. This brings, a pop up window shown in Figure 5.5 for the user to select the type of function to be considered for concept generation. If the user selects the "Both Primary and Secondary Functions" option, the system will search for concepts whose primary and secondary function match with the subfunctions in the function structure. On the other hand, if "Only Primary Functions" is selected only primary functions are considered in searching for the alternative concepts.

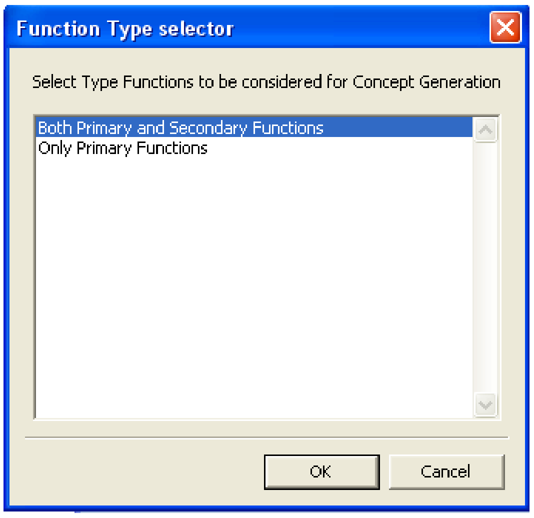
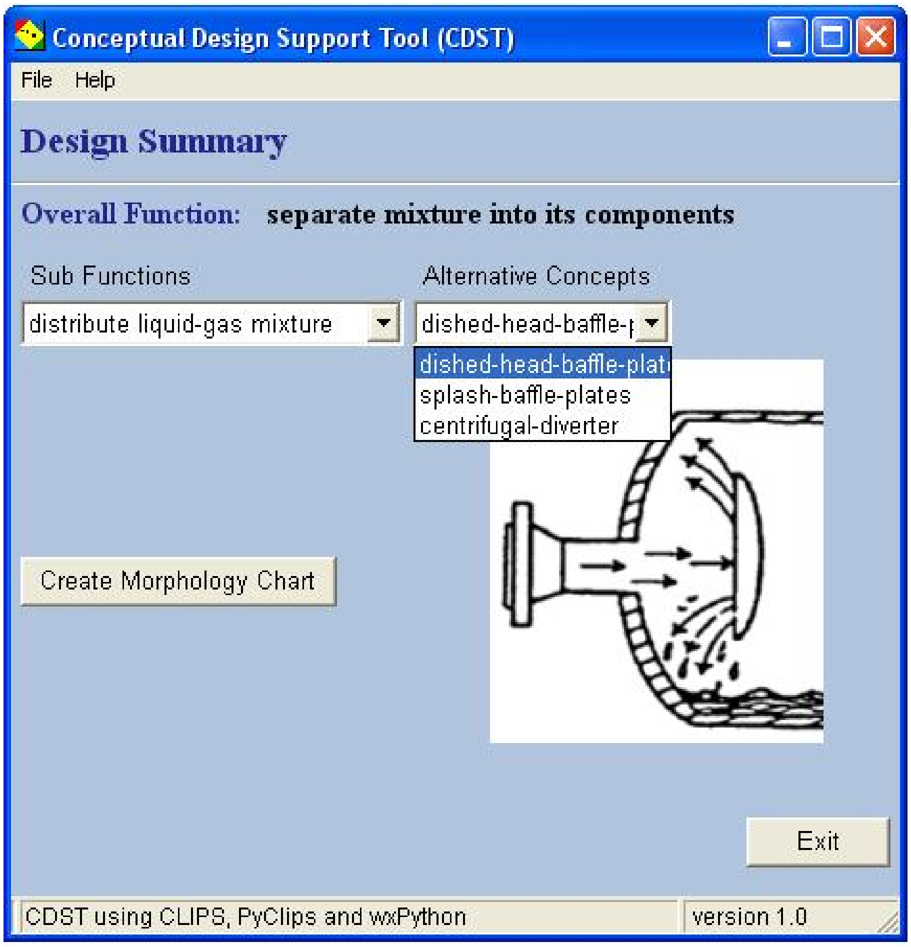

Function Type selector
Select Type Functions to be considered for Concept Generation
Both Primary and Secondary Functions
Only Primary Functions
OK
Cancel

Figure 5.5 Function type selector for concept generation

In both cases, if there are alternative concepts in the database for each subfunction, a text message which states “concepts are generated for all subfunctions successfully” will be displayed. If one or more subfunctions have no alternative concept, then message windows will popup to generate concepts manually. Furthermore, if some of the generated concepts have side effects, the system will suggest to the user for possible consideration of the side effects as new requirement. This effect demonstrates the evolving nature of requirements during conceptual design process and how CDST integrate this effect.

In this particular case all the subfunctions have alternative concepts from the database with no side effects, thus there is no need to generate concepts manually or consider the side effects as new requirement. In addition to this, a new concept can be added to the database by using the “Generate” pull down menu in the function library window which brings an alternative concept input window (shown in Figure 4.10) for manual concept generation.

The “Next” button on the function library window brings design summary window (Figure 5.6) where the designer can review all the subfunctions and their respective alternative concepts generated individually. For each subfunctions in the functional structure given as input to the system, the user can view the respective alternative concepts in the database both textually and schematically in this window. The subfunctions and all the alternative concepts can be viewed on the morphology chart using the “Create Morphology Chart” button on this window. The morphology chart for this design is shown in Figure 5.7.



Design Summary

Overall Function: separate mixture into its components

Sub Functions: distribute liquid-gas mixture

Alternative Concepts: dished-head-baffle-plates, splash-baffle-plates, centrifugal-diverter

CDST using CLIPS, PyClips and wxPython | version 1.0

Figure 5.6 Design summary window

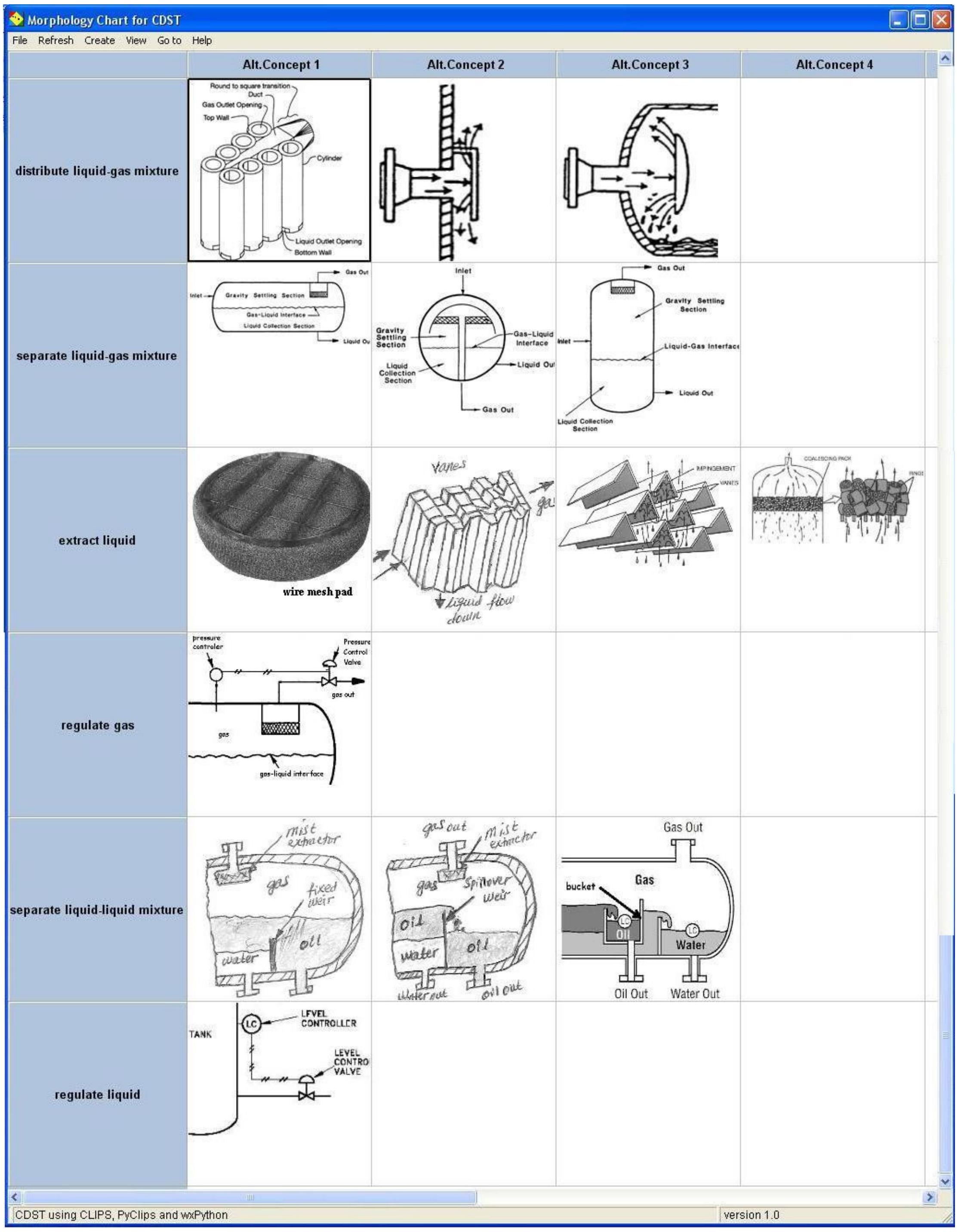
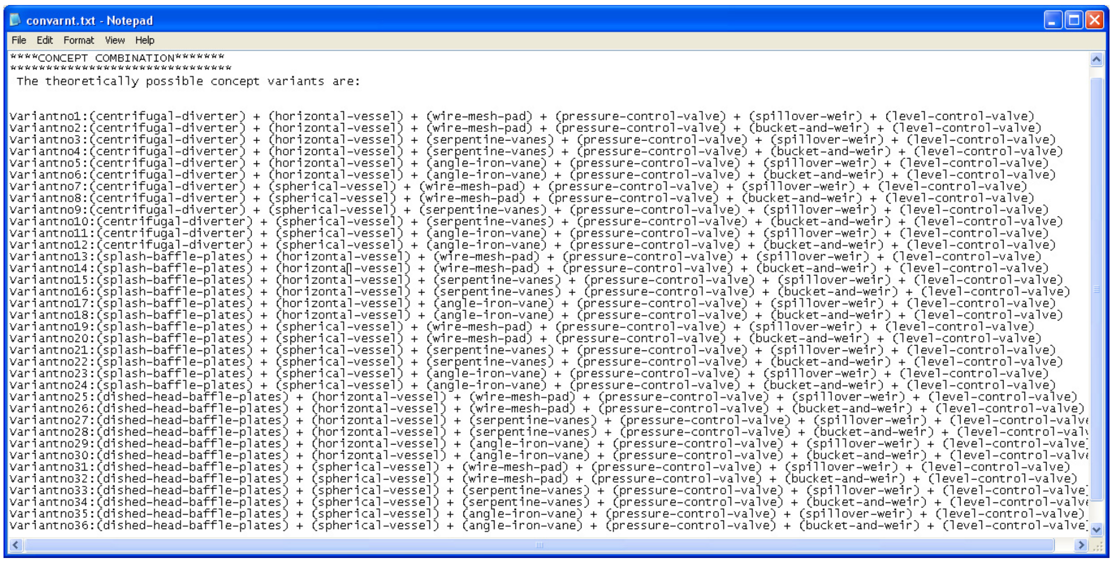


Figure 5.7 Morphology chart

In the morphology chart the user examines the generated concepts and can reject some of the infeasible concepts based on experience. As the number of concepts in the database increases the number of alternative concepts generated by the tool also increases and the user needs to decide which of the concepts have to be rejected. This will be imminent especially when the concepts in the database are extracted from existing products and used to design other products. The user of CDST has an option to reject infeasible concepts right on the morphology chart. This helps to reduce the combinatorial explosion later in the concept combination process. When the user clicks on any of the concepts in the morphology chart, the concepts name is displayed textually. The user deletes the text if the concept is considered to be infeasible and use the “Refresh” pull down menu to remove the rejected concept’s sketch and redraw the morphology chart.

The concept combination process is initiated by using "create" pull down menu on the morphology chart. The user has two options, either to create all theoretically possible concept variants or flow compatible concept variants. However, the flow compatible concept variant combination works only for non-branching single flow functional structure; thus this option is not applicable for this particular case study. Combining the concepts in the morphology chart results in 108 theoretically possible concept variants obtained from the concept combination process. Based on the customer’s requirement and feasibility of the concepts to be used for subsea applications the following concepts are removed from the morphology chart: vertical-vessel from separate liquid-gas mixture; coalescence-pack from extract liquid droplet; and fixed-weir from separate liquid-liquid mixture. After refreshing the morphology chart, the remaining concepts are combined resulting in a total of 36 theoretically possible concept variants.

The concept variants are displayed textually indicating the name of all the concepts (components) in that particular concept variant as shown in Figure 5.8. In addition, the concepts (components) in each concept variant can be viewed schematically by using “view” pull down menu on the morphology chart window. Figure 5.9 shows the concepts (components) of concept variant number 1 schematically.



```
****CONCEPT COMBINATION*******
*********************************
 The theoretically possible concept variants are:

Variantno1:(centrifugal-diverter) + (horizontal-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno2:(centrifugal-diverter) + (horizontal-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno3:(centrifugal-diverter) + (horizontal-vessel) + (serpentine-vanes) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno4:(centrifugal-diverter) + (horizontal-vessel) + (serpentine-vanes) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno5:(centrifugal-diverter) + (horizontal-vessel) + (angle-iron-vane) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno6:(centrifugal-diverter) + (horizontal-vessel) + (angle-iron-vane) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno7:(centrifugal-diverter) + (spherical-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno8:(centrifugal-diverter) + (spherical-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno9:(centrifugal-diverter) + (spherical-vessel) + (serpentine-vanes) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno10:(centrifugal-diverter) + (spherical-vessel) + (serpentine-vanes) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno11:(centrifugal-diverter) + (spherical-vessel) + (angle-iron-vane) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno12:(centrifugal-diverter) + (spherical-vessel) + (angle-iron-vane) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno13:(splash-baffle-plates) + (horizontal-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno14:(splash-baffle-plates) + (horizontal-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno15:(splash-baffle-plates) + (horizontal-vessel) + (serpentine-vanes) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno16:(splash-baffle-plates) + (horizontal-vessel) + (serpentine-vanes) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno17:(splash-baffle-plates) + (horizontal-vessel) + (angle-iron-vane) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno18:(splash-baffle-plates) + (horizontal-vessel) + (angle-iron-vane) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno19:(splash-baffle-plates) + (spherical-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno20:(splash-baffle-plates) + (spherical-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno21:(splash-baffle-plates) + (spherical-vessel) + (serpentine-vanes) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno22:(splash-baffle-plates) + (spherical-vessel) + (serpentine-vanes) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno23:(splash-baffle-plates) + (spherical-vessel) + (angle-iron-vane) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno24:(splash-baffle-plates) + (spherical-vessel) + (angle-iron-vane) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno25:(dished-head-baffle-plates) + (horizontal-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno26:(dished-head-baffle-plates) + (horizontal-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno27:(dished-head-baffle-plates) + (horizontal-vessel) + (serpentine-vanes) + (pressure-control-valve) + (spillover-weir) + (level-control-valve
Variantno28:(dished-head-baffle-plates) + (horizontal-vessel) + (serpentine-vanes) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valv
Variantno29:(dished-head-baffle-plates) + (horizontal-vessel) + (angle-iron-vane) + (pressure-control-valve) + (spillover-weir) + (level-control-valve
Variantno30:(dished-head-baffle-plates) + (horizontal-vessel) + (angle-iron-vane) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve
Variantno31:(dished-head-baffle-plates) + (spherical-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno32:(dished-head-baffle-plates) + (spherical-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve)
Variantno33:(dished-head-baffle-plates) + (spherical-vessel) + (serpentine-vanes) + (pressure-control-valve) + (spillover-weir) + (level-control-valve
Variantno34:(dished-head-baffle-plates) + (spherical-vessel) + (serpentine-vanes) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve
Variantno35:(dished-head-baffle-plates) + (spherical-vessel) + (angle-iron-vane) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)
Variantno36:(dished-head-baffle-plates) + (spherical-vessel) + (angle-iron-vane) + (pressure-control-valve) + (bucket-and-weir) + (level-control-valve
```

Figure 5.8 Concept variants represented textually

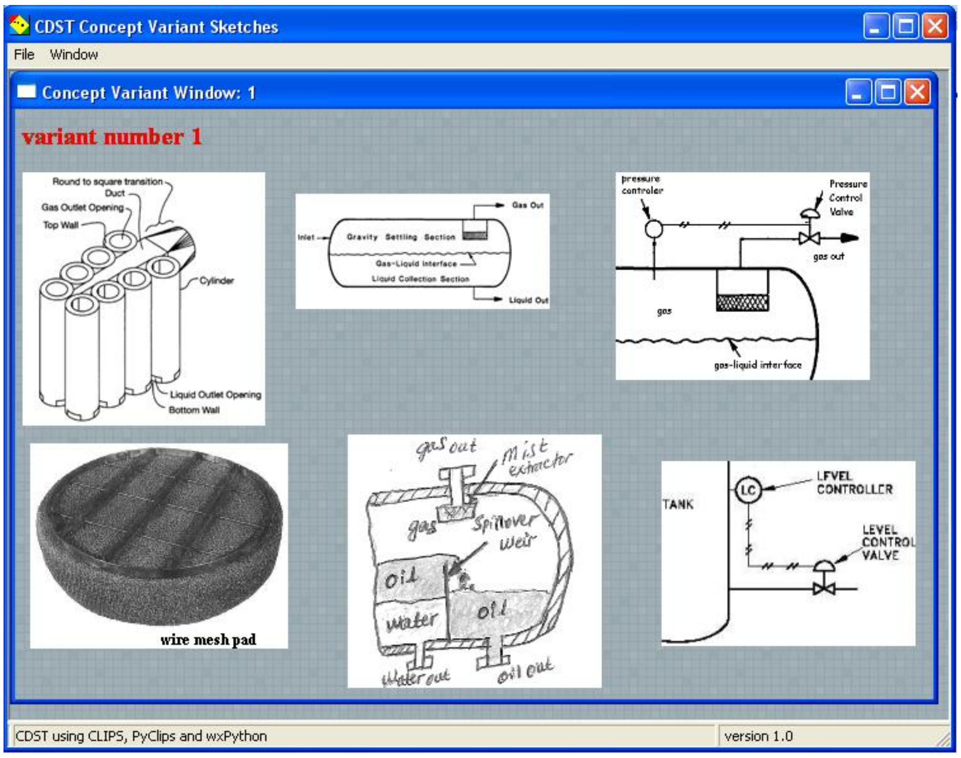


Figure 5.9 Schematic representation of the concepts

5.2.3 Concept Evaluation

Once the concept variants are created, the user uses the “Go to” pull down menu in the morphology chart window (Figure 5.7) and selects the concept evaluation option. The main window of the concept evaluation process is shown in Figure 5.10.

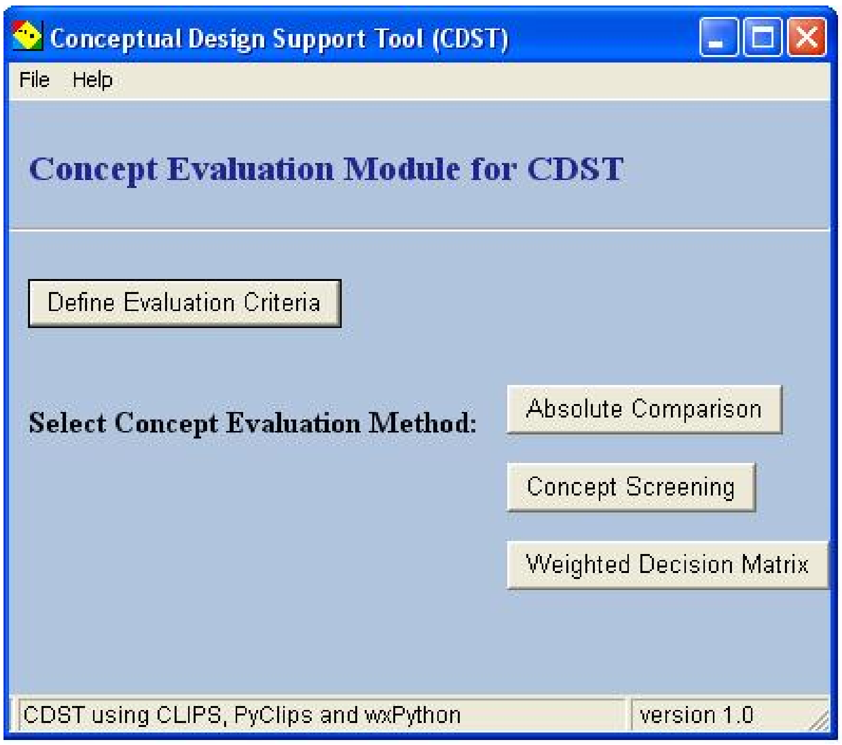
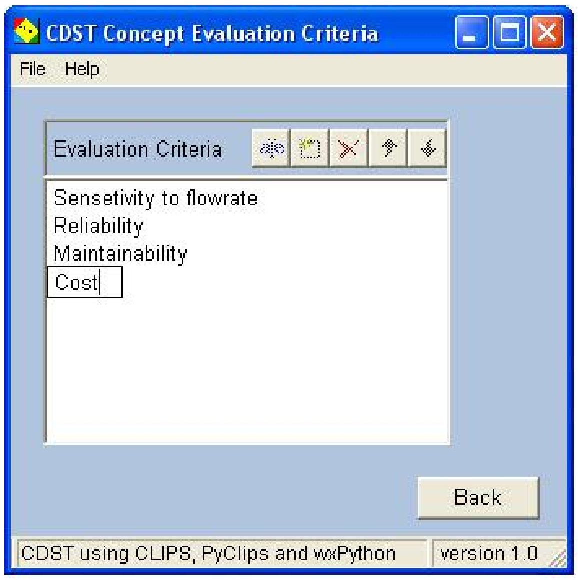

Conceptual Design Support Tool (CDST)
File
Help
Concept Evaluation Module for CDST
Define Evaluation Criteria
Select Concept Evaluation Method:
Absolute Comparison
Concept Screening
Weighted Decision Matrix
CDST using CLIPS, PyClips and wxPython
version 1.0

Figure 5.10 Concept selection process main window

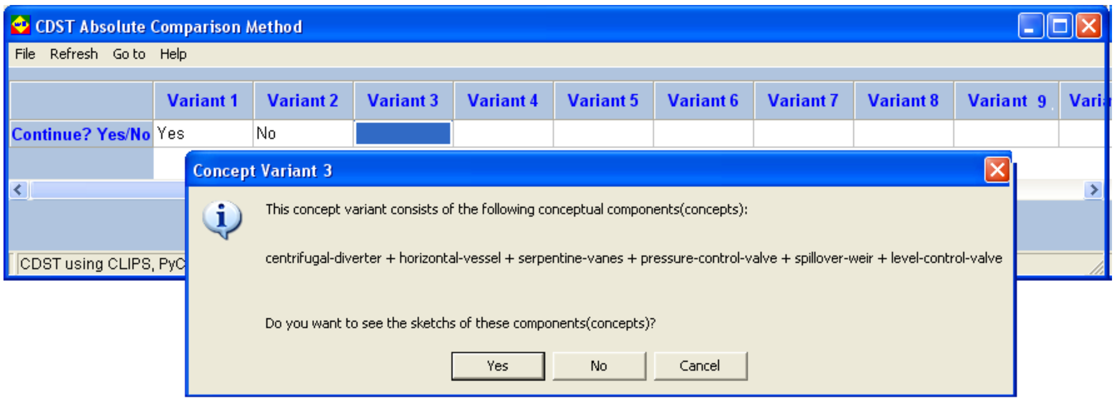
As stated in the Section 3.2.4, the concept evaluation process starts by identifying evaluation criteria. The user can select predefined evaluation criteria (default values) or give new evaluation criteria using criteria input window. Figure 5.11 shows the window that accepts criteria from the user.


CDST Concept Evaluation Criteria
File
Help
Evaluation Criteria
Back
CDST using CLIPS, PyClips and wxPython
version 1.0

Sensetivity to flowrate
Reliability
Maintainability
Cost

Figure 5.11 Evaluation criteria input window

The first evaluation process is using absolute comparison method. In this method concept variants are evaluated based on go/no-go screening of customer requirement, judgment of feasibility of the design, and assessment of technological readiness using absolute comparison window. From the customer requirement the separator is to be used for subsea application in a three phase flow. This impose a constraint that those concept variants with spherical vessel to be excluded because of its high cost of manufacturing and less efficiency to separate three phase flow. Similarly, those concept variants with bucket-and-weir as a means to separate liquid-liquid mixture are considered to be rejected. Accordingly those concept variants with spherical vessels and bucket-and-weir will be eliminated at this stage. The user decides whether to continue or reject each concept variant by choosing either yes or no on the absolute comparison window. The components of each concept variants either textually or schematically can be viewed by double clicking on the respective concept variant column in the absolute comparison window. For example the components of concept variant number three are displayed as shown in Figure 5.12. Once the decision is done for all the concept variants, then the user uses "Refresh" pull down menu to eliminate those rejected concept variants. This reduces the total number concept variant to 9.



	Variant 1	Variant 2	Variant 3	Variant 4	Variant 5	Variant 6	Variant 7	Variant 8	Variant 9	Vari
Continue? Yes/No	Yes	No								

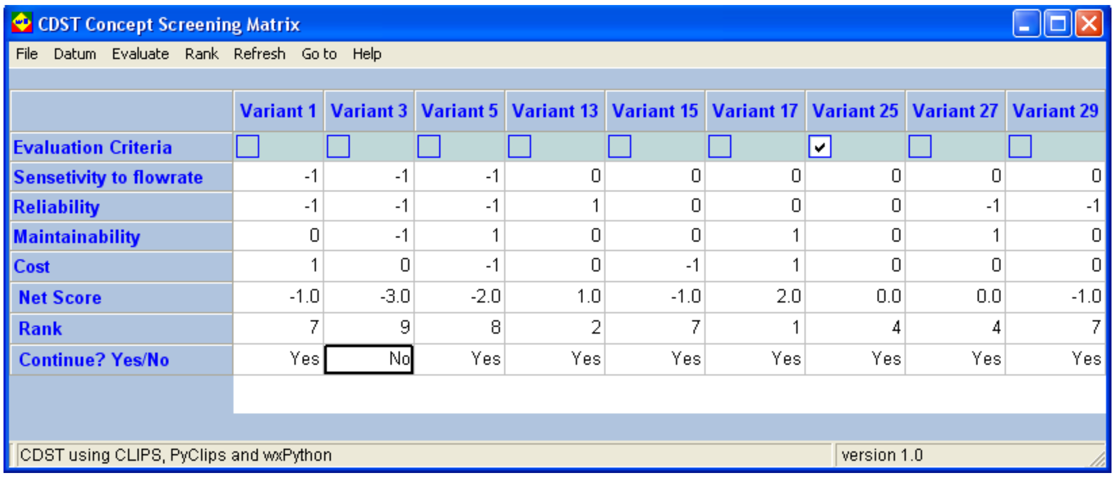
This concept variant consists of the following conceptual components(concepts):

centrifugal-diverter + horizontal-vessel + serpentine-vanes + pressure-control-valve + spillover-weir + level-control-valve

Do you want to see the sketchs of these components(concepts)?

Figure 5.12 Absolute comparison window with list of concepts for concept variant 3

From the absolute comparison window (Figure 5.12), using the "Go to" pull down menu, the user selects either concept screening method or weighted decision matrix method to evaluate the remaining concept variants. First let's consider the concept screening method to evaluate the concept variants. The concept screening window has pull down menu to select one of the concept variants or add a new concept variant as a datum. A datum concept variant is the one considered to be the best among the concept variants or a competitive products concept variant reduced to the same level of abstraction. Here, Concept variant 25 is considered as a datum concept from the concept variants. The user rates each concept variant as: 1 if it is better than the datum concept, 0 if it is the same as the datum concept and -1 if it is worse than the datum concept for each criterion. This is an iterative process and after each evaluation those concept variants with poor performance may be eliminated. For brevity, only the final concept screening window is shown in Figure 5.13. The concept variants with "No" value for the last row of the concept screening matrix are removed using the "Refresh" pull down menu.

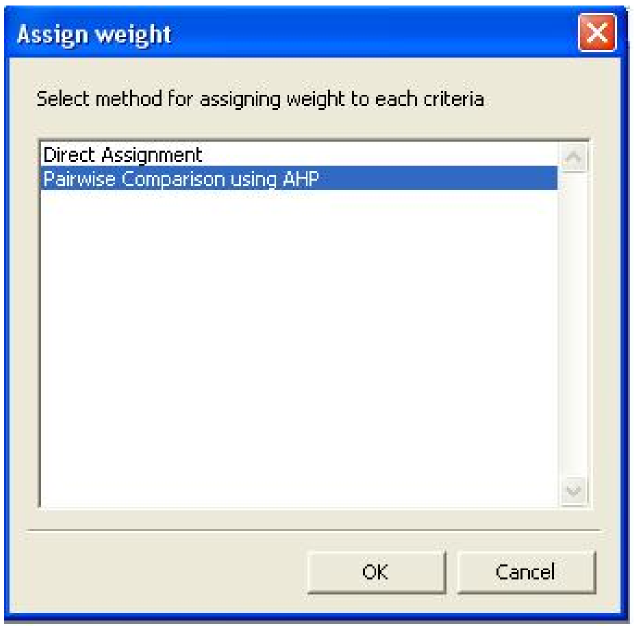

CDST Concept Screening Matrix
File Datum Evaluate Rank Refresh Go to Help

	Variant 1	Variant 3	Variant 5	Variant 13	Variant 15	Variant 17	Variant 25	Variant 27	Variant 29
Evaluation Criteria	☐	☐	☐	☐	☐	☐	✓	☐	☐
Sensetivity to flowrate	-1	-1	-1	0	0	0	0	0	0
Reliability	-1	-1	-1	1	0	0	0	-1	-1
Maintainability	0	-1	1	0	0	1	0	1	0
Cost	1	0	-1	0	-1	1	0	0	0
Net Score	-1.0	-3.0	-2.0	1.0	-1.0	2.0	0.0	0.0	-1.0
Rank	7	9	8	2	7	1	4	4	7
Continue? Yes/No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

CDST using CLIPS, PyClips and wxPython
version 1.0

Figure 5.13 Screenshot of concept screening window

The remaining concept variants are finally evaluated by weighted decision matrix method. The weighted decision matrix window is obtained by using the "Go to" pull down menu on the concepts screening window. This brings the popup window for selecting weight assignment method for each criterion (Figure 5.14). The direct assignment method is selected when the designer assign weight based on his/her previous experience. Here, a pairwise comparison matrix is used to assign the weight as shown in Figure 5.15. In pairwise comparison, each criterion is compared with all the criteria and rated using the guide line discussed in Section 3.2.4.

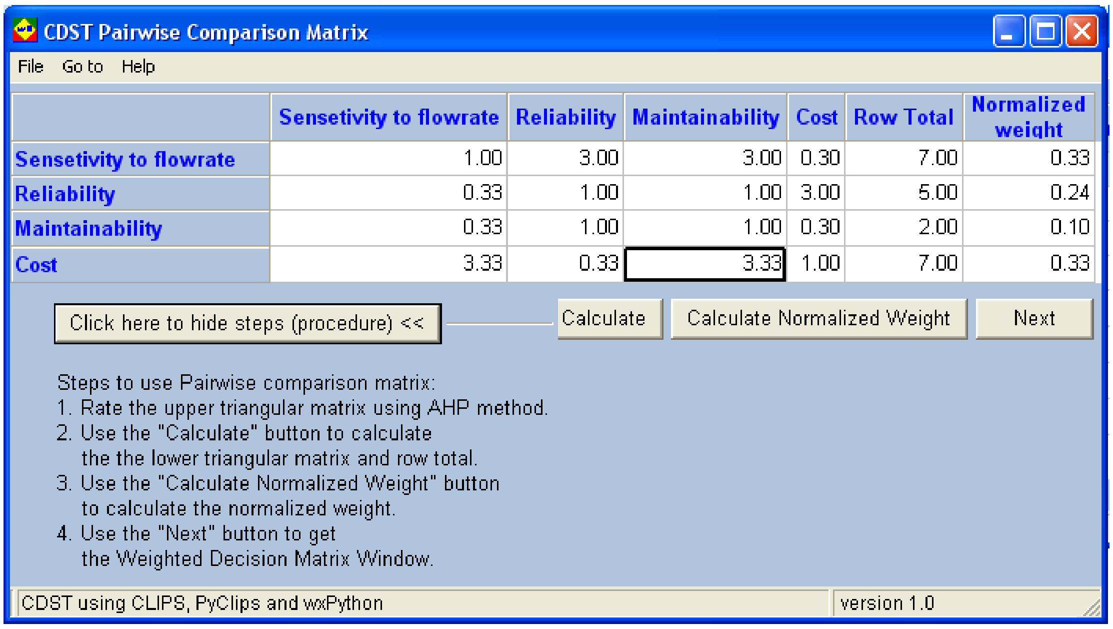


Assign weight

Select method for assigning weight to each criteria

Direct Assignment
Pairwise Comparison using AHP

Figure 5.14 Weight assigning method selection popup window



CDST Pairwise Comparison Matrix

	Sensetivity to flowrate	Reliability	Maintainability	Cost	Row Total	Normalized weight
Sensetivity to flowrate	1.00	3.00	3.00	0.30	7.00	0.33
Reliability	0.33	1.00	1.00	3.00	5.00	0.24
Maintainability	0.33	1.00	1.00	0.30	2.00	0.10
Cost	3.33	0.33	3.33	1.00	7.00	0.33

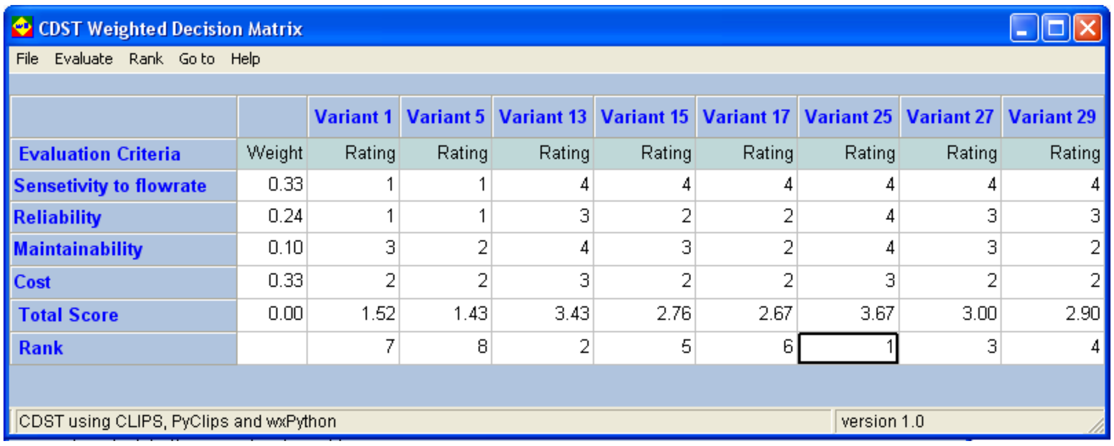
Click here to hide steps (procedure) <<
Calculate
Calculate Normalized Weight
Next

Steps to use Pairwise comparison matrix:
1. Rate the upper triangular matrix using AHP method.
2. Use the "Calculate" button to calculate the the lower triangular matrix and row total.
3. Use the "Calculate Normalized Weight" button to calculate the normalized weight.
4. Use the "Next" button to get the Weighted Decision Matrix Window.

CDST using CLIPS, PyClips and wxPython version 1.0

Figure 5.15 Pairwise comparison matrix

Once the user rates each criterion in the upper triangular matrix, the row total and the normalized weights are calculated using the “Calculate” and “Calculate Normalized weight” buttons respectively. The normalized weight is the relative weight of each criterion. Then, the user uses the “Next” button to go to the weighted decision matrix and rate each concept variant using a 5-point scale. The total score is calculated and the concept variants ranked using the "Evaluate" and "Rank" pull down menus respectively (Figure 5.16). According to the evaluation result concept variant number 25 ((dished-head-baffle-plates) + (horizontal-vessel) + (wire-mesh-pad) + (pressure-control-valve) + (spillover-weir) + (level-control-valve)) is selected for further development whose schematic view is shown in Figure 5.17.



		Variant 1	Variant 5	Variant 13	Variant 15	Variant 17	Variant 25	Variant 27	Variant 29
Evaluation Criteria	Weight	Rating	Rating	Rating	Rating	Rating	Rating	Rating	Rating
Sensetivity to flowrate	0.33	1	1	4	4	4	4	4	4
Reliability	0.24	1	1	3	2	2	4	3	3
Maintainability	0.10	3	2	4	3	2	4	3	2
Cost	0.33	2	2	3	2	2	3	2	2
Total Score	0.00	1.52	1.43	3.43	2.76	2.67	3.67	3.00	2.90
Rank		7	8	2	5	6	1	3	4

CDST using CLIPS, PyClips and wxPython | version 1.0

Figure 5.16 Final concept evaluation using weighted decision matrix

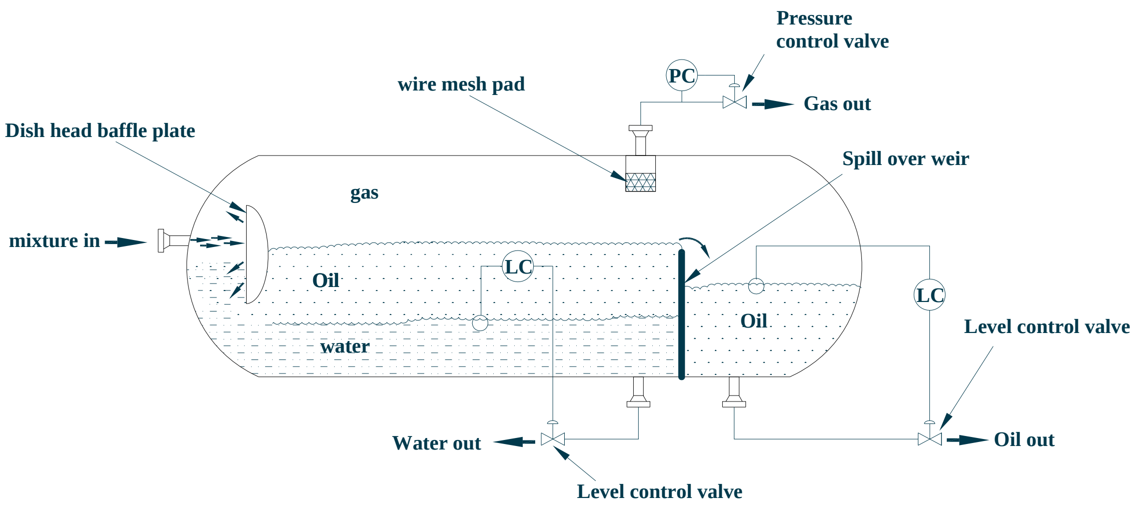


Figure 5.17 Schematic view of the selected concept variant

The system automatically creates a text file as report while the user is performing the design process. The file contains date and time at which the design is conducted, the overall function, the subfunctions, the concepts generated, the concepts rejected by the user on morphology chart, the combined concept variants, and the selected concept variant. This helps to preserve the design history for future reference.

It can be summarized from this case study that, the conceptual design support tool developed based on the proposed model assists designer by:

i. Supplementing designer's knowledge with the generated concepts from the knowledge base, and

ii. Handling the repetitive and time consuming tasks such as constructing the morphology chart, combining concepts, and creating concept evaluation matrices.

5.3 CDST for Subsea Process Equipment Design

In this section, the knowledge-base developed for CDST is used to build a conceptual design support tool specifically for subsea process equipments to demonstrate component selection for existing products. The main objective is to demonstrate how design knowledge of existing products can systematically be represented and saved in the computer system as knowledge management system for future use or training novice designers.

The conceptual design support tool for subsea process equipment design (CDSTsped) is developed using the knowledge-base for CDST by modifying the GUI. The main difference between CDST and CDSTsped lays on the functional modeling. In CDSTsped, instead of selecting the subfunctions from the function library, the user will select the overall function from the given choices (i.e., the functional modeling is built in). Once the user selects the overall function, the system will populate the subfunctions for that particular choice. This is followed by generating alternative concepts for each subfunction from the alternative concepts database. The flow chart for CDSTsped is shown in Figure 5.18.

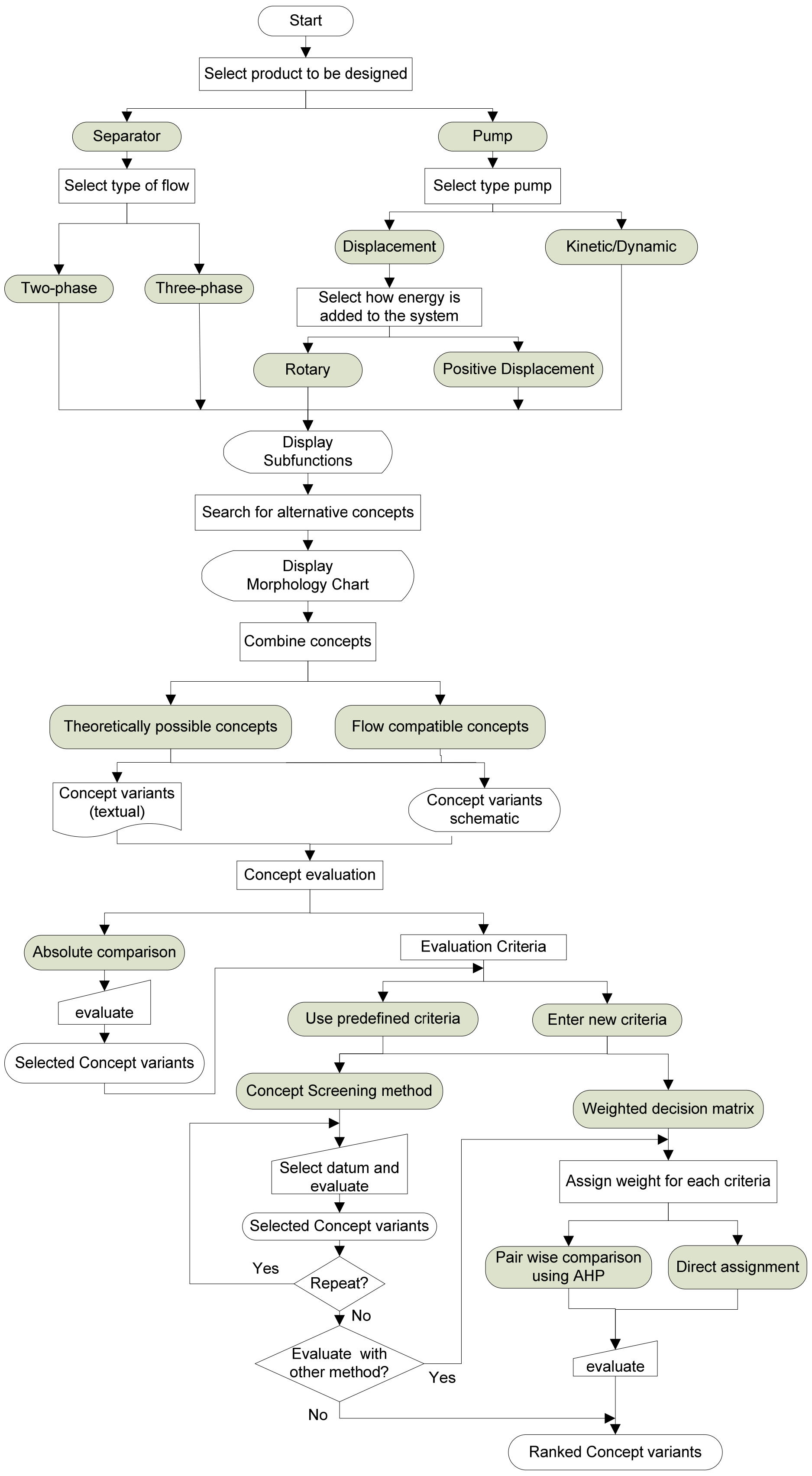


Figure 5.18 Flow chart for CDSTsped

As can be seen from this flow chart, the concept generation, concept combination and concept evaluation processes are identical with the CDST flowchart shown in Figure 3.6. Through a user friendly GUI developed, the user can easily explore the design information and perform conceptual design using push buttons, choice items, and menu bars. Next, the design process using CDSTsped is demonstrated with screenshots of GUI.

The design session with CDSTsped can be initiated using two approaches:

i. Double click on the CDSTsped setup file in the executable folder of the software; or

ii. Go to start -> Programs -> Conceptual Design Support -> CDSTsped, if the software is installed on the machine.

Both options bring the welcoming window shown in Figure 5.19.

Next, the designer selects the type of product to be designed (Separator or pump) in the initial window of CDSTsped. Depending on the selection, specific product design window through which the remaining design process continues in the form of question and answer will be displayed. Selecting the pump option brings the pump design window (Figure 5.20), where the user selects the type of pump to be designed.

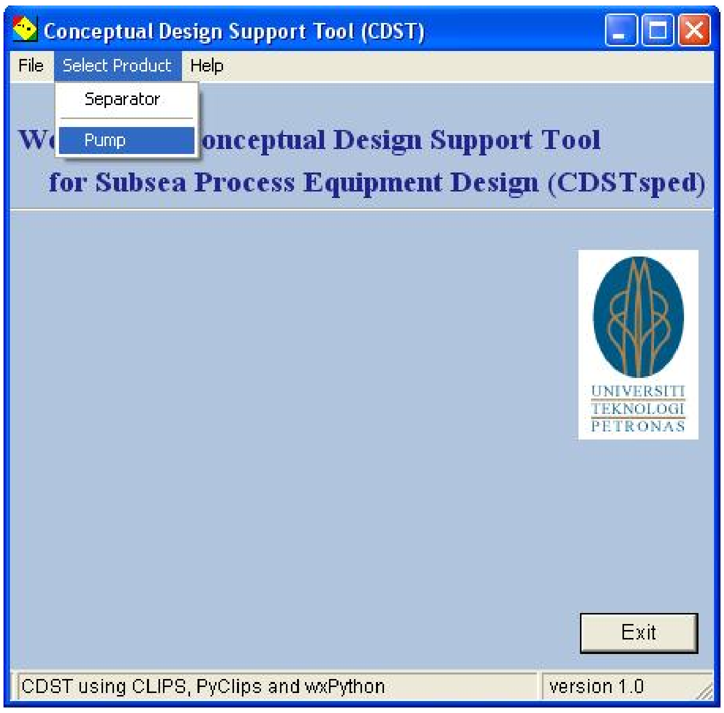

Conceptual Design Support Tool (CDST)
File
Select Product
Help
Separator
Pump
We onceptual Design Support Tool
for Subsea Process Equipment Design (CDSTsped)
UNIVERSITI TEKNOLOGI PETRONAS
Exit
CDST using CLIPS, PyClips and wxPython
version 1.0

Figure 5.19 Screenshot of CDSTsped initial window

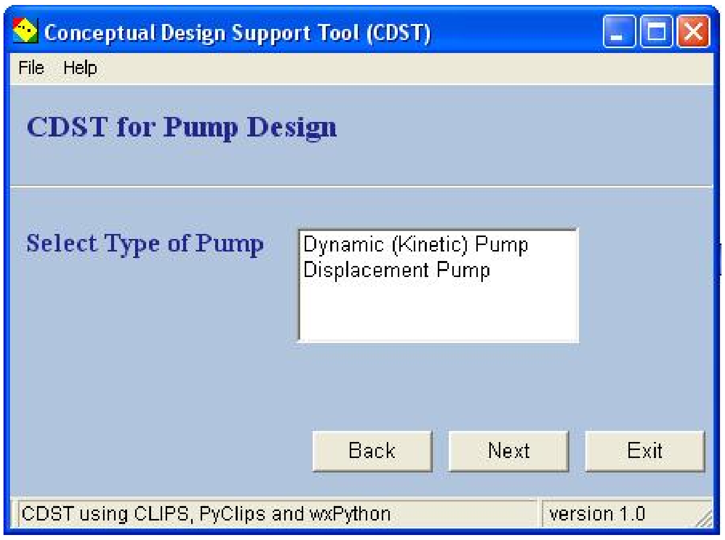
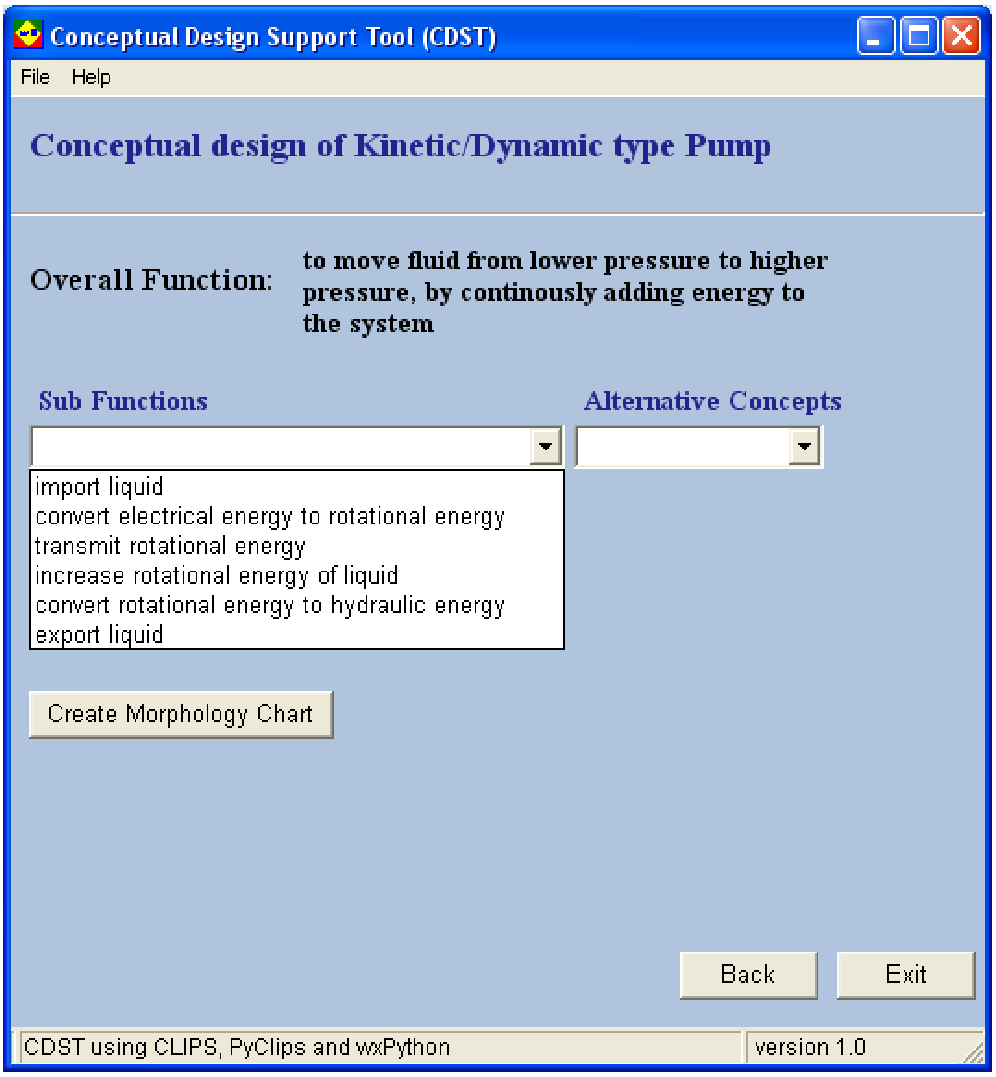


Figure 5.20 Selection of pump type

When the user selects the dynamic (kinetic) pump type from Figure 5.20 and clicks on the "Next" button, a kinetic design window which shows the overall function, the subfunctions and all the alternative concepts generated from the alternative concepts database is displayed (Figure 5.21). However, if the user selects displacement pump from Figure 5.20 , a displacement pump design window (Figure 5.22) will appear where the user can select the energy type and view the subfunctions and the generated alternative concepts.



Conceptual design of Kinetic/Dynamic type Pump

Overall Function: to move fluid from lower pressure to higher pressure, by continously adding energy to the system

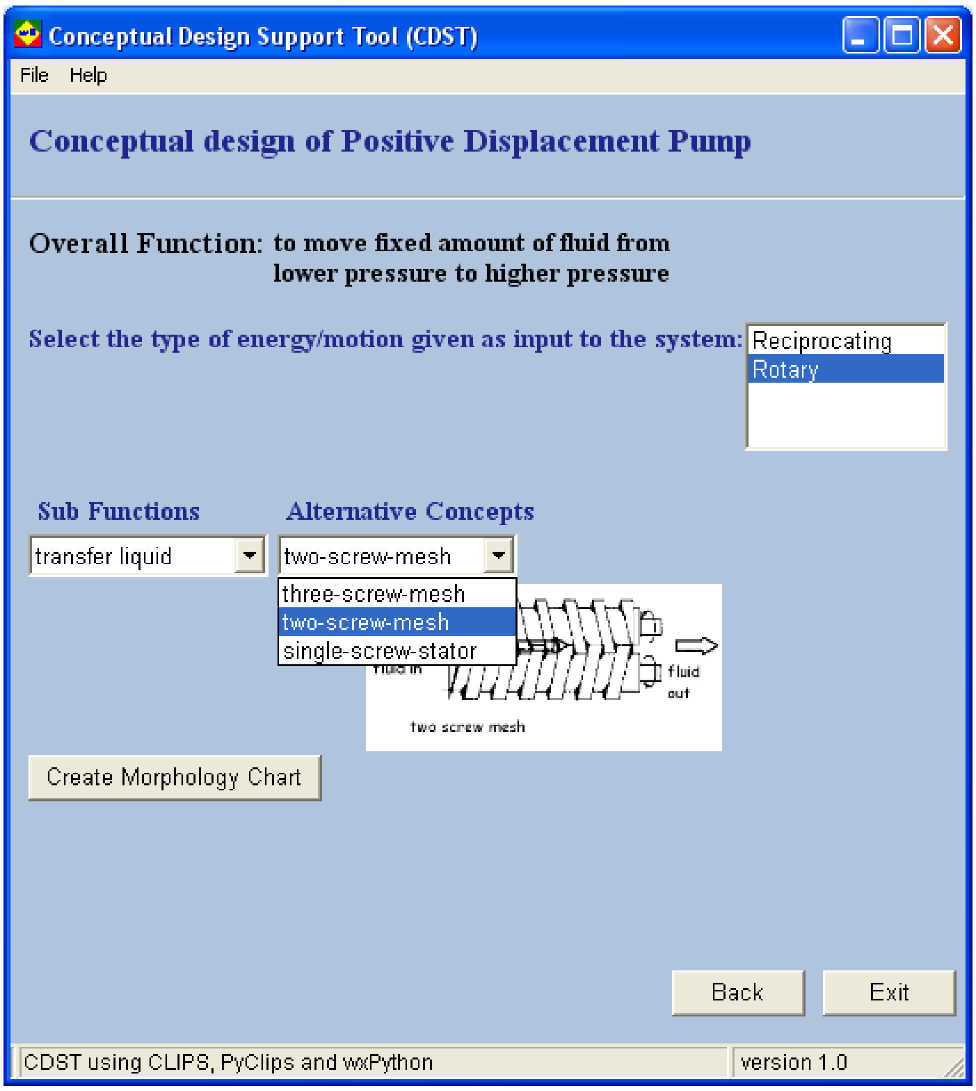
Sub Functions

- import liquid
- convert electrical energy to rotational energy
- transmit rotational energy
- increase rotational energy of liquid
- convert rotational energy to hydraulic energy
- export liquid

Alternative Concepts

CDST using CLIPS, PyClips and wxPython | version 1.0

Figure 5.21 Kinetic pump design window


Conceptual Design Support Tool (CDST)

File Help

Conceptual design of Positive Displacement Pump

Overall Function: to move fixed amount of fluid from lower pressure to higher pressure

Select the type of energy/motion given as input to the system: Reciprocating [✓ Rotary]

Sub Functions: transfer liquid

Alternative Concepts: two-screw-mesh (three-screw-mesh, [✓ two-screw-mesh], single-screw-stator)

CDST using CLIPS, PyClips and wxPython | version 1.0

Figure 5.22 Viewing alternative concepts for a given subfunction

Similarly if the user selects the separator option in Figure 5.19, the separator design window shown in Figure 5.23 will appear. In this window the user has to select either a two-phase or three-phase flow separator as the overall function. Based on this selection the system will populate the respective subfunctions and generate their alternative concepts.

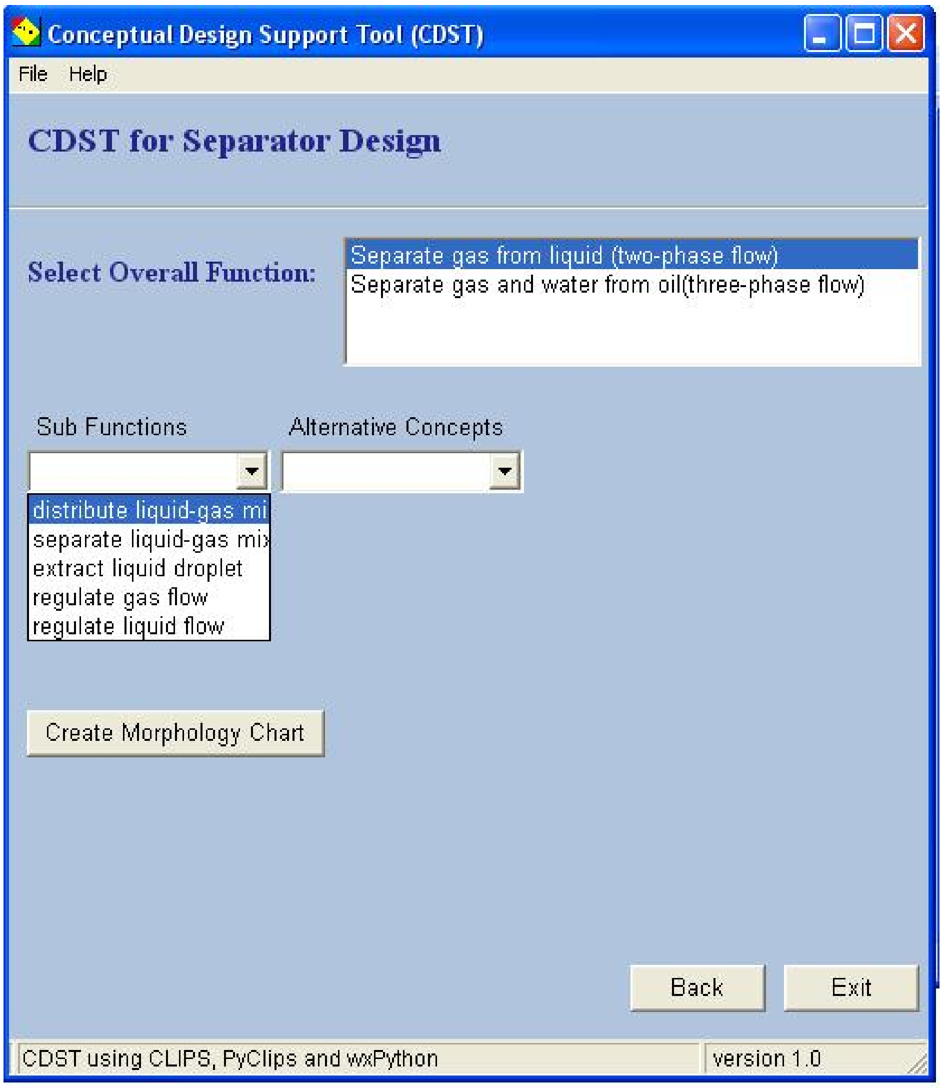


Figure 5.23 Separator design window

In all cases (Figure 5.21, Figure 5.22, Figure 5.23) the next step is to view all the subfunctions and their alternative concepts on the morphology chart using the “Create Morphology Chart” button. The remaining design process such as concept combination and concept evaluation follow the same procedure as CDST and will not be repeated here.

The CDSTsped demonstrated in this section shows its importance in preserving the design knowledge for future use and to train novice designers. For each product designed in given company similar tool can be developed with the methodology proposed in this thesis. The importance of such tools is inevitable with the current high turnover of experienced designers looking for better payment and retirement.

CHAPTER 6: VERIFICATION AND VALIDATION OF CDST

6.1 Verification of CDST

The conceptual design support tool has undergone verification tests by reviewers. Verification has been defined as the process of evaluating a system or component to determine whether the products of a given development phase satisfy conditions imposed at the start of that phase (Rakitin, 2001). Accordingly, the verification of CDST has been done by two lecturers (both having PhD degree) from computer and information science department of Universiti Teknologi Petronas (UTP). The main purpose of this verification process was to test run and inspect the program and verify the underlying program logic is correct. Both gave positive response with minor comments to improve the GUI and make it more users friendly and standardize. The comments were taken positively and necessary changes have been made accordingly before the software undergoes validation test in the next phase.

6.2 Validation Test

Validation has been defined as the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements (Rakitin, 2001). Validation activities are performed after the software is developed to determine if the software correctly implements the requirements. The standard approach for validation is to collect data from the system under study and compare them to their model counter parts. The manual conceptual design process, from which the conceptual design model is developed, can be compared with CDST, had the objective been automating the conceptual design process. However, the objective here is to assist human designer during the conceptual design process with the developed tool, and it cannot be directly compared with human designer. Instead, the validation is done by performing validation tests by independent experts in the field. The objective of validation test is to determine if the software meets all of its requirements as defined in the software requirements specification (Section 3.3.1).

6.2.1 Selection of Evaluators

Several attempts have been made to get experts from industry working on subsea process equipment design to test CDST. Unfortunately, this did not materialize because of different reasons. One of the reasons is that a number of companies working on the subsea process equipment design have manufacturing plants here in Malaysia, but the designs come from overseas. Our requests to test the tool were referred to the parent company located overseas. The next option explored to get design experts within our premises is faculty members and postgraduate students who have taught design courses and have industrial experience. Accordingly, three faculty members and three PhD students were nominated to perform validation test on CDST. Five of the evaluators have master degree and one PhD degree in mechanical engineering. All the evaluators have more than eight years of work experience with some having both academic and industrial experiences.

6.2.2 Evaluation Methods and Procedures

The test methods employed were both functional and Act-like-a-customer (ALAC) tests (Rakitin, 2001). In functional or black box testing, the test is strictly based on the requirements and the functionality of the tool; where as in ALAC testing the tests is developed based on knowledge of how customers use the software. Evaluation metrics were prepared based on standard test types for the experts to get their ratings. The test types used were:

- Functional test to determine if specific functions/features work as specified, i.e., test if all the steps in conceptual design process included in the software are working;
- Positive testing to determine if a feature produces results that are consistent with the stated requirements when the software is used properly;
- Startup/shutdown testing to determine if startup and shutdown functions have been implemented correctly; and

- Usability test to determine the user interface features behavior, as would be expected by trained or untrained users.

Based on these tests and the general objective of the tool, evaluation criteria in the form of questionnaire was prepared and given to the evaluators.

Before the evaluators use CDST, two conceptual design problems (problem I: three-phase separator design demonstrated in Section 5.2 and problem II: design of handheld nailer) have been given to them to perform manual conceptual design for two weeks. In each case, a short description about the design problem and a conceptual functional structure (functional modeling) were given. The main purposes in performing manual conceptual design were:

- to refresh the evaluators with manual conceptual design process;
- to help the evaluators judge how supportive the tool is and its coverage with regards to the steps in conceptual design process; and
- to compare the manually generated concepts with the concepts generated by the tool and if there are new concepts generated by the evaluators to demonstrate the knowledge acquisition process.

After the evaluators did the manual conceptual design process, they were briefed how to use the CDST and given the help document. They next perform conceptual design using CDST for the two problems followed by other design problems whose alternative concepts are already stored in the database.

6.2.3 Validation Test Results

I. Comparison between Manual and Software Generated Concepts

The evaluators generated concepts based on their personal experience for each subfunction in the functional structure of the design problems given. However, some of the evaluators combine the subfunctions and generate concepts, while others omit to generate concepts for some of the subfunctions. This makes the direct comparison between the concepts generated by the tool with concepts generated manually a bit difficult. Nevertheless, the concepts generated by two of the evaluators comply with tools output. Thus, the concepts generated by both evaluators are compared with the concepts generated by the CDST and presented in Table 6.1. Even though this is not intended to be quantitative proof of the efficiency of the tool, in both cases CDST generates more number of concepts than the manual concept generation. The new concepts generated by the evaluators are archived in the alternative concepts database using the knowledge acquisition module provided for future use. Sample concepts generated by one of the evaluators are shown in Figure 6.1.

Table 6.1 Overview of the number of concepts generated by two of the evaluators compared with CDST

Problem	Concepts generated using CDST	Evaluator	Concepts generated by the evaluator		
			Total concepts generated	Concepts similar with CDST	New concepts
Problem 1	15	I	13	6	3
Problem 2	9		4	4	
Problem 1	15	II	9	9	
Problem 2	9		7	6	1

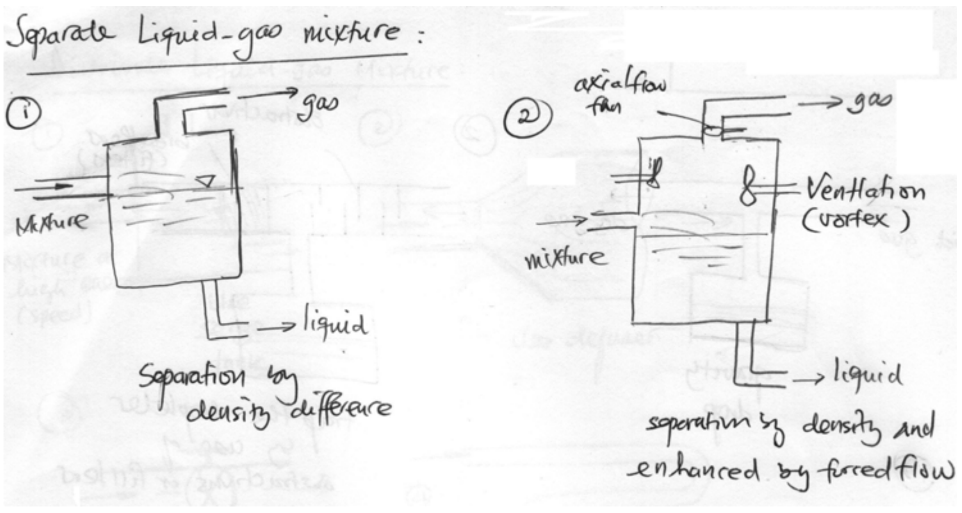

Separate Liquid-gas mixture :
①
gas
Mixture
liquid
Separation by density difference
②
axial flow fan
gas
Ventilation (vortex)
mixture
liquid
separation by density and enhanced by forced flow

Figure 6.1 Concepts generated by the evaluators to separate liquid-gas mixture

II. Validation Test Ratings

The evaluators use the software and test for functional, positive, startup/shutdown and usability tests and rate the CDST on the questionnaire prepared. The questionnaire is found in Appendix C. The evaluators rate the tool with respect to each criterion in the questionnaire as: 5 = Outstanding, 4 = Good, 3 = Satisfactory, 2 = Poor, and 1 = Unsatisfactory.

The percentage rating of each criterion by the evaluators in the questionnaire is summarized as follows:

1. *Functional test*: The functional tests are evaluated with the evaluation criteria number 1-4. The overview of percentage ratings for these criteria is shown in Figure 6.2.

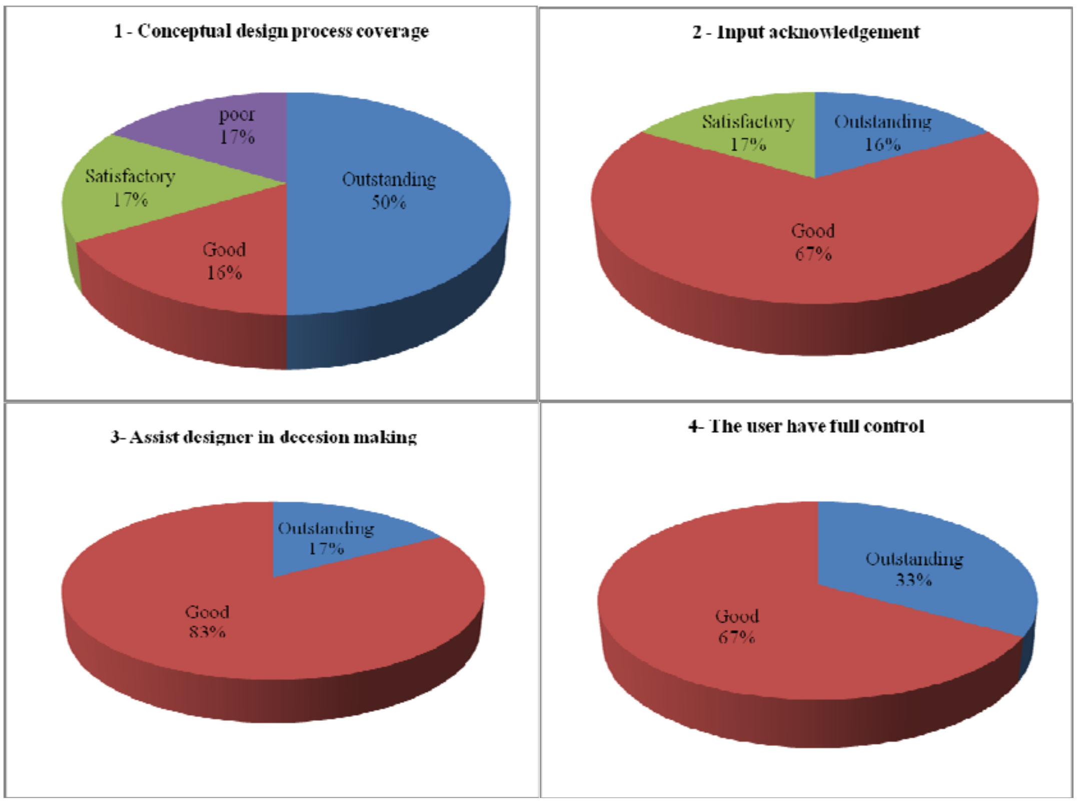


Figure 6.2 Percentage rating for functional testing

2. *Usability test*: The usability test is evaluated with evaluation criteria number 5 to determine how comfortable the users are with the organization of the user interface. As shown in Figure 6.3, this criterion got the lowest rating with 50%

of the evaluators rating as good, 17% as satisfactory and the remaining 33% as poor. This can be rectified by improving some of the features of the user interface and familiarizing the users with the tool. In practice, it takes some time to learn how to use any new software until the user gets use to it.

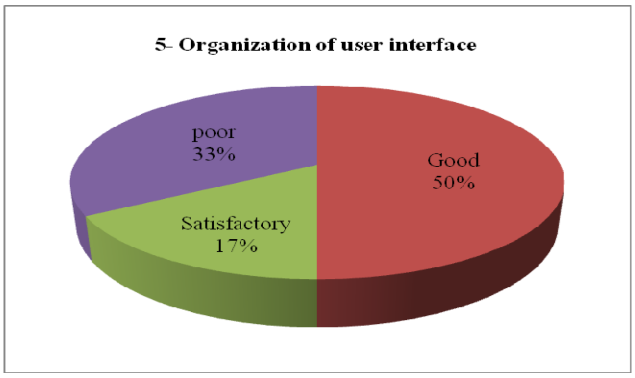


Figure 6.3 Evaluators' rating for organization of the user interface

3. *Startup/shutdown and positive tests:* The startup/shutdown and positive tests are evaluated with criteria number 6 and 7 respectively. Evaluators' rating results for these tests are shown in Figure 6.4.

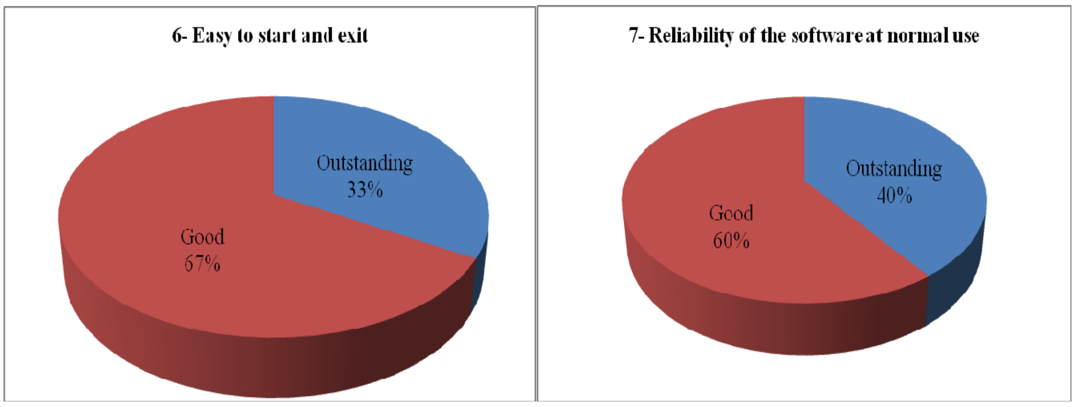


Figure 6.4 Startup/shutdown and positive test results

4. General use test: The remaining criteria are used to get the evaluators opinion on the general use of the tool. Figure 6.5 shows the summary of evaluators rating on the general use of the tool which includes: enhancing creativity, training aid for conceptual design education, preserving experts' knowledge for future, and demonstrating conceptual design can be computer assisted. The

last evaluation criterion is on whether the software achieves its purpose or not. The rating for this criterion is shown in Figure 6.6.

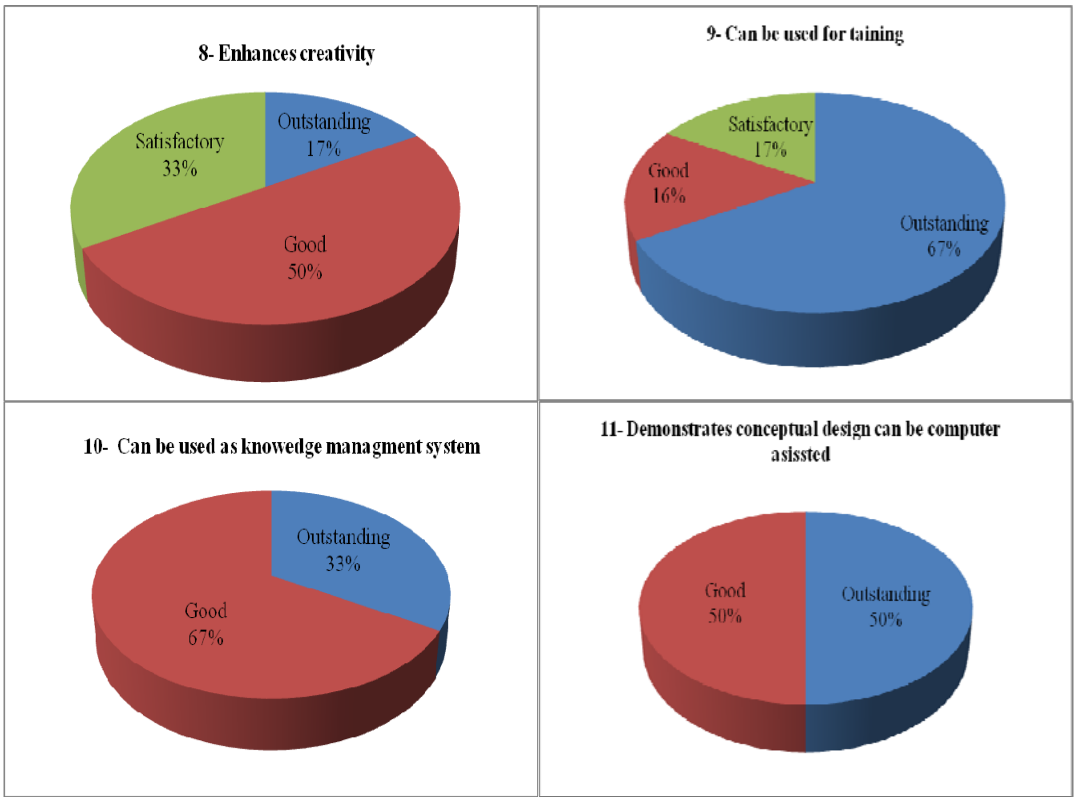


Figure 6.5 Summary of evaluators rating on general use of CDST

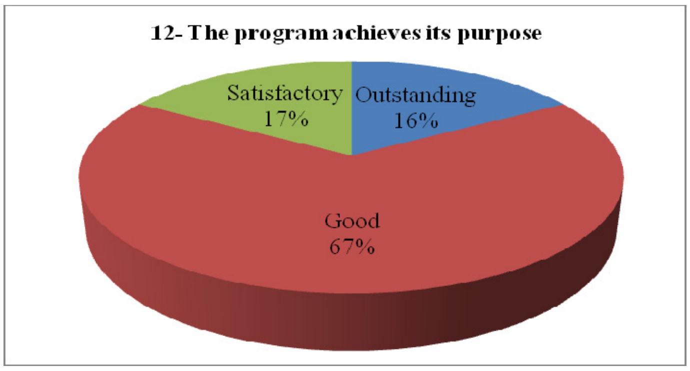


Figure 6.6 Evaluators rating on the overall achievement of the program

In general the overall result from the validation test is good indicating areas where further improvements are required. Tool development is cyclic process which involves continues refinements and evaluations to satisfy the users demand.

In addition to the ratings, the evaluators also gave comments and suggestion to improve the tool. The comments include:

- The system did not allow saving what have been done.
- The system did not have undo options if the user made mistake in between.

The first comment is addressed by generating a text file that automatically save the design history and the actions taken by the user. Therefore, the current version of CDST saves the design history in text format. As an example, an excerpt from the saved design history is shown in Figure 6.7. Furthermore, the subfunctions are saved in a separate text file in a reloadable format, so that if the designer wants to repeat the design some other time to load directly to the system in the function library window. The second comment on the undo option is also addressed on the function library window (Figure 5.3), by adding the "Undo Add Function" button to retract subfunction from the working memory.

```
Design summary
Date: 2009-07-06
Time: 09:00
This is an automatically generated file while you are conducting
conceptual design process using the conceptual design support tool
(CDST).

The overall function of the design problem is: provide translational
energy
This overall function is decomposed into the following subfunctions:
convert electrical energy to rotational energy
convert rotational energy to translational energy

The result of the concept generation process is as follows: (Subfunction
followed by alternative concepts)
convert electrical energy to rotational energy == > electric-motor;
convert rotational energy to translational energy == > slider-crank-
mechanism;   cam;   screw-drive;   rack-and-pinion;

You have considered the following concepts infeasible and rejected:
cam; screw-drive;

******************************
****CONCEPT COMBINATION*******
*******************************
The theoretically possible concept variants are:

variantno1:(electric-motor) + (slider-crank-mechanism)
variantno2:(electric-motor) + (rack-and-pinion)
```

Figure 6.7 Excerpt from the saved design history

6.3 Summary

In this chapter, the proposed function-based conceptual design support system is verified and validated by experts. Valuable comments have been obtained from the evaluators and addressed. The validation process is done with only six evaluators; thus, further tests needs to be done by mechanical engineers in the industry with varying degree of experience to improve the tool. The effectiveness of the tool with respect to the number of concepts generated and the time required in conducting the conceptual design process with and without the tool has not been addressed in the validation test.

CHAPTER 7: CONCLUSION AND RECOMMENDATIONS

7.1 Contributions of the Research

Motivated by the need to support designers with computer tools during conceptual design process where major design decisions are made with imprecise and incomplete information, the research reported in this thesis proposed an approach that integrates human creativity with computer capabilities to perform conceptual design efficiently than solely manual design. The system is based on design reuse philosophy. Design reuse plays a major role in product development especially during the early concept generation phase supplementing the designer’s knowledge by providing stored knowledge outside the designer’s area of expertise.

The methodology used in this thesis is based on systematic design approach integrated with knowledge-based system. Complex design problems are represented in functional terms and decomposed systematically into less complex subfunctions using top-down design decomposition manually. These subfunctions are selected from the function library developed via its GUI and given as input to the system. The solutions or alternative concepts for those subfunctions are generated from the database of the knowledge-based system which stores past design solutions. The solutions are then composed to achieve the overall function (concept variants) from which one or two concept variants are selected based on their merits for further development using successive concept evaluation methods provided.

The methodology proposed in this research has been demonstrated in a computer system. This demonstration was accomplished by using public domain open source programming environments (CLIPS, wxPython, pyCLIPS, and Python). The following modules have been developed to achieve the overall objective of the research:

- *Functional modeling*: Provide exhaustive function library that can be used for mechanical conceptual design process. This will help to define and represent functions in such a way that it is understood both by the designer and the computer. The function library also assists designers by providing a clue where to stop functional decomposition, i.e., functional decomposition should be stopped when all the subfunctions can be represented with the elemental mechanical functions (functions in the library).

- *Concept generation*: This includes representing and archiving concepts in the database, generating concepts using domain independent production rules, assisting designers in performing concept generation manually, and displaying the generated concepts on morphology chart both schematically and textually.

- *Concept combination*: Provide domain independent production rules to combine generated concepts to create concept variants and displaying the concept variants textually and schematically. It also includes flow compatibility criterion to reduce the combinatorial explosion for non branching single flow functional structures.

- *Concept evaluation*: This includes assisting designers to define evaluation criteria for a given design problem and evaluating concept variants using absolute comparison, concept screening and weighted decision matrix methods.

- *Knowledge acquisition module*: This module helps to acquire knowledge from the designer and save it in the database during the program development and the software life time without modifying the source code. This will assist the designer to conduct manual concept generation or capture expert's knowledge and archive in the computer system for future use.

- *Central graphical user interface*: This includes the development of a user friendly graphical user interface (GUI) through which the user interacts with the system. The GUI consists of standard windows which can easily be controlled by mouse using buttons, menus, choice items, and popup menus to perform conceptual design process and explore the design options giving visual feedback about the actions being performed.

The main contributions of the research presented in this thesis can be summarized as follows:

i. The developed conceptual design support tool allows designers to carryout conceptual design process with the aid of computers. Once the designer is familiar with the developed tool, the designer can use the tool to perform some of the repetitive and time consuming tasks. The monotonous activities supported by the tool include concept generation form database, accepting new concepts from the designer and archiving in the database for future use, displaying the generated concepts on the morphology chart, concept combination and creating evaluation matrices for concept evaluation process. From the interaction between the tool and the designer and the nature of the conceptual design process, it is difficult to automate conceptual design in general since there are cases where human interventions (decision) are required.

ii. The proposed domain independent production rules make CDST generic and easily expandable tool. Furthermore, the knowledge acquisition module introduced helps to gain conceptual design knowledge throughout the tool's life time. The tool can be updated with new design concepts over time and takes into consideration future inventions to be included. This makes CDST novel compared to other tools such as MODESSA (Kersten, 1995), Web-based morphological chart (Huang and Mak, 1999), and EFDEX (Zhang et al., 2001b).

iii. It is possible to develop customized tool following the proposed framework for specific domain of application. For example, CDSTsped presented in Section 5.3, is developed to specifically address subsea processing equipment design. Currently there is no known conceptual design tool to address this domain.

iv. The electronic version of morphological chart developed in this research which displays schematically all the alternative concepts generated can save time compared to manual morphological charts posted on the wall which requires redrawing the concepts each time used. On the other hand, like manual design process, the designer has greater control over the generated concepts where infeasible concepts can be rejected right on the morphology chart before the concept combination process.

v. CDST can be used as knowledge management system to capture and reuse design knowledge in industry. Design knowledge resides in the brains of experienced designers. This knowledge can be archived into the computer system following the proposed knowledge representation scheme. In general, CDST provides a way to capture abstract design knowledge in the form of concepts which will be used by the less experienced designers to complement their knowledge.

vi. CDST can also be used as inspirational tool. Exploring the generated concepts of the tool can stimulate cognitive process and generation of new ideas (Benami and Jin, 2002, Chakrabarti et al., 2005). However, it is always advisable to generate concepts manually before using the tool to minimize mental fixation to the existing concepts. Thus, the designer must always try to generate concepts manually and then use the tool to see other options and generate more concepts inspired by the existing ones.

vii. CDST can be used as means to train designers about conceptual design process.

7.2 Critique of the Research

The conceptual design support tool presented in this thesis demonstrated how conceptual design process can be computer assisted with existing design knowledge archived in computer system. However, there are some limitations in the current version of the tool which requires further research to enhance its functionality. These limitations are discussed next.

The first limitation is on functional modeling. The tool does not have mechanism to decompose the overall function into subfunctions by itself. Because of the subjectivity in functional decomposition, it is not possible to ensure different designers to achieve identical functional structure which makes it difficult to generalize decomposition rule for all products. Thus functional decomposition is done manually by the designer with the aid of the function library developed to assist as a stopping criterion when the decomposition reaches the elemental functions in the library. However, for specific product it is possible to develop production rules that can automatically provide the subfunctions for a given overall function as demonstrated with CDSTsped.

The second limitation is on the number of concepts generated and means to sort based on importance. Currently, the alternative concepts database is limited to subsea process equipment design and few common mechanical design problems added during the validation process. This limits the number of concepts generated. However, the concepts database can easily be expanded with the proposed knowledge acquisition module. In relation to this, the tool provides a means to generate and display on morphology chart as many alternative concepts as possible depending on the availability of concepts in the database. However, the approach reported in this thesis lacks the means to sort based on importance so that it could be easy for the user to reject the less likely alternative concepts. Currently the designer rejects the less likely concepts based on experience.

The third limitation is the number of subfunctions handled at a time for concept combination process. The available rules to combine flow compatible concept variants in the knowledge base is limited to a maximum of ten subfunctions at a time in addition to the requirement that the functional structure should be single flow and none branching. Although the rules for combining theoretically possible concept variants have no limitation on the number of subfunctions, during test runs, because of memory limitations on the available computers it was not possible to run the program for more than ten subfunctions at a time. Thus, when there are more than ten subfunctions in the functional structure, the user needs to divide manually and give only a maximum of ten subfunctions at a time and combine later to get the overall solution.

The fourth limitation is related with the incremental addition of new concepts to the database. In its current version, the tool is stand alone and works only on the computer on which the software is installed. Further research to make the tool server based and accessible through intranet and/or internet would maximize the use of concepts from designers in different locations.

7.3 Recommendation for Future Research

Although the research presented in this thesis has lay down the framework how conceptual design process can be computer assisted with the developed conceptual design support tool, the effectiveness of the tool would benefit from additional research. In addition to those pointed out in the previous section (Section 7.2) as limitations, areas that need further research are summarized as follows:

i. Conceptual design is inherently evolutionary process, where the requirements change as the design process progresses. This is because, decisions made at one point creates additional requirements to the design and needs to be addressed. The approach presented in this thesis deals with only initial static set of requirements and does not support the evolutionary changes, even though it can give suggestion to the user regarding the side effects from the generated concepts to be considered as additional requirements. Thus, further research to extend the dynamic functionality of the tool would be an added advantage in supporting the designer.

ii. With increasing in the number of concepts in the database the number of alternative concepts generated and their possible combination will become difficult to evaluate. One of the possible options to reduce the combinatorial explosion in addition to those proposed in this thesis includes geometric compatibility. Even though this research is not aimed at addressing challenges involving decisions about parametric details that govern the shape, or geometry of a component, further research to include the parametric details can help to include geometric compatibility criterion. The geometric compatibility criterion reduces the number of concept variants by combining only those concepts which are geometrically compatible. To extend the existing tool to address geometric compatibility the concepts geometric and material information should be captured and new production rules needs to be written.

iii. Eventually the concept variants are further embodied with currently available commercial CAD tools. Further research to integrate CDST with those CAD tools is required. This will enable to easily modify concepts, create their 3D model, and conduct simulation studies to evaluate the concept variants in addition to the current evaluation methods implemented.

REFERENCES

Allada, V. (2001) Feature-based design in integrated manufacturing. In *Computer-Aided Design, Engineering, and Manufacturing: Systems Techniques and Applications.* Leondes, C. T. (Ed.) Vol. 5. CRC Press.

Altiok, T. & Melamed, B. (2007) *Simulation Modeling and Analysis with Arena*, Elsevier Inc.

Arnold, K. & Stewart, M. (1999a) *Surface Production Operations: Design of Gas-Handling systems and Facilities,* 2nd ed. Vol. 2, Butterworth-Heinemann.

Arnold, K. & Stewart, M. (1999b) *Surface Production Operations: Design of Oil-Handling Systems and Facilities,* 2nd ed. Vol. 1, Butterworth-Heinemann.

Benami, O. & Jin, Y. (2002) Creative Stimulation in Conceptual Design. In *DETC'02.* Montreal, Canada, ASME pp 251-263.

Blessing, L. T. M. & Chakrabarti, A. (2009) *DRM, A Design Research Methodology*, Springer-Verlag London Limited.

Boley, R. E. (1985) Horizontal gas and liquid separator. *In U.S. Patent*, Patent No: 4539023.

Bonnema, G. M. & Houten, F. J. A. M. V. (2004) Conceptual Design in a high-tech environment. In *Methods and tools for co-operative and integrated design.* Tichkiewitch, S. & Dordrecht, D. B. (Eds.). Kluwer Academic Publishers.

Bracewell, R. H. & Sharpe, J. E. E. (1996) Functional descriptions used in computer support for qualitative scheme generation—Schemebuilder. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 10**,** 333-346.

Brunettia, G. & Golob, B. (2000) A feature-based approach towards an integrated product model including conceptual design information. *Computer-Aided Design,* 32**,** 877-887.

Bryant, C. R., Mcadams, D. A., Stone, R. B., Kurtoglu, T. & Campbell, M. I. (2005) A Computational Technique for Concept Generation. In *IDETC/CIE 2005.* Long Beach, California USA, ASME pp 267-276.

Chakrabarti, A. & Blessing, L. (1996) Special Issue: Representing functionality in design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 10**,** 251-253.

Chakrabarti, A. & Bligh, T. P. (2001) A scheme for functional reasoning in conceptual design. *Design Studies,* 22**,** 493-517.

Chakrabarti, A., Sarkar, P., Leelavathamma, B. & Nataraju, B. S. (2005) A functional representation for aiding biomimetic and artificial inspiration of new ideas. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 19**,** 113-132.

Chiang, W.-C., Pennathur, A. & Mital, A. (2001) Designing and manufacturing consumer products for functionality: a literature review of current function definition and design support tools. *Integrated Manufacturing Systems,* 12**,** 430-448.

Cole, E. L., Jr. (1998) Functional analysis: a system conceptual design tool *IEEE Transactions on Aerospace and Electronic Systems,* 34**,** 354-365.

Collins, J. A., Hagan, B. T. & Bratt, H. M. (1976) The Failure-Experience Matrix - A Useful Design Tool. *Journal of Engineering in Industry,* 98**,** 1074-1079.

Cross, N. (2008) *Engineering Design Methods: Strategies for Product Design,* 4th ed, John Wiley & Sons, Ltd.

Deng, Y.-M., Britton, G. A. & Tor, S. B. (1998) A Design Perspective of Mechanical Function and its Object-Oriented Representation Scheme. *Engineering with Computers,* 14**,** 309-320.

Dieter, G. E. (2000) *Engineering Design: A Materials and Processing Approach,* 3rd ed, New York McGraw-Hill.

Ditria, J. C. & Hadfield, D. A. (2001) Subsea multiphase fluid separating system and method. *In U.S. Patent*, Patent No: 6197095 B1.

Dunn, R., *wxPython,* Last accessed on 2 July 2009, http://www.wxpython.org/.

Erdena, M. S., Komotoa, H., Beeka, T. J. V., D'amelioa, V., Echavarriaa, E. & Tomiyamaa, T. (2008) A review of function modeling: Approaches and applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 22**,** 147-169.

Filho, F. H. D. C. (1992) Deep-water oil and gas production and transportation system. *In U.S. Patent*, Patent No: 5154741.

Fisher, A. S. (1991) *CASE: Using Software Development Tools,* 2nd ed, John Wiley &Sons, Inc.

French, M. (1998) *Conceptual Design for Engineers,* 3rd ed, London, Springer.

Garosi, F., (2008a) *PyClips,* Last accessed on 2 July 2009, http://pyclips.sourceforge.net/web/?q=mainpage.

Garosi, F. (2008b) Pyclips User Manual.

Giarratano, J. C. (2007) CLIPS User's Guide.

Giarratano, J. C. & Riley, G. (1998) *Expert systems : principles and programming,* 3rd ed, Boston, PWS Publishing.

Goel, A. K. & Chandrasekaran, B. (1990) A task structure for case-based design. In *IEEE International Conference on System, Man and Cybernetics.* pp 587-592.

Hashim, F. M., Juster, N. P. & Pennington, A. D. (1994) A Functional Approach to Redesign. *Engineering with Computers,* 10**,** 125-139.

Hatton, G. J. (1998) Power efficient multi-stage twin screw pump. *In U.S. Patent*, Patent No: 5779451.

Hirtz, J., Stone, R., Mcadams, D., Szykman, S. & Wood, K. (2002) A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts. *Research in Engineering Design,* 13**,** 65-82.

Hopgood, A. A. (2001) *Intelligent Systems for Engineers and Scientists,* 2nd ed, CRC Press.

Hsu, W. & Liu, B. (2000) Conceptual design: issues and challenges. *Computer-Aided Design,* 32**,** 849-850.

Hsu, W. & Woon, I. M. Y. (1998) Current research in the conceptual design of mechanical products. *Computer-Aided Design,* 30**,** 377-389.

Huang, G. Q. & Mak, K. L. (1999) Web-Based Morphological Charts for Concept Design in Collaborative Product Development. *Journal of Intelligent Manufacturing,* 10**,** 267-278.

Hundal, M. S. (1990) A Systematic Method for Developing Function Structures, Solutions and Concept Variants. *Mechanism and Machine Theory,* 25**,** 243-256.

Jager, A. (1994) Eccentric screw pump. *In U.S. Patent*, Patent No: 5358390.

Kamrani, A. & Vijayan, A. (2006) A methodology for integrated product development using design and manufacturing templates. *Journal of Manufacturing Technology Management,* 17**,** 656-672.

Karassik, I. J., Messina, J. P., Cooper, P. & Heald, C. C. (Eds.) (2001) *Pump Handbook*, McGRAW-HILL.

Kersten, T. (1995) MODESSA: A Computer Based Conceptual Design Support System. In *AI system support for conceptual design, Lancaster International workshop on engineering design.* Sharpe, J. (Ed.). Lancaster, Springer pp 241-259.

Kirschman, C. F. & Fadel, G. M. (1998) Classifying Functions for Mechanical Design. *Journal of Mechanical Design,* 120**,** 475-482.

Konar, A. (2000) *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain*, CRC Press.

Kurfman, M. A., Stock, M. E., Stone, R. B., Rajan, J. & Wood, K. L. (2003) Experimental Studies Assessing the Repeatability of a Functional Modeling Derivation Method. *Journal of Mechanical Design,* 125**,** 682-693.

Kuttig, D. (1993) Potential and Limits of Functional Modelling in the CAD Process. *Research in Engineering Design,* 5**,** 40-48.

LPA, Logic Programming Associated Ltd, (n.d.) *WIN-PROLOG with Chimera, VisiRule and Flex*, Last accessed on 17 May 2007, http://www.lpa.co.uk/dow_tri.htm.

Luger, G. F. & Stubblefield, W. A. (1998) *Artificial Intelligence: Structures and Strategies for Complex Problem Solving,* 3rd ed, Addison Wesley Longman, Inc.

Lush, D., Eng, J., Ucok, H., Hopgood, D., Landeck, C. & Carmon, K. (2007) Subsea separation system. *In U.S. Patent*, Patent No: 7210530 B2.

Lyons, W. C. & Plisga, G. J. (Eds.) (2005) *Standard Handbook of Petroleum & Natural Gas Engineering*, Gulf Professional Publishing.

Mak, T. & Shu, L. (2008) Using descriptions of biological phenomena for idea generation. *Research in Engineering Design,* 19**,** 21-28.

Massinon, R. M. J. (1992) Multiphase fluid mass transfer pump. *In U.S. Patent*, Patent No: 5156537.

Mcmunigal, J. E., Ungvari, S., Slocum, M. & Mcmunigal, R. E. (2006) TRIZ. In *Mechanical Engineers' Handbook: Materials and Mechanical Design.* Kutz, M. (Ed.) Vol. 1, 3rd ed., John Wiley & Sons, Inc.

Miller, G. A. (1956) The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *Psychological Review,* 63**,** 81-97.

Moulianitis, V. C., Dentsoras, A. J. & Aspragathos, N. A. (1999) A knowledge-based system for the conceptual design of grippers for handling fabrics. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 13**,** 13-25.

Nikolopoulos, C. (1997) *Expert systems: Introduction to first and second generation and hybrid knowledge based systems*, Marcel Dekker, Inc.

Nilsen, P. J. & Wolff, E. A. (2005) Method and a system for separating a mixture. *In U.S. Patent*, Patent No: 6872239.

Pahl, G. & Beitz, W. (1996) *Engineering Design: A Systematic Approach,* 2nd ed, London, Springer.

Pugh, S. (1990) *Total Design: Integrated Methods for Successful Product Engineering*, Addison-Wesley Publishing Company.

Qiu, S. L., Fok, S. C., Chen, C. H. & Xu, S. (2002) Conceptual Design Using Evolution Strategy. *International Journal of Advanced Manufacturing Technology,* 20**,** 683–691.

Rakitin, S. R. (2001) *Software Verification and Validation for Practitioners and Managers,* 2nd ed, Artech House, Inc.

Rao, S. S., Nahm, A., Shi, Z., Deng, X. & Syamil, A. (1999) Artificial intelligence and expert systems applications in new product development - a survey. *Journal of Intelligent Manufacturing,* 10**,** 231-244.

Rappin, N. & Dunn, R. (2006) *wxPython in Action*, Manning Publications Co.

Rentema, D. & Jansen, E. (2000) An AI Tool for Conceptual Design of Complex Products. In *Design Research in the Netherlands 2000.* Achten, H., Vries, B. D. & Hennessey, J. (Eds.). Eindhoven University of Technology, pp 119-131.

Riley, G., (2008) *CLIPS: A Tool for Building Expert Systems,* Last accessed on 2 July 2009, http://pyclips.sourceforge.net/web/.

Robert O. Parmely, P. E. (Ed.) (2005) *Machine Devices and Components Illustrated Sourcebook*, MacGraw Hill.

Robertson, B. F. & Radcliffe, D. F. (2009) Impact of CAD tools on creative problem solving in engineering design. *Computer-Aided Design,* 41**,** 136-146.

Saaty, T. L. (1994) How to Make a Decision: The Analytic Hierarchy Process. *Interfaces,* 24**,** 19-43.

Salonen, M. & Perttula, M. (2005) Utilization of Concept Selection Methods - A Survey of Finnish Industry. In *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* Long Beach, California USA, ASME.

Sapihie (2007) Introduction to Deepwater. In *Deepwater Technology Workshop*. Universiti Teknologi Petronas.

Saruwatari, M. (1988) Progressive cavity pump. *In U.S. Patent*, Patent No: 4773834.

Scott, S. L., Devegowda, D. & Martin, A. M. (2004) Assessment of Subsea Production & Well Systems. In *Final Report Submitted to the U.S. Department of Interior – Minerals Management Service (MMS), Technology Assessment & Research (TA&R) Program*. Department of Petroleum Engineering, Texas A&M University.

Sen, P. & Yang, J.-B. (1998) *Multiple Criteria Decision Support in Engineering Design*, New York, Springer.

Sieger, D. B. & Salmi, R. E. (1997) Knowledge representation tool for conceptual development of product designs. In *1997 IEEE International Conference on Systems, Man, and Cybernetics.* pp 1936-1941.

Smart, J., (1997) *wxCLIPS,* Last accessed on 11 September 2007, http://www.anthemion.co.uk/wxclips/.

Stone, R. B. & Wood, K. L. (2000) Development of a Functional Basis for Design. *Journal of Mechanical Design,* 122**,** 359-370.

Sturges, R. H., O'shaughnessy, K. & Kilani, M. I. (1996) Computational model for conceptual design based on extended function logic. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 10**,** 255-274.

Sturges, R. H. J. R., O'shaughnessy, K. & Reed, R. G. (1993) A Systematic Approach to Conceptual Design. *Concurrent Engineering,* 1**,** 93-105.

Suh, N. P. (1990) *The Principles of Design*, Oxford University Press.

Szykman, S., Racz, J. W. & Sriram, R. D. (1999) The representation of function in computer-based design. In *1999 ASME Design Engineering Technical Conferences.* Las Vegas, Nevada, ASME pp 233-246.

Tomiyama, T. (2007) Intelligent computer-aided design systems: Past 20 years and future 20 years. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 21**,** 27-29.

Tong, C. & Gomory, A. (1993) A knowledge-based computer environment for the conceptual design of small electromechanical appliances. *Computer,* 26**,** 69-71.

Tor, S. B., Britton, G. A., Chandrashekar, M. & Wee, N. K. (1998) Functional design. In *Integrated Product and Process Development: Methods, Tools, and Technologies.* John M Usher, Utpal Roy & Parsael, H. R. (Eds.). New York, John Willey &Sons, Inc.

Ullman, D. G. (2003) *The mechanical design process,* 3rd ed, McGraw-Hill.

Ulrich, K. T. & Eppinger, S. D. (2004) *Product Design and Development,* 3rd ed, McGRAW-Hill inc.

Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y. & Tomiyama, T. (1996) Supporting conceptual design based on the function-behavior-state modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 10**,** 275-288.

Umeda, Y. & Tomiyama, T. (1997) Functional Reasoning in Design. *IEEE Expert***,** 42-48.

Vargas-Hernandez, N. & Shah, J. J. (2004) 2nd-CAD: A Tool for Conceptual Systems Design in Electromechanical Domain. *Journal of Computing and Information Science in Engineering,* 4**,** 28-36.

Wang, L., Shen, W., Xie, H., Neelamkavil, J. & Pardasani, A. (2002) Collaborative conceptual design - state of the art and future trends. *Computer-Aided Design,* 34**,** 981-996.

Woldemichael, D. E. & Hashim, F. M. (2007) Development of Computer Aided Conceptual Design Tool for Subsea Process Equipment Design. In *Conference on Design, Simulation, Product Development and Optimization.* Ripin, Z. M., Abdullah, M. Z., Hassan, A. Y., et al. (Eds.). Penang, Malaysia, pp 1-6.

Woldemichael, D. E. & Hashim, F. M. (2008) Concept Modeler: A Computer Aided Conceptual Design Support Tool. In *International Graduate Conference on Engineering and Science 2008 (IGCES 2008).* Johor Bahru, Malaysia.

Zhang, W. Y., Tor, S. B. & Britton, G. A. (2001a) A Prototype Knowledge-Based System for Conceptual Synthesis of the Design Process. *International Journal of Advanced Manufacturing Technology,* 17**,** 549-557.

Zhang, W. Y., Tor, S. B., Britton, G. A. & Deng, Y.-M. (2001b) EFDEX: A Knowledge-Based Expert System for Functional Design of Engineering Systems. *Engineering with Computers,* 17**,** 339-353.

Zuo, J. & Director, S. (2000) An Integrated Design Environment for Early Stage Conceptual Design. In *Design, Automation and Test in Europe Conference and Exhibition 2000.* IEEE.

Zwicky, F. (1967) The Morphological Approach to Discovery, Invention, Research and Construction. In *New Methods of Thought and Procedure: Contributions to the Symposium on Methodologies.* Zwicky, F. & Wilson, A. G. (Eds.). New York, Springer-Verlag.

APPENDIX A: SOURCE CODE AND SUPPORT DOCUMENTS

A.1 Contents of the Attached CD

The source codes for CDST and support documents are found in the attached CD. The attached CD contains:

i. Readme file on how to use the resources on the CD.

ii. CDST users guide in portable document format (pdf)

iii. CDST help document, a compiled html help file, in chm file format

iv. CDST software in different file formats

v. CLIPS, Python, PyClips, and wxPython installable programs

A.2 CDST in Different File Formats

CDST is prepared in three file formats for convenience. These are:

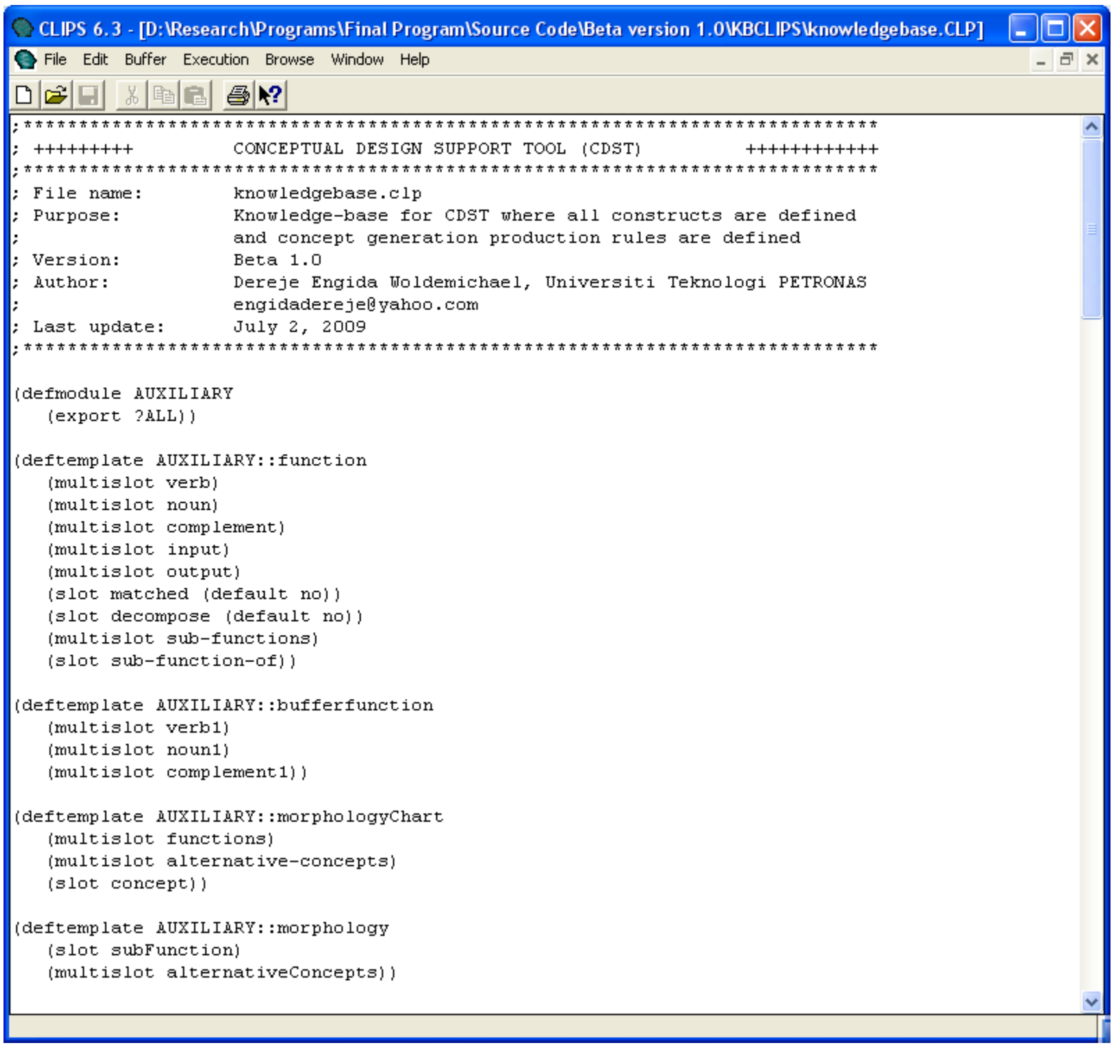
1. *Source code*: To run the program from the source code CLIPS, Python, PyClips, and wxPython software should be installed first.

2. *Self executable file format*: The source code is converted into an executable file format using a Python module known as Py2exe which is also open source software.

3. *Windows installable file format*: The executable files are converted into windows installable file format using Inno setup software. Inno setup is free installer for window programs.

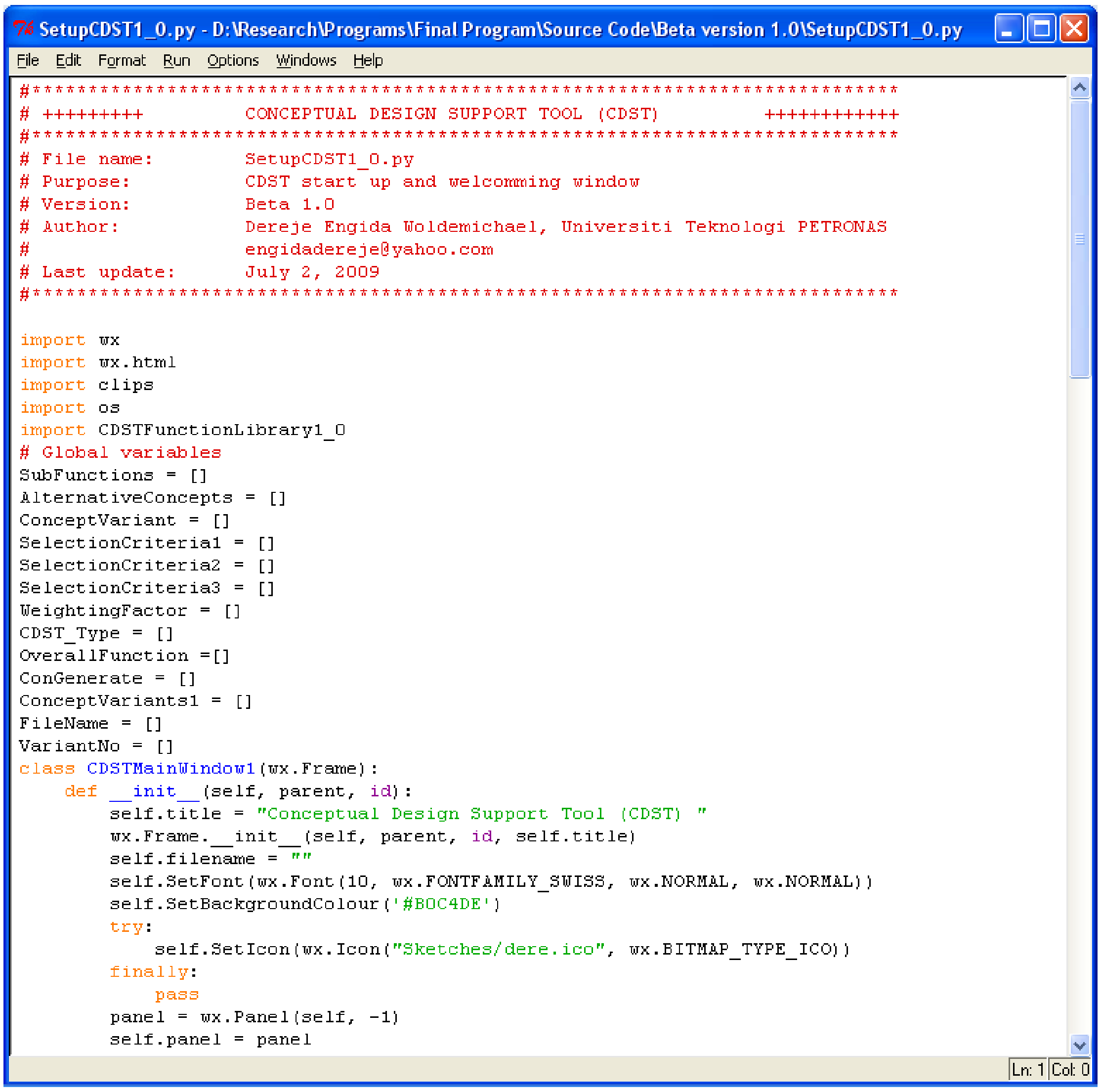
A.3 Source Code

The source code of CDST consists of thousands of lines and several files. Considering the number of pages required, only excerpts of sample program from the source code in CLIPS and Python are shown in Figure A.1 and Figure A.2 respectively.


CLIPS 6.3 - [D:\Research\Programs\Final Program\Source Code\Beta version 1.0\KBCLIPS\knowledgebase.CLP]
File Edit Buffer Execution Browse Window Help

```
;*****************************************************************************
; +++++++++          CONCEPTUAL DESIGN SUPPORT TOOL (CDST)          ++++++++++++
;*****************************************************************************
; File name:        knowledgebase.clp
; Purpose:          Knowledge-base for CDST where all constructs are defined
;                   and concept generation production rules are defined
; Version:          Beta 1.0
; Author:           Dereje Engida Woldemichael, Universiti Teknologi PETRONAS
;                   engidadereje@yahoo.com
; Last update:      July 2, 2009
;*****************************************************************************

(defmodule AUXILIARY
   (export ?ALL))

(deftemplate AUXILIARY::function
   (multislot verb)
   (multislot noun)
   (multislot complement)
   (multislot input)
   (multislot output)
   (slot matched (default no))
   (slot decompose (default no))
   (multislot sub-functions)
   (slot sub-function-of))

(deftemplate AUXILIARY::bufferfunction
   (multislot verb1)
   (multislot noun1)
   (multislot complement1))

(deftemplate AUXILIARY::morphologyChart
   (multislot functions)
   (multislot alternative-concepts)
   (slot concept))

(deftemplate AUXILIARY::morphology
   (slot subFunction)
   (multislot alternativeConcepts))
```

Figure A.1 An excerpt from CDST source code in CLIPS


SetupCDST1_0.py - D:\Research\Programs\Final Program\Source Code\Beta version 1.0\SetupCDST1_0.py
File Edit Format Run Options Windows Help
Ln: 1 Col: 0

```
#***************************************************************************
# +++++++++          CONCEPTUAL DESIGN SUPPORT TOOL (CDST)          ++++++++++++
#***************************************************************************
# File name:         SetupCDST1_0.py
# Purpose:           CDST start up and welcomming window
# Version:           Beta 1.0
# Author:            Dereje Engida Woldemichael, Universiti Teknologi PETRONAS
#                    engidadereje@yahoo.com
# Last update:       July 2, 2009
#***************************************************************************

import wx
import wx.html
import clips
import os
import CDSTFunctionLibrary1_0
# Global variables
SubFunctions = []
AlternativeConcepts = []
ConceptVariant = []
SelectionCriteria1 = []
SelectionCriteria2 = []
SelectionCriteria3 = []
WeightingFactor = []
CDST_Type = []
OverallFunction =[]
ConGenerate = []
ConceptVariants1 = []
FileName = []
VariantNo = []
class CDSTMainWindow1(wx.Frame):
    def __init__(self, parent, id):
        self.title = "Conceptual Design Support Tool (CDST) "
        wx.Frame.__init__(self, parent, id, self.title)
        self.filename = ""
        self.SetFont(wx.Font(10, wx.FONTFAMILY_SWISS, wx.NORMAL, wx.NORMAL))
        self.SetBackgroundColour('#B0C4DE')
        try:
            self.SetIcon(wx.Icon("Sketches/dere.ico", wx.BITMAP_TYPE_ICO))
        finally:
            pass
        panel = wx.Panel(self, -1)
        self.panel = panel
```

Figure A.2 An excerpt from the source code of GUI in Python environment

APPENDIX B: LIST OF SOFTWARE USED TO DEVELOP CDST

Table B.1 List of software used to develop CDST

Name	Version	Sources (download link)	Purpose	Remark
CLIPS	6.30 Beta	http://clipsrules.sourceforge.net/	To develop the knowledge-based system (KBS)	Open source
Python	2.5.1	http://www.python.org/	To integrate GUI with KBS	Open source
wxPython	2.8	http://www.wxpython.org/	To build the GUI	Open source
PyCLIPS	1.0.7.348	http://pyclips.sourceforge.net/web/	To embed CLIPS in Python program	Open source
Py2exe	0.6.9	http://www.py2exe.org/	To convert the source code into executable file	Open source
HTML Help Workshop	4.74.8702.0	http://www.softpedia.com/get/Authoring-tools/Help-e-book-creators/HTML-Help-Workshop.shtml	To compile html help document into .chm file and online help	Free software (Microsoft)
Inno setup	5.2.4-dev	http://www.jrsoftware.org/isinfo.php	To convert executable file into windows installable file format	Free software

APPENDIX C: EVALUATION QUESTIONNAIRE

The questionnaire used to evaluate and validate the CDST is shown in Table C.1. The evaluators rate the tool with respect to each criterion in the questionnaire as: 5 = Outstanding, 4 = Good, 3 = Satisfactory, 2 = Poor, and 1 = Unsatisfactory.

Table C.1 CDST evaluation questionnaire

No.	**Evaluation Criteria**	**5**	**4**	**3**	**2**	**1**
1	All the steps in conceptual design process are covered in a comprehensive manner					
2	The program acknowledges input					
3	The program supports decision making by helping the users generate ideas, obtain necessary information, and evaluate alternatives					
4	The user, not the program, controls the decision making					
5	The organization of the user interface is clear, logical, and effective, making it easy for the intended user to understand					
6	The user can easily start and exit the program					
7	The program is reliable in normal use. Software is bug free					
8	Using the program contributes to the user's creativity by initiating ideas					
9	The program can be a useful resource in academia to teach conceptual design process					
10	The program can be used as means to preserve experts knowledge for future use					
11	The structure of the program demonstrates that conceptual design can be computer assisted					
12	The program achieves its purpose					