

A FRAMEWORK FOR FUNCTION SPECIFICATIONS-TO-CONCEPTUAL FORM  
TRANSLATION TOOL IN FUNCTION-ORIENTED MECHANICAL DESIGN SYSTEMS

by

Amer Mohammad Momani

B. S. in Mechanical Engineering, Jordan University of Science and Technology, Jordan, 1995

M. S. in Mechanical Engineering, Jordan University of Science and Technology, Jordan, 1998

Submitted to the Graduate Faculty of  
School of Engineering in partial fulfillment  
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2005

UNIVERSITY OF PITTSBURGH  
SCHOOL OF ENGINEERING

This dissertation was presented

by

Amer Mohammad Momani

It was defended on

June 7th, 2005

and approved by

Dr. Bopaya Bidanda, Professor, Industrial Engineering

Dr. Mary Besterfield-Sacre, Associate Professor, Industrial Engineering

Dr. Michael Lovell, Associate Professor, Industrial & Mechanical Engineering

Dr. Meng-En Wang, Assistant Professor, Industrial Engineering

Dr. Bartholomew O. Nnaji, Professor, Industrial Engineering  
Dissertation Director

# A FRAMEWORK FOR FUNCTION SPECIFICATIONS-TO-CONCEPTUAL FORM TRANSLATION TOOL IN FUNCTION-ORIENTED MECHANICAL DESIGN SYSTEMS

Amer Mohammad Momani, PhD

University of Pittsburgh, 2005

Design is functionality driven. All products and parts have some intended reason behind their existence. Although computer aided systems have made considerable advances in capturing and representing geometrical shape, not much progress has been made in capturing and modeling product functionality and its physical realization. This research proposes a methodology to assist designers during the first stages of design. This methodology provides a framework to help the designer translate functional specifications into conceptual forms. This research develops a translation tool to model functionality and to carry out conceptual design with the aid of the computer. This tool serves as a bridge between the conceptual design phase and the detailed design phase of a product.

The translation tool developed in this research supports the conceptual design phase by providing a functional data model, a function server model, and a conceptual product model. The functional model includes the use of operands and relations to define and capture product functionality. The function server model represents the physical realization of the specified functions. The conceptual product model organizes and documents the product information in both the functional and the physical domains. The knowledge base for the function servers is stored in a function driven database. This database allows the designer to view design possibilities that may never have occurred to them.

Models provided in this work have been implemented as a relational database system by using MySQL. A web-based graphic user interface is developed with PHP to provide an interactive environment for modeling and for searching the function driven database. Propagation of functional and physical information to downstream design activities has been enabled by the use of the XML data format. The models and concepts developed in this research are validated through a case study of a realistic mechanical device.

#### DESCRIPTORS

Composite Function Server	Function-Oriented Design
Conceptual Design	Function-Oriented Design
Conceptual Product Model	MySQL
Function Server	Operand
Function Server Model	PHP
Function Specifications	Primitive Function Server
Functional Data Model	Translation Tool
Functional Feature	Web-Based Design
Functionality Modeling	XML

## TABLE OF CONTENTS

ACKNOWLEDGMENTS .....	xiv
1.0 INTRODUCTION .....	1
1.1 PROBLEM STATEMENT .....	6
1.2 RESEARCH OBJECTIVES .....	8
1.3 CONTRIBUTIONS .....	9
1.4 METHODOLOGY .....	11
1.5 RESEARCH ORGANIZATION .....	13
2.0 BACKGROUND AND LITERATURE REVIEW .....	15
2.1 ENGINEERING DESIGN PROCESS .....	15
2.1.1 Design Process .....	16
2.1.2 Design Axioms.....	20
2.2 FUNCTION-DRIVEN DESIGN .....	21
2.2.1 Types of Mechanical Function.....	23
2.2.2 Classification of Fundamental Mechanical Functions.....	24
2.2.3 The Function Analysis Method.....	25
2.2.4 Functionality Modeling.....	30
2.2.5 Function-to-Form.....	36
2.2.6 Computer-Aided Conceptual Design.....	44
2.3 FEATURES .....	46
2.3.1 Definitions of Features.....	46

2.3.2	Feature-Based Design .....	48
2.3.3	Features as Integrating Keys Linking Design and Manufacturing .....	49
2.4	PRODUCT MODELING.....	50
2.4.1	Types of Product Model.....	50
2.4.2	Model Standardization Using STEP .....	53
2.4.3	Product Modeling to Support the Design Process .....	54
2.5	PRODUCT DESIGN AND SPATIAL RELATIONSHIPS .....	57
3.0	OVERVIEW OF FUNCTION-TO-CONCEPTUAL FORM TRANSLATION MODEL... ..	59
3.1	SCOPE OF WORK.....	60
3.2	FUNCTIONALITY MODEL .....	61
3.2.1	Functionality Operations.....	61
3.2.2	Generic Functionality Model .....	64
3.3	FUNCTION SERVER MODEL.....	66
3.4	FUNCTION-TO-FUNCTION SERVER MAPPING.....	68
3.5	CONCEPTUAL PRODUCT MODEL .....	72
4.0	FUNCTIONAL DATA MODEL.....	74
4.1	FUNCTIONALITY OPERANDS .....	74
4.1.1	Solid material operand .....	76
4.1.2	Mechanical energy operand .....	79
4.2	FUNCTIONALITY RELATIONS AND STATES.....	82
4.3	MECHANICAL FUNCTION CLASSIFICATION BASED ON OPERANDS INTERACTION.....	85
5.0	FUNCTION SERVER REPRESENTATION AND MODELING.....	88
5.1	FUNCTION SERVER CLASSIFICATION.....	88

5.2	FUNCTION SERVER REPRESENTATION .....	91
6.0	IMPLEMENTATION AND CASE STUDY .....	98
6.1	DATA STRUCTURE OF THE TRANSLATION TOOL.....	98
6.1.1	Functionality Object Model .....	99
6.1.1.1	Function Operation Class.....	101
6.1.1.2	Operand Class .....	104
6.1.1.3	Relation Class .....	104
6.1.2	Function Server Object Model.....	104
6.1.3	Conceptual Product Object Model.....	107
6.2	TRANSLATION TOOL XML DATA FORMAT .....	108
6.2.1	XML Syntax.....	108
6.2.2	Functionality XML Data Format .....	109
6.2.3	Function Server XML Data Format.....	112
6.3	ARCHITECTURE OF THE FUNCTION TO CONCEPTUAL FORM TRANSLATION TOOL.....	116
6.4	WEB-BASED IMPLEMENTATION AND THE GRAPHIC USER INTERFACE .	118
6.4.1	Screen Presentation and General Usage .....	119
6.5	CASE STUDY AND VALIDATION .....	139
6.5.1	Procedure of Translating Function-to-Conceptual Form.....	139
6.5.2	Case Study: Toggle Clamp .....	142
6.5.3	Experimental Test.....	168
7.0	CONCLUSIONS AND FUTURE WORK.....	171
7.1	CONCLUSIONS.....	171
7.2	FUTURE WORK.....	173

APPENDIX A.....	175
ABSTRACT SHAPES OF FUNCTIONAL FEATURES.....	175
APPENDIX B.....	179
LIST OF FUNCTIONS USED IN FUNCTION DRIVEN DATABASE.....	179
APPENDIX C.....	180
FUNCTIONS OF FUNCTIONAL FEATURES.....	180
BIBLIOGRAPHY.....	182



## LIST OF TABLES

Table 2.1 Function representation (proposed by Roy et al. [18]).....	34
Table 4.1 Classification of basic mechanical functions.....	86
Table 5.1 Classification of primitive function servers.....	90
Table 6.1 Operand list for function operation 1.....	149
Table 6.2 Functionality attributes of solid material operand; solid A .....	150
Table 6.3 Functionality attributes of force energy operand; F1.....	150
Table 6.4 Functionality attributes of force energy operand; F2.....	151
Table 6.5 Functionality attributes of force energy operand; F3.....	151
Table 6.6 Functionality attributes of force energy operand; F4.....	151
Table 6.7 Functionality attributes of force energy operand; F5.....	152
Table 6.8 Functionality attributes of force energy operand; F6.....	152
Table 6.9 Relations in function operation 1 .....	154
Table 6.10 Description of the relation between solid A and F1 in function operation 1 .....	154
Table 6.11 Description of the relation between solid A and F2 in function operation 1 .....	155
Table 6.12 Description of the relation between solid A and F3 in function operation 1 .....	155
Table 6.13 Description of the relation between solid A and F4 in function operation 1 .....	156
Table 6.14 Description of the relation between solid A and F5 in function operation 1 .....	156
Table 6.15 Description of the relation between solid A and F6 in function operation 1 .....	157
Table 6.16 Equilibrium relation between force operands (F1, F2, F3, F4, F5, F6).....	157
Table 6.17 Sub-functions used to search with, and solutions.....	162

Table 6.18 Description of Function server model for solution 1 (Rode-1) .....	165
Table 6.19 The recorded design period times .....	169
Table A.7.1 Abstract shapes of functional features .....	175
Table B.1 List of functions and synonyms .....	179
Table C.1 Functional features and their possible functions in generic domain .....	181

## LIST OF FIGURES

Figure 1.1 Opportunity in early design stage.....	2
Figure 1.2 Flow diagram of function specifications-to-conceptual form translation tool .....	13
Figure 2.1 Design process phases .....	18
Figure 2.2 Design as the mapping from functional space to physical space .....	19
Figure 2.3 The black box system model.....	26
Figure 2.4 A transparent box model .....	28
Figure 2.5 Overall function and sub-functions of a testing machine .....	29
Figure 2.6 Process of generalized mapping .....	40
Figure 2.7 Functional feature “frictional connection” and its variation .....	42
Figure 2.8 Types of spatial relationships: (a) against, (b) parallel-offset, (c) include-angle, (d) parax-offset, (e) aligned, (f) incline-offset (adapted from Muogboh [29]).....	58
Figure 3.1 Function driven design cycle.....	62
Figure 3.2 Example of mechanical functionality (support of a shaft) .....	63
Figure 3.3 Function/function server decomposition structure .....	69
Figure 3.4 Function-to-function server mapping process .....	70
Figure 4.1 Components of a functionality operation (operands and relations).....	74
Figure 4.2 Friction functionality operations showing the interaction between functionality .....	75
Figure 4.3 Force-to-torque transformation operation .....	75
Figure 4.4 Material performance requirements .....	79
Figure 4.5 Example of hold functionality operation.....	83

Figure 4.6 Example of unfasten a bolt.....	85
Figure 5.1 Function server types.....	88
Figure 5.2 Example of compound feature.....	91
Figure 5.3 Example of abstract shapes of some functional features: (a) wall, (b) boss, (c) slot, (d) rod.....	93
Figure 5.4 Engineering materials.....	94
Figure 5.5 Manufacturing processes.....	95
Figure 6.1 Data structure of functional model.....	100
Figure 6.2 Example of function decomposition to illustrate the function address attribute.....	103
Figure 6.3 Data structure for function server.....	105
Figure 6.4 Data structure of the conceptual product model.....	107
Figure 6.5 The translation tool architecture.....	116
Figure 6.6 Beginning Screen.....	121
Figure 6.7 Login Screen.....	122
Figure 6.8 Main Screen.....	123
Figure 6.9 Function list.....	124
Figure 6.10 Function driven database.....	125
Figure 6.11 Add/Modify products.....	127
Figure 6.12 Choosing add/modify product functions or add/modify product function servers..	128
Figure 6.13 Add/Modify product functions (function decomposition).....	129
Figure 6.14 Add/Modify functionality model components (operands, relations, constraints)...	130
Figure 6.15 Add function operands.....	131
Figure 6.16 Operands attributes.....	132
Figure 6.17 Add function relations.....	133

Figure 6.18 Add function constraints.....	134
Figure 6.19 Search results.....	135
Figure 6.20 Product function servers (solutions).....	136
Figure 6.21 Components of product function server model (material, shape, manufacturing info, working condition, and interface).....	137
Figure 6.22 Material type for product function server.....	138
Figure 6.23 Translating function-to- conceptual form flow diagram .....	141
Figure 6.24 Function Decomposition for toggle clamp .....	144
Figure 6.25 Topology structure of solid operand (solid A) in function operation 1 .....	146
Figure 6.26 Force operands (F1, F2, F3, F4, F5, F6) interaction with solid A: (a) initial state. (b) final state.....	147
Figure 6.27 Adding toggle clamp as a new product .....	158
Figure 6.28 Function decomposition structure of the toggle clamp .....	159
Figure 6.29 Add/Modify function model components (operands, relations, and constraints) of the toggle clamp.....	160
Figure 6.30 Search process results.....	161
Figure 6.31 Interface of function server rod-1 with other function servers.....	164
Figure 6.32 User page to add/modify function server model .....	164
Figure 6.33 Conceptual form of toggle clamp's handle .....	166
Figure 6.34 Conceptual forms of toggle clamp components .....	167
Figure 6.35 The conceptual form of the toggle clamp assembly.....	167
Figure 6.36 Function server tree for toggle clamp.....	168

## ACKNOWLEDGMENTS

First, all praises and glory are due to Allah (God) for all the bounty and guidance granted to me. I would like to thank him for giving me the strength, energy, and intellect to complete my doctoral journey.

I would like to express my sincere appreciation to my advisor Professor Bartholomew Nnaji for his invaluable contributions and direction during my time as a graduate student. I thank him for his excellent instruction, guidance, advice, friendship, and support throughout this dissertation. I also would like to extend my gratitude to my committee members, Dr. Mary Besterfield-Sacre, Dr. Michael Lovell, Dr. Meng-En Wang, and Professor Bopaya Bidanda for accepting to serve on my committee and for their valuable suggestions.

I would also like to express my appreciation to the wonderful faculty members and staff of the Department of Industrial Engineering at the University of Pittsburgh, for their support, understanding, and kindness throughout the course of my doctoral studies. My special thanks and appreciation also go to all of the Automation and Robotics Laboratory members for their friendship and valuable comments. Invaluable technical advice and support was provided by my friend Sherif Khattab and I thank him for all his help.

I would like to express my thanks to my family members. My deep love and appreciation go to my parents for life long and endless love, support, prayers, and continues sacrifices. They taught me the value of education and determination. My warmest thanks go to my dear wife Suhad for her love, understanding, and continuous encouragement throughout my doctoral study. I would like to express my sincere love to my kids Obaidah and Sarah for making my life

beautiful. Special thanks go to my brothers and sisters for their love and encouragements. I dedicate this dissertation to my family. May Allah bless them all.

## **1.0 INTRODUCTION**

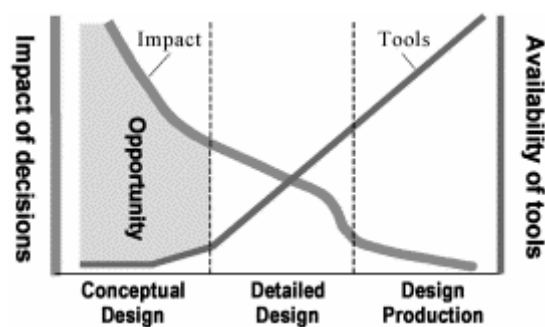
The technological advances of the last few years in the field of product development have lent a greater intensity to the pursuit of approaches that reduce the cost and time of product development, enhance the quality of the product, and help designers be more creative. Companies have to offer customers better and cheaper products with more functionality at shorter intervals than their competitors. This necessitates a decrease in development time that, in turn, creates a need to utilize resources more efficiently, be more flexible (both when it comes to work processes and products developed), and to create and communicate knowledge/experiences in a way that supports ongoing and future design activities. Currently, computer-aided systems and software have concentrated on the capture and representation of geometrical shape and technical information as opposed to providing support for the earlier stages of product design, such as capturing and modeling product functionality and its physical realization. This is because knowledge of the design requirements and constraints during this early phase of a product's life cycle is usually imprecise and incomplete, making it difficult to utilize computer-based systems or prototypes.

In this research, a methodology is proposed to assist designers during the first stages of design. This methodology contains a framework to help the designer translate the function specifications into conceptual forms. The objective is not to construct automatically the shape but to automate a certain number of heavy and tiresome tasks and to assist designers during collaborative design. This framework supports the conceptual design phase by providing a conceptual product model, which represents and organizes product information in both the



functional and the physical domains in an object-oriented manner. This makes it possible to retain design intentions and allows designers from different backgrounds with various interests to access the design information and to communicate with one another easily. This framework also serves as a bridge between the conceptual design phase and the detailed design phase of a product.

Competitive time scale forces designers to make critical decisions without exploring alternative strategies in-depth. The negative impacts of decisions made in the early design stages may not be evident on downstream activities until a product model starts to get fairly complete. As shown in Figure 1.1, the impact of design decisions is initially very high and declines steeply as the design matures <sup>[1]</sup>. A great opportunity exists at the preliminary design stage to make a few reading decisions. The concept generated at this stage affects the basic shape generation and material selection of the product concerned. In the subsequent phase of detailed design, it becomes extremely difficult, or even impossible, to compensate for or to correct the shortcomings of a poor design concept formulated in the conceptual design phase <sup>[2,3]</sup>.



**Figure 1.1 Opportunity in early design stage**

It is now well recognized that a conceptual design model should incorporate more information than just a description of the physical structure of the desired product. A conceptual design model should include the product's constituent components and relationships. Functions, as a description of the purpose of the physical structure, should be explicitly represented also [4]. Functions play a key role during the upstream stages of design, just as geometry does for the downstream stages. Hence, research in this area requires a comprehensive understanding of function and its effective representations [5].

Functional modeling of a mechanical product during the conceptual design phase can be regarded as a process of establishing a functional model of a desired but physically indefinite product. This is different from the functional modeling of an existing physical device, because the object being modeled has not yet been physically defined. The major issue here is how to represent and manipulate functions so that they can be associated with physical models later on in the design process. A closely related issue is how to represent functions in a computer program so that they can be easily defined, retrieved, edited, and manipulated by a designer in a specific design situation. Clearly, any function representation scheme should directly represent a product's functions and be amenable to computer implementation.

This research introduces a functional data model, which allows the functions of a product to be represented in a form that can be used by a number of engineering applications. The functional model is used to maintain information about functions and, at the same time, to initiate the search for solutions.

The functions of the product can be broken down into a number of sub-functions. Sub-functions are the individual operations that a product performs. These sub-functions are usually expressed in schematic form to explain in greater detail how the product operates. This creates a

language for the product that is functionally driven. This, then, allows thinking and brainstorming to occur around each sub-function in order to determine how that function will be accomplished. Some of the functional elements of an inkjet printer are store papers, store ink, deliver ink to paper, increment paper, and process print information. Each of these functions says what the function must do, but not how it is to be accomplished.

During conceptual design, the decomposition of function reaches a stage where the designer begins to map the lower level of functions to physical elements. A physical element refers to a complete mechanical product, its components, or the component's features, which implement the overall function and sub-functions of the product. These physical elements become more defined as the product is designed. Some physical elements depend upon the original product concept and some of the physical elements are not defined until the detailed design phase. Each physical element usually has at least one function associated with it, and sometimes there are multiple functions associated with each physical element. Some of the physical elements of a printer are chassis, the carriage, the printed circuit board, motors, and paper trays <sup>[6]</sup>. One example of a physical element that has multiple functions is the chassis. It not only supports the product but also defines the aesthetics of the product and helps direct airflow through the printer to cool the electronics. The decomposition of function and mapping to form is not a linear process. Functional decomposition may be modified, added, or deleted. The mapped form may also be modified, added, or deleted. Therefore, a tool to help the designer to choose and evaluate forms is very essential.

The ability to present a wide selection of design to the designer is of great importance. During the design process, the designer makes use of many different technologies. It is impossible for the designer engineer to be an expert in all the domain fields such as casting,

machining, sheet metal forming, and plastics, as well as in the usage of the variety of components that are found within the multitude of design catalogs. To be an expert in only one of these domain fields requires years of experience, therefore, a designer cannot be expected to know the details of domain fields outside their expertise. A solution library brings knowledge and information to the designer when needed. Part of this research project includes the development of the basic requirements for a function driven solution library, which will help the designer during the conceptual phase. The knowledge of the features, components, and assemblies will be contained within this function driven database, so that the designer need not be an expert in every domain field. This will allow designers to view design possibilities that may never have occurred to them.

Rapid product development and increasingly detailed documentation require more complete design development tools. No longer can the designer spend his or her time on a cumbersome CAD system, burdened with the many menus and time-consuming design development. The designer requires speed, reliability, consistency, and readily available solutions in product development. The designs that are created during the conceptual design phase need to be completely documented, along with the reasons why they were developed. This research develops a conceptual product model to document the product information in both the physical and the functional domains and to support the conceptual design phase. The information should be organized in a way that is easy for the users to access, which means that the information structure should bear close resemblance to the user's thought patterns.

## 1.1 PROBLEM STATEMENT

Development in the area of computer-aided design (CAD) has mainly been focused on geometric modeling. Most commercial CAD systems are successful in dealing with geometric information rather than aspects of the design process, such as function analysis, concept generation, and exploration. However, a study conducted by Lotter <sup>[7]</sup> indicates that as much as 75% of the cost of a product occurs during the design phase. The majority of design decisions are made during the conceptual design phase, where few representations and reasoning tools are available for support. More importantly, a poorly conceived design concept can never be compensated for by a high-quality detailed design. Consequently, current CAD systems cannot meet the requirements for complicated engineering design.

Most CAD systems lack the ability to advise or support designers at a functional level. Current CAD systems were developed to provide access to low-level geometry and topology, with some reasoning about the characteristics of the design provided. The technological hurdle that must be overcome in the development of a function-based CAD system is the modeling of the functions of associated features, components, and sub-assemblies. Function-oriented design aims at assisting the designer during the whole design process, from specifications to manufacturing. Usually, the designer only knows the need or the function of the product. Therefore, it is very important to utilize and record the functional data effectively and efficiently. This will help the designer to convey his or her intent and to translate these functional specifications into an acceptable concept design. Thus, this will also support the conceptual design phase. The representation of the design's functionality has uses beyond the design process. Once the design has been developed, a functional data model can be used to provide

information during life cycle processes, such as analysis, manufacturing, and product support, which may be lost if only geometric representations are available.

One of the key stages of the function-oriented design is the function-to-shape translation. Shape is today the main representation of a product, even though the current trend is to move away from geometry in order to add high-level information. However, geometry constitutes the starting point of many activities, such as mechanical optimization and kinematics simulation [8]. The function-to-shape translation remains one of the most important activities of the design process and, up till now, it was done manually. This activity is very important both for the choices that are made and for the quantity of work that it represents. The functional description of a product is only sufficient at an abstract stage of design. At some point, the designer must choose the forms that will enact the functions chosen. At this point, a tool to help the designer choose forms and evaluate them would be very useful.

Although considerable advances have been made in the development of function-to-form mapping systems, there has been little progress on systems that really aid the designer in converting functional specifications to concept design. The creation of rough realizations of physical forms that are both completed and innovative, from loosely stated design goals and required functions, is still not possible. Moreover, the relation between form and function is not completely understood. One difficulty is that function can be a composite result of many interacting sub-functions. A second difficulty is that there is no unique mapping between function and form (i.e., the same function could be accomplished by several different forms and a given form could be used to perform different functions). Another problem is that the product function is context dependent. This means a physical element can perform different functions in different situations. A hole for example may be used for alignment, for assembly, for stress

relief, or for support. So the use of a physical element depends on the context in which the designer wishes to employ it. All of these issues require the concept of functionality to be tied more closely to the design object.

The designs that are created, after function-to-form mapping, need to be completely documented, which includes what was developed and the reasons why it was developed. This large volume of information about the complex inter-relationships of a new product needs to be efficiently managed and to be represented clearly. As a result, when a feature, component, or assembly is modified the designer can then be aware of which functions and which parts of the specifications may be affected. Therefore, there is a need for a system that will adequately model and propagate product information during the conceptual design phase. There is also a need for a system that retains design intentions and allows designers from different backgrounds with various interests to access the design information at early stages and to communicate with one another easily.

## **1.2 RESEARCH OBJECTIVES**

The primary objective of this research is to develop, for mechanically engineered products, a framework for translating functional specifications into conceptual forms as automatically as possible. This translation tool bridges the gap between the functional and the physical realm and supports the conceptual design phase. The translation tool also provides a means for designers, who only know the function of the product, to access and extract the necessary solutions from a function driven database. The translation tool targets the functionality and the relationships possessed by an object (feature, component, or sub-assembly) as the primary search source in finding solution possibilities. This allows the designer to have access to

a great variety of design solutions and can enhance his or her productivity. Finally, the translation tool contains the ability to document the results of the translation process in a model, including what was developed and why.

The primary objectives of this project are realized through the development of the following research activities:

- Functional data model: This model captures and maintains the function-related information during the conceptual design phase and helps the designer to search for possible physical solutions.
- Function server model: This model represents and captures the information about the physical embodiments associated with the functional requirements. It also founds the basic structure of a function driven database.
- Searching mechanism: This mechanism uses function-form relations to search the function driven database for solutions (i.e., function servers).
- Conceptual product modeling: This model documents the searching results and can be used for the propagation of product information in a transparent manner. This model supports the conceptual design phase by representing and organizing product information in both the functional and the physical domains.

### **1.3 CONTRIBUTIONS**

A function-to-conceptual form translation methodology together with associated computer tools is developed in this work to support the linkage of design functions with the physical embodiments used to realize the functions. Therefore, the integration between function



specifications, conceptualization, and detailed design process can be promoted. The anticipated contributions of this work are listed in the following points:

- Development of a function-to-conceptual form translation tool to support function driven design of mechanical products.
- Provision of functionality modeling methodology that supports the definition and representation of product functionality, and helps to cross the boundary and enter the physical realm.
- Introduction of a new classification scheme for the mechanical functions based on the interaction between functional elements (material and energy).
- Development of a basic structure for function driven database that contains the design knowledge about possible physical embodiments to function specifications. This development is made possible by the provision of a function driven database prototype that contains the design knowledge for the most primitive physical embodiment-functional feature. The implementation utilizes the functionality a feature possesses to obtain solutions.
- Provision of the function server model that supports the definition and representation of physical solutions, and introduces the conceptual realization of a product form.
- Provision of the conceptual product model that enables the documentation and organization of the conceptual product information in both functional and physical domains.
- Provision of a means for propagating and exchange conceptual product information with the detailed design stage. This provision is made possible by the description of functional and physical information as an extensible markup language (XML) data file.

- Provision of an interface between conceptual design and detailed design phase of a product.

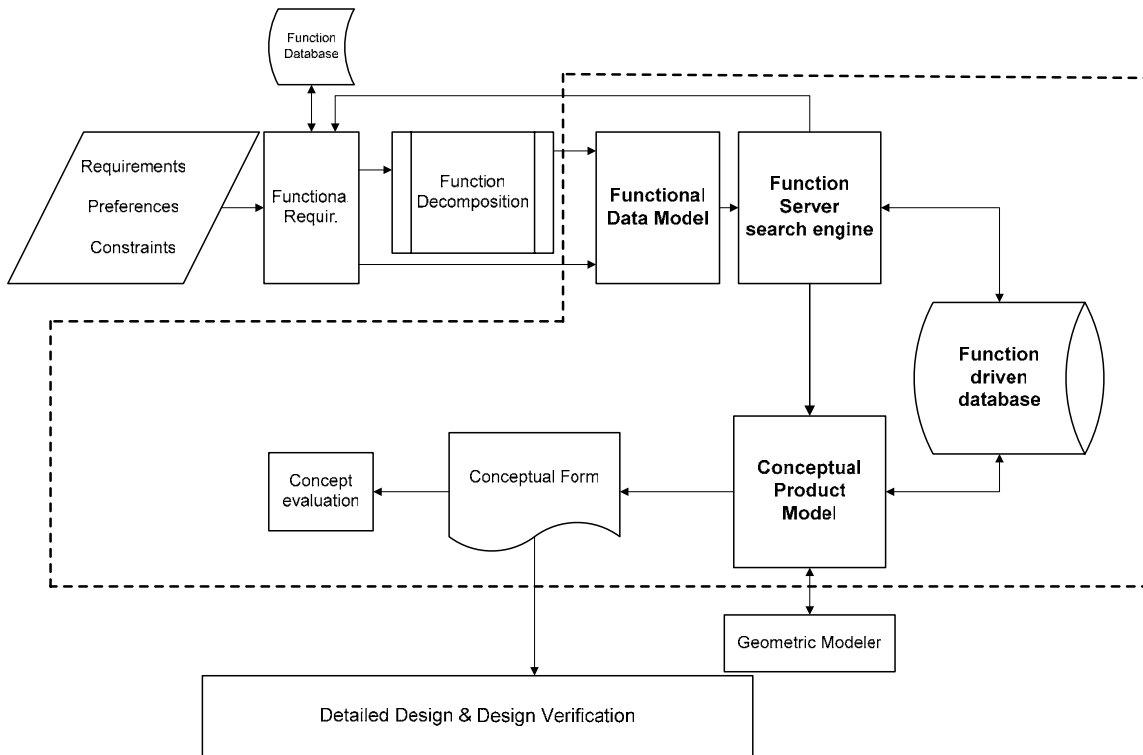
## 1.4 METHODOLOGY

During the conceptual design phase, the designer adds information to a design that has very low information content. At this very abstract stage, the design team has only a primary objective function and a handful of constraints and criteria. Information must be added to the design incrementally, slowly making it more concrete with each step. One typical approach is to decompose the objective function into sub-functions. Then, the design must cross the function-form boundary and enter the embodiment stage.

In this research, a framework for translating function specifications into conceptual forms is developed as a support for the conceptual design phase. This translation tool translates the function specifications into conceptual form—first in a very imprecise format and then more and more precisely. It aims at transforming an abstract representation of a product into a more concrete representation. The flow diagram for this translation tool, which shows the workings of the system, is depicted in Figure 1.2. The initial information obtained during the conceptual design phase of product development is from the customer's needs and preferences. These are in turn mapped into engineering specifications and functional requirements. The functions are then broken down by the designer but without developing the actual form of the product. The function structure takes the form of verb-noun pairs. The verb indicates the function of the object and the noun represents the physical effect that is manipulated by the verb: material or energy.

The function-oriented problem definition and detailed function decomposition are converted into a functionality-based data structure called a functional data model. This data

structure is used for maintaining the relationship elements between the function objects, as well as for initiating the search process for possible solutions in physical domain. The next step is to search for solutions. The sub-functions to be satisfied are passed, along with pertinent constraints, to a function driven database, which contains information about function servers in the physical domain. The function server can be any physical element such as assembly, component, or feature. Solutions are then searched for with the sub-function or sub-functions, and, if solutions are found, they will be presented to the designer for evaluation. Once the designer has decided on a solution, it is then passed with all related information to the product's function server database (product tree). If no solution is found, the system will ask the designer to return to the engineering specifications and functional requirements and consider any possible modifications. After the designer has finished searching and building the product tree, the new product information in both the functional and the physical domain is represented and organized into a model known as a conceptual product model.



**Figure 1.2 Flow diagram of function specifications-to-conceptual form translation tool**

## 1.5 RESEARCH ORGANIZATION

In this dissertation, chapter 2 presents the research background and literature review of relevant research areas and of important concepts related to this work. This includes previous work in functionality modeling, function-to-form mapping, and product modeling. Chapter 3 provides an overview of the function-to-conceptual form translation model. It includes the representation and modeling approach for the translation tool components developed in this dissertation. Chapter 4 focuses on functionality data modeling and representation. It also introduces a new function classification scheme. Chapter 5 focuses on function server representation and modeling. Chapter 6 discusses the computer implementation and validation.

Here, the developed models and concepts are tested and validated using a case study. The data structures of the developed models are also discussed. Chapter 7 concludes this dissertation and recommends areas of future research.

## **2.0 BACKGROUND AND LITERATURE REVIEW**

### **2.1 ENGINEERING DESIGN PROCESS**

Product design is the process of transforming relatively vague goals into specific information needed to manufacture the product, while meeting certain performance criteria and resource limitations. This information is in the form of drawings, computer-aided design (CAD) data, notes, instructions, and so forth. Design problems normally originate as some form of problem statement provided to the designer by someone else, the client, or the company management. These problem statements, normally called a “design brief,” can vary widely in their form and content. At the start of design process, the designer is usually faced with a very poorly defined problem, yet he or she has to come up with a well-defined solution. The designer’s difficulties are therefore two-fold: understanding the problem and finding a solution. These two complementary aspects of design (problem and solution) have to be developed side-by-side. The designer makes a solution proposal and uses that to help understand what the problem really is and what appropriate solutions might be. The very first conceptualizations and representations of the problem and solutions are therefore critical to the kind of search and other procedures that follow, and so to the final solution that is designed.

CAD technology provides effective support for activities undertaken during the downstream stages of design. These activities include geometric modeling, parametric design, finite element analysis, as well as many other geometry-related applications. However, it provides little support for activities undertaken during the upstream stages of design (e.g., conceptual development in which the designer works with the functional requirements of parts

<sup>[5]</sup>. The main reasons for this little support for the conceptual design phase can be summarized in the following points <sup>[9]</sup>:

1. CAD systems have concentrated on capturing and representing geometric shape, as opposed to providing support for conception.
2. Systems that attempt to provide conceptual design support are based on little explicit relation to function.
3. CAD systems require a detail of representation that is too restrictive for conceptual design.

In the future, more intelligent CAD systems will have to handle more than just geometric information. From the point of view of design, these systems have to process functional information as units together with geometrical data. Consequently, in order to develop future CAD systems, an extensive understanding of the relationship between function and shape is absolutely necessary <sup>[10]</sup>. The shape is still the main representation of a product, even though the current trend is to remove geometry from its central position in order to add high-level information. The function-to-shape translation appears to be one of the most important activities of the design process and up till now done manually. This activity is very important both for the choices that are made and for the quantity of work that it represents <sup>[8]</sup>.

### **2.1.1 Design Process**

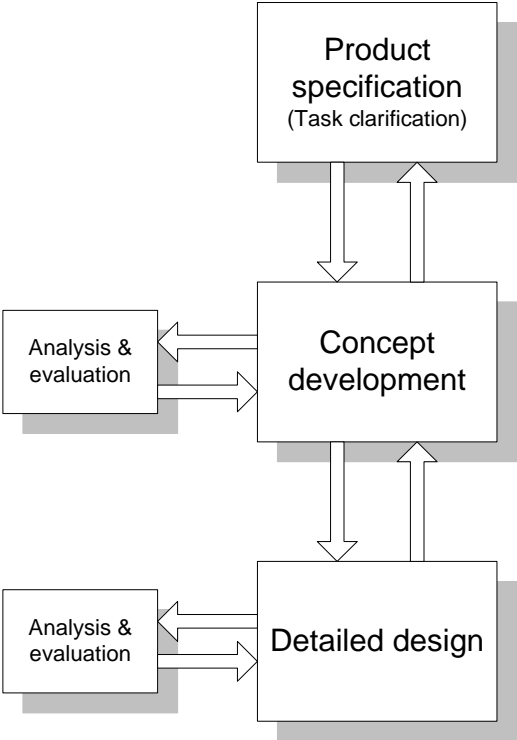
The design process can be original, adaptive, or variant <sup>[2]</sup>. In original design, new tasks and problems incorporate new solution principles. These can be realized either by selecting and combining known principles and technology or by inventing completely new technology. The term “original design” is also used when existing or slightly changed tasks are solved using new

solution principles. Original designs usually proceed through all design phases, depend on physical and process fundamentals, and require careful technical and economic analyses of the task. Original designs can involve the whole product or just assemblies or components. In adaptive design, one keeps to known and established solution principles and adapts the embodiment to changed requirements. It may be necessary to undertake original designs of individual assemblies or components. The geometrical, analytical (e.g., strength, stiffness), production, and material issues are very important in this type of design. Finally, in variant design the size and arrangements of parts and assemblies are varied within the limits set by previously designed product structures. Variant design requires an original design effort only once. It includes designs in which only the dimensions of individual parts are changed to meet a specific order. The boundaries between the three types of design cannot be defined precisely all the time, so they must be considered only as a broad classification.

The product design process is an iterative, complex, decision-making engineering process. It usually starts with the identification of a need, proceeds through a sequence of activities to seek an optimal solution to the problem, and ends with a detailed description of the product <sup>[3]</sup>. Generally, a design process consists of three phases (see Figure 2.1). The first phase is product design specification, in which information about the product and initial need is collected and defined in precise, yet neutral, terms. This phase possesses the problem statement consisting of: (a) a general statement of the design problem; (b) the limitations and constraints upon the solution (e.g., customer, engineering, government, design code requirements, completion dates); and (c) the creation of excellence toward which the designer is working <sup>[11]</sup>. These three elements are considered the “clarification of the task,” or the initial gathering of



information about the design requirements needed to describe the solution, pinpoint the known constraints, and describe the initial need of a product.



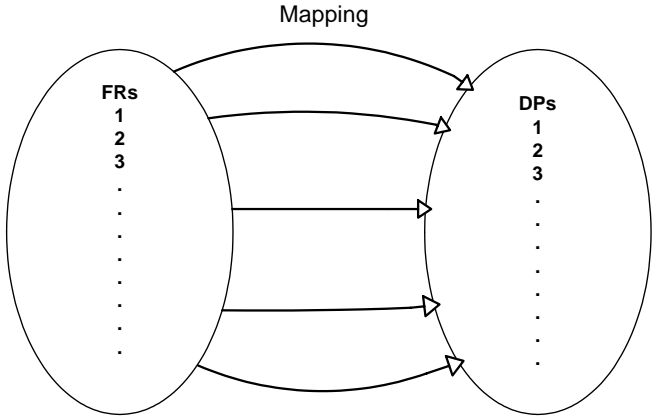
**Figure 2.1 Design process phases**

The second phase is the concept development, in which the primary concern is the generation of a physical solution to meet the design specifications. The concept development comprises the conceptual design, which establishes the functional structure, investigates potential solution concepts, and combines them into possible design alternatives; and embodiment design, which determines the layout design and then analyzes, evaluates, refines, and develops the solutions from the conceptual stage. The final phase is the detailed design. In this phase, final

decisions about dimensions and about the arrangement and shapes of individual components and materials are made with due consideration given to the manufacturing function.

The model described above represents what is widely considered to be “good design practice.” It is based on a design process that moves from the establishment of design requirements to the generation of a solution. It recognizes that the sequence of requirement specification, candidate solution generation, analysis, and evaluation is a logical necessity. It further recognizes that design is necessarily a succession of stages (e.g., planning, conceptual design, embodiment design, detailed design) that must be carried out in an orderly sequence. The decision of a preceding stage is a more or less rigid constraint on the following process, with a substantial resource penalty for returning to a preceding stage.

The objective of the design is stated in the functional domain, whereas the physical solution is generated in the physical domain. Therefore, the design process involves relating the functional requirements (FR) of the functional domain to the design parameters (DP) of the physical domain. The DPs in the physical domain are chosen to satisfy the FRs in the functional domain (see Figure 2.2).



**Figure 2.2 Design as the mapping from functional space to physical space**

This mapping process is non-unique; therefore, more than one design may ensue from the generation of the DPs that satisfies the FRs. The determination of a good set of FRs from diffuse and often poorly defined perceived needs requires skill, extensive market study, and many iterations. The mapping process is an important step in the design process; therefore, the FRs should be clearly and concisely defined to represent the correct design problem or need. At the same time, the function-form relations should be clearly represented in a well-constructed model.

### 2.1.2 Design Axioms

Axiomatic design theory, introduced by Suh <sup>[12]</sup>, put forward two axioms: the independence axiom and the information axiom. These are stated in declarative form as follows:

- Axiom 1: The independence axiom: maintain the independence of FRs
- Axiom 2: The information axiom: minimize the information content

The axiomatic design may be characterized mathematically. Both FRs and DPs can be treated as vectors, with  $m$  and  $n$  components, respectively. The design process then involves choosing the right set of DPs to satisfy the given FRs, and may be expressed as:

$$\{\text{FR}\} = [\text{A}] \{\text{DP}\}$$

where,

$\{\text{FR}\}$ : the functional requirement vector,

$\{\text{DP}\}$ : the design parameter vector, and

$[\text{A}]$ : the design matrix.

So we can write any line in the vector above as:

$$FR_j = \sum_j A_{ij} DP_j$$

The design matrix  $[A]$  is of the form:

$$[A] = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix}$$

$A_{ij}$  element relates a component of the FR vector with another one in the DP vector, and can be represented as:

$$A_{ij} = \partial FR_i / \partial DP_j$$

The left-hand side of the design equation represents what we want in terms of design goals, and the right-hand side of the equation represents how we hope to satisfy the design parameters' DPs.

## 2.2 FUNCTION-DRIVEN DESIGN

The type of knowledge, its level of granularity, and the operations on the knowledge needed in engineering design vary throughout the design process <sup>[13]</sup>. However, some important information developed early in the design process (i.e., during conceptual design) needs to be maintained and accessible for the design engineer during the later stages of design. One of the more important types of information needed is the set of required functions for the design.

In engineering design, the end goal is the creation of an artifact, product, system, or process that performs a function or functions to fulfill customer needs <sup>[2,12,14,15,16]</sup>. Modeling a design at the functional level and mapping these functions to embodied solution concepts aid the designer throughout the design process in validating the design against the requirements. Moreover, direct mapping to form (i.e., geometric and topologic) may provide the structure for functional constraint satisfaction, propagation, and truth maintenance in the design.

Function is a critical aspect of a design, but has no clear, uniform, objective, and widely accepted definition, as pointed out by Umeda et al. <sup>[17]</sup>. Pahl and Beitz <sup>[2]</sup> defined function as the general input/output relation of a system whose purpose is to perform a task, typically stated in verb-object form. Cole <sup>[18]</sup> stated that functions are the actions a system must perform in response to its environment in order to achieve the mission or goals given to it. Tomiyama et al. <sup>[19]</sup> defined function as a description of behavior abstracted by humans through recognition of that behavior in order to utilize it. Stone et al. <sup>[20]</sup> defined function as a description of an operation to be performed by a device or artifact, expressed as the active verb of the sub-function.

The function definitions given in the design literature are diverse and even contradictory, but can be categorized according to three main viewpoints <sup>[5]</sup>:

1. System viewpoint: In this case, a function is viewed as a relationship between the input, the output, and the stated variables of a system. When a system transforms inputs to outputs, it exhibits a particular function.
2. Performance viewpoint: In this case, a function is viewed as an abstraction of physical behavior. For example, consider a mechanical product that performs a specified behavior in a specified situation (working conditions), and these achieve the same results. The set of behaviors defines a functional class, and the results are its function.
3. Designer viewpoint: In this case, a function is viewed as a description of the design intention (i.e., the intended purpose of a product).

A good definition of function should include all these viewpoints. In this research, the focus is on the mechanical product functions that can be produced by the product or by some of its components.

### 2.2.1 Types of Mechanical Function

The formal functional specification of a product is a communication from a designer to someone else. A designer may specify different functions for the same product in order to convey functional information to different groups of people. Thus, the type of function that is considered in this research is what we refer to as “performance function” (or “design function”). Other types of function can be identified by taking a functional view of the concurrent engineering technique (e.g., Design-for-X) <sup>[5]</sup>:

1. *Assembly function* defines the purpose of component features so they can be assembled easily and economically. For example, the function of a “chamfer” feature of a shaft might be “to facilitate mating contact between a shoulder on the shaft and bearing.”
2. *Manufacturing function* defines the purpose of a feature so that it can be made economically and within tolerance-specification limits. For example, a hole may be machined in a component to locate it during manufacturing.
3. *Marketing function* specifies the purpose of features to meet customer requirements that are not performance-related. These functions typically relate to the look and feel of a product; they are aesthetic in nature.
4. *Maintenance function* specifies the purpose of features that facilitate maintenance activities, but excludes functions related to assembly and disassembly. A grease nipple is an example of a part designed for maintenance reasons only. It allows lubricant to be applied to a bearing without stripping the complete assembly.

Apart from the above distinctions between function types, functions can also be classified in terms of other criteria. Functions can be differentiated as primary or secondary by categorizing them in terms of their importance in a design.

### 2.2.2 Classification of Fundamental Mechanical Functions

Few researchers have attempted classification of mechanical functions. Pahl and Beitz <sup>[2]</sup> suggested the elaboration of the function structure from generally valid sub-functions. The concept of generally valid sub-functions is, in effect, derived from a classification scheme in which five kinds of sub-functions are identified: “change” from *type* characteristics, “vary” from *magnitude* characteristic, “connect” from *number* characteristic, and “channel” from *place* characteristic, and “store” from *time* characteristic. Keuneke <sup>[21]</sup> classified general device functions into four types: “ToMake,” “ToMaintain,” “ToPrevent,” and “ToControl.” Other classification schemes have been described by a number of authors <sup>[20, 22, 23]</sup>.

Deng et al. <sup>[5]</sup> classified fundamental mechanical functions depending on previous efforts into the following categories:

1. Functions relating to supplying or storing energy or material (e.g., the functions of electric motor, spring, flying wheel, oil tank)
2. Functions relating to transmitting energy or material; this category can be further classified as:
  - Transmitting motion (e.g., the functions of the shaft, gear, belt, chain)
  - Transmitting force or moment; these functions usually couple with the above functions
  - 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)

4. Functions relating to changing the form or magnitude of energy or material, or physical quantities relating to energy; this category can be further classified as:
  - Changing form of energy, physical quantities relating to energy or material, such as the functions of an electric motor (changing electric energy to mechanical energy) or cam (changing rotational motion to linear motion)
  - Changing magnitude of physical quantities related to energy or flow of material, such as gear pair (changing magnitude of angular speed) or gate (changing flow of material)

### **2.2.3 The Function Analysis Method**

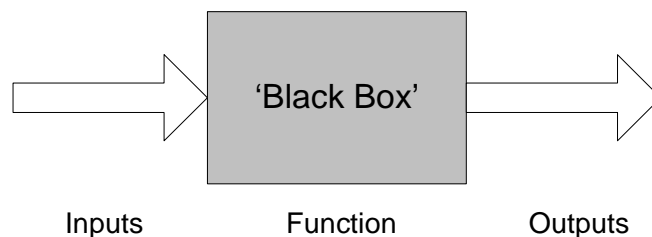
The aim of function analysis is to establish the functions required, decompose them into discrete tasks or sub-functions, and define the system boundary of a new design. The procedure is as follows <sup>[11]</sup>:

1. Express the overall function for the design in terms of the conversion of inputs into outputs by a black box.
2. Break down the overall function into a set of essential sub-functions. The sub-functions comprise all the tasks that have to be performed inside the black box.
3. Draw a block diagram showing the interactions between sub-functions. The black box is made transparent so the sub-functions and their interconnections are clarified.
4. Draw the system boundary. This boundary defines the functional limits for the product or device to be designed.
5. Search for appropriate components to perform the sub-functions and their interactions. Many alternative components may be capable of performing the identified functions.



Express the overall function in terms of the conversion of inputs into outputs

The starting point for the analysis method is to concentrate on what has to be achieved by a new design, and not on how it is to be achieved. The simplest and most basic way of expressing this is to represent the product or device to be designed as simply a black box that converts certain inputs into desired outputs. The black box contains all the functions that are necessary for converting the inputs into the outputs (see Figure 2.3).



**Figure 2.3 The black box system model**

It is preferable to try to make this overall function as broad as possible at first; it can be narrowed down later if necessary. It would be wrong to start with an unnecessarily limited overall function that restricts the range of possible solutions. The designer can make distinct contribution at this stage of the design process by asking the clients or users for definitions of the fundamental purpose of the product and asking about the required inputs and outputs (e.g., from where the inputs come, what the outputs are for, what the next stage of conversion is). This kind of questioning is known as “widening the system boundary.” The system boundary is the conceptual boundary used to define the function of the product. Often this boundary is defined too narrowly, with the result that only minor design changes can be made, rather than a radical rethinking of the design.

It is important to try to ensure that all the relevant inputs and outputs are listed. They can all usually be classified as flows of material, energy, or information, and these same classifications can be used to check if any input or output type has been omitted.

*Break down the overall function into a set of essential sub-functions*

Usually the conversion of the set of inputs into the set of outputs is a complex task inside the black box and has to be broken down into sub-tasks or sub-functions. The goal of this step is to refine the overall function statements as much as possible. The three reasons for doing this decomposition are as follows <sup>[14]</sup>:

1. The resulting decomposition controls the search for solutions to the design problem. Because concepts follow functions and products follow concepts, we must fully understand the function before wasting time generating products that solve the wrong problem.
2. The division into finer functional detail leads to a better understanding of the design problem. Although all this detail work sounds counter to creativity, most good ideas come from fully understanding the functional needs of the design problem.
3. Breaking down the functions may lead to the realization that components exist that can provide some of the functionality required.

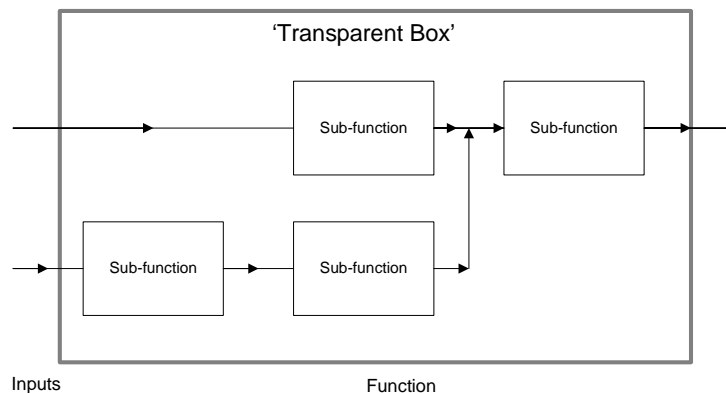
No truly objective, systematic way exists to do this; the analysis into sub-functions may depend on such factors as the kinds of components available for specific tasks, the necessary or preferred allocations of functions to the machine or to human operators, and the designer's experience.

In specifying sub-functions, it is helpful to ensure that they are all expressed in the same way. Each one should be a statement with a verb plus a noun (e.g., amplify signal, count items,

separate waste, reduce volume). Each sub-function has its own input(s) and output(s), and compatibility between these should be checked. Some auxiliary sub-functions may need to be added (e.g., remove waste), but these do not contribute to the overall function.

Draw a block diagram showing the interactions between sub-functions

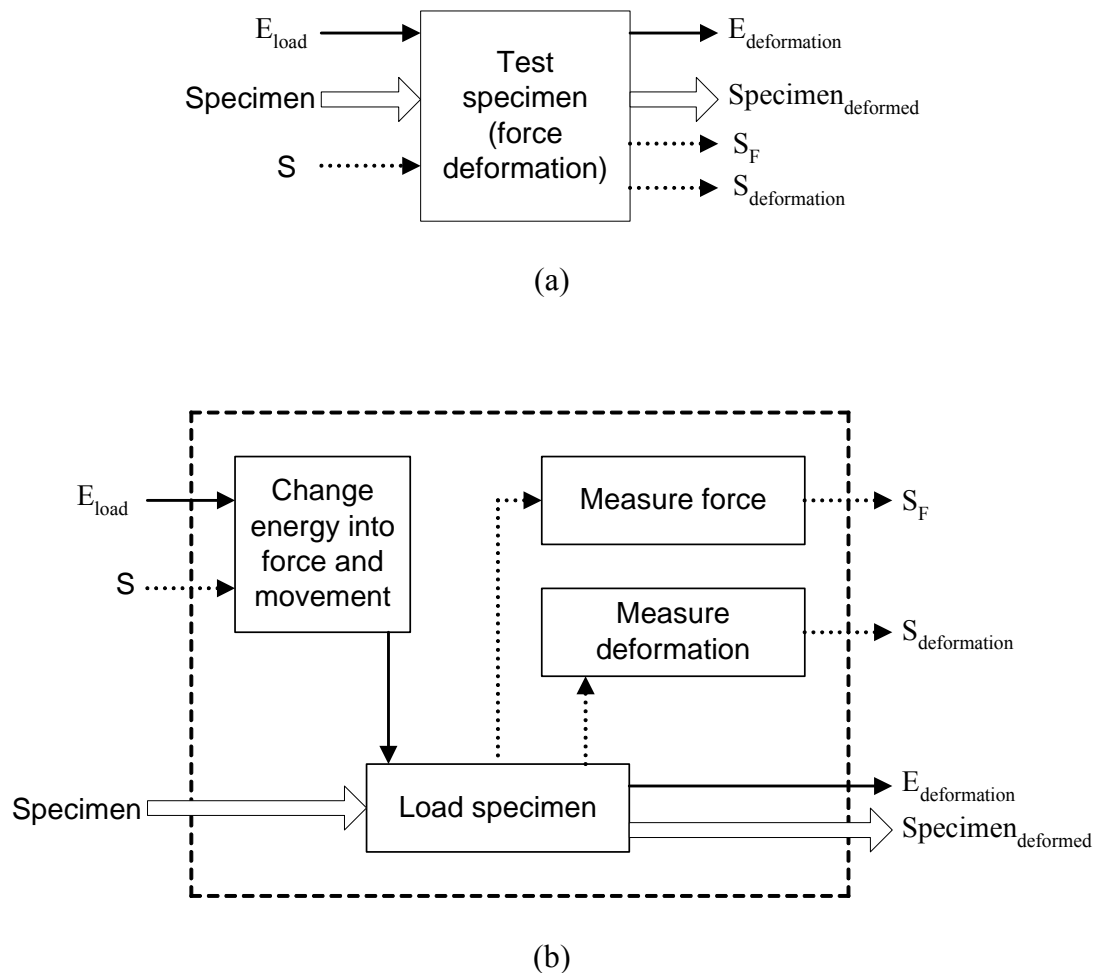
A block diagram consists of all the sub-functions, separately identified by enclosing them in boxes and by linking them together by their inputs and outputs to satisfy the overall function of the product being designed. In other words, the original black box of the overall function is redrawn as a transparent box, in which the necessary sub-functions and their links can be seen (see Figure 2.4).



**Figure 2.4 A transparent box model**

In drawing this diagram, it is necessary to decide how the internal inputs and outputs of the sub-functions should be linked together to make a feasible, working system. It may be necessary to juggle inputs and outputs, and perhaps redefine some sub-functions, so everything is connected together. It is useful to employ different conventions (i.e., different types of lines) to show the different types of input and output (i.e., flows of materials, energy, or information). An

example of functions and sub-functions as represented in a function converter black box model for a testing machine is provided in Figure 2.5. This illustrates the decomposition from the top-level function to lower levels for the refinement of energy, signal, and material [2].



**Figure 2.5 Overall function and sub-functions of a testing machine**

Draw the system boundary

In drawing the block diagram, it is necessary to make decisions about the precise extent and location of the system boundary. For example, the diagram can have no loose inputs or outputs, except those that come from or go outside the system boundary. It may be that the

boundary has to be narrowed again, following its earlier broadening during the consideration of inputs, outputs, and overall function. In order to define a feasible product, the boundary has to be drawn around a subset of the functions that have been identified. The designer may not have complete freedom in drawing the system boundary; it probably will be decided on the basis of management policy or client requirements. Usually, many different system boundaries can be drawn to define different products or solution types.

*Search for appropriate components to perform the sub-functions and their interactions*

If the sub-functions have been defined adequately and at an appropriate level, it should be possible to identify a suitable component for each sub-function. This identification of components depends on the nature of the product or device or more general system being designed.

#### **2.2.4 Functionality Modeling**

Technical system representations exist for different levels of abstraction. An engineering drawing represents the geometric aspects of a system; for increasingly complex systems, more abstract representations are needed (e.g., circuit diagrams, bond graphs, block diagrams, mathematical, or mechanical models). Models are useful tools to analyze, discuss, or design systems. Models are simplified abstract constructs used to predict the behavior of a system and to get a quantitative understanding of its operations and find possible critical points before the actual system is built <sup>[24]</sup>. Models can be catalogued into functional (mathematical) or structural (physical). Functional models try to represent how a system works, while structural models represent how the system is built. Both types of models have something in common: they only

reflect certain features of a real system—only those aspects intended to be relevant to the characteristic under study. What to include in a model must be considered carefully; including irrelevant details may make models complex, while oversimplified models may disregard important effects.

Functionality modeling provides an abstract, yet direct, method for understanding and representing an overall product or artifact function. Functionality modeling also strategically guides design activities (e.g., problem decomposition, physical modeling, product architecting, concept generation, and team organization).

A common starting application in functionality modeling is hierarchical decomposition. In this step, the overall function is decomposed into sub-functions that will be satisfied by function carriers in the product <sup>[2,18,25]</sup>. The hierarchical structure is a graph technique used to visually keep track of the evolving design and the parent-child relations. Schmekel <sup>[25]</sup> used both the composition and decomposition of functional objects in his functional model. In the decomposition step (top-down design), the designer starts with a functional model and decomposes functions into sub-functions that will be satisfied by parts in the product. In the composition step (bottom-up design), a function in functional model is satisfied by composing parts of a product into assemblies such that the assembly satisfies the specified functional requirements.

Pahl and Beitz <sup>[2]</sup> described how a function structure could be established by breaking down an overall function requirement into sub-functions. The design process proceeds by establishing the sub-systems and components that together will perform the overall function. Cole <sup>[18]</sup> employed the hierarchical structure in his functional analysis methodology. He used

functional identification diagrams (FID) to define the structure, components, and functions of the system. The FIDs portray the system as a hierarchical structuring of the system's functions.

Functionality representation and modeling during the conceptual design phase have long constituted a challenging research area. Function plays a central role during conceptual design, just as geometry does in a detail design <sup>[4]</sup>. Many researchers spend much effort developing acceptable function ontology and function representation. Schmekel <sup>[25]</sup> described a representation of functional models in terms of functional objects. A functional object is a symbolic representation of a functional description. It describes the functional requirements of a product such that a structure of functional objects describes a functional model of that product. A functional object is given by a three-tuple  $\langle Q_{set}, T_{set}, Constraints \rangle$  where:

$Q_{set}$ : is a set of parameters characterizing input and output quantities in a function.

$T_{set}$ : is a set of parameters characterizing the transformation.

$Constraints$ : is a set of constraints between parameters in  $Q_{set}$  and  $T_{set}$ .

A structure of a functional object is defined in terms of primitive functional objects; that is, a four-tuple  $\langle F_{set}, Types, Functions, Connections \rangle$  where:

$F_{set}$ : is a set of functional objects.

$Types$ : is a set of *type* relations between members of  $F_{set}$ , such that a generic taxonomy of functional objects is defined. A type of objects consists of sub-types and instance of type.

$Functions$ : is a set of *function* relations among members in  $F_{set}$ , such that a component structure of functional objects is defined whereby a compound functional object is composed of primitive functional objects.

$Connections$ : is a set of relations between members of  $F_{set}$  that connects corresponding parameters in different functional objects, such that networks of parameters are defined.

In developing a standard taxonomy for function, Szykman et al. <sup>[26]</sup> identified three benefits: reduction of ambiguity between function definitions, clarification of uniqueness from minimal vocabulary, and uniformity of information exchange between systems. They developed their representation based on a minimalist description of function and flow. Examples of functions include usage, combination, and transformation. Similar to the flow types proposed by Pahl and Beitz, <sup>[2]</sup> examples of flow described by Szykman et al. include material, signal, and energy.

Stone et al. <sup>[20]</sup> introduced a design language called “functional basis,” according to which product function is characterized in a verb-object (function-flow) format. The set of functions and flows is intended to comprehensively describe the mechanical design space. Eight basic types of functions are introduced: branch, channel, connect, control magnitude, convert, provision, signal, and support. The functional basis offers greater consistency than did previous systems. The basic flows proposed by Pahl and Beitz <sup>[2]</sup>, and later refined by Hundal <sup>[22]</sup>, are broken down into more specific terms to form a taxonomy.

Tomiyaama et al. <sup>[19]</sup> developed a methodology to deal with functions called function-behavior-state (FBS) modeling. In this model, a function is represented by two concepts (i.e., its symbol is represented by the relationship between function and behavior). The researchers assumed that the representation of function included human intention, whereas the representation of the behavior of an entity could be determined objectively by its attributes and its relations to other entities, based on physical principles. The state of an entity is the attributes and relations of that entity.

Cole <sup>[18]</sup> described several tools that designers can use in functional analysis. Function identification diagrams are used to hierarchically define the functions of the design problem.



Function lists are similar to diagrams, yet not structured. The design functions are described in a narrative list. Function flow diagrams show the interrelations of the functions identified. A functional interface dictionary is a collection of known functions and their flow elements that may be used to assemble function diagrams. Functional allocation databases integrate the function, performance, interface, activity, and organizational requirements imposed on the design. State transition diagrams are used to determine and define the capability to control the system of functions.

Roy et al. <sup>[27]</sup> introduced a definition of function that captures some of the basic features at an abstract level as well as a detailed design level. Functions defined by Roy et al. also have a mechanism to incorporate functional equivalence classes associated with a function. This means that a function will have references to other functions that can collectively be considered equivalent to the function. The collective equivalence could be of a combination of functions connected by “AND” or “OR” or combinations thereof. Their functional representation is shown in Table 2.1.

**Table 2.1 Function representation (proposed by Roy et al. [18])**

Name	String
Input	{{ <b>Input</b> }}
Output	{{ <b>Output</b> }}
Function_of_Artifact	Reference to <b>Artifact</b> through which functional requirements are achieved
Relations	{{ <b>Constraint</b> } defined over the inputs, outputs, and internal attributes of the function
Sub_Function_Of	{{ <b>Function</b> }}
Sub_Functions	{{ <b>Function</b> }}
Optimality_Measure	{{ <b>Constraint</b> }} Goal

Mukherjee <sup>[28]</sup> represented function as follows:

$[\langle v \rangle \langle n \rangle \langle m \rangle \langle d \rangle \langle o \rangle]_{locn} \langle Keywords \rangle$ , where:

v: set of verbs,

n: set of names,

m: set of magnitude attributes,

o: another set of nouns representing objects to which the functions applies,

locn: distinguishes between more than one similar functions,

Keywords: an additional set of specialized words asked to enhance the functional representation.

Deng et al. <sup>[5]</sup> used object-oriented technology to come up with a functional representation scheme. They proposed a multiple-attribute object-oriented representation scheme that explicitly incorporates the functional environment. The most generic function can be represented as the top-most functional object. The class of these objects is defined as follows:

```
Class function {  
    Name:          Act + Target  
    Complement:  Additional information  
    Type:         Performance/ Assembly/ Manufacturing/  
                   Marketing/ Maintenance/ Others  
    Level:        Overall/ Embodiment/ Geometric  
}
```

The *Name* attribute is expressed by the *Act* and *Target* variables. *Act* is used to describe the action relating to the function (e.g., transmit, change), which is normally a verb. *Target* is used to describe the target of the *Action*. *Complementary* attribute is used to provide additional

information for the *Name* attribute. *Type* attribute refers to the mechanical function type that can be identified by taking a functional view of the concurrent engineering technique Design-for-X. *Level* attribute represents the levels of mechanical function. Overall functions describe the overall functionality of the mechanical system or mechanical assembly. Embodiment functions describe the functionality of the components of an assembled product. They are individually necessary and collectively sufficient to achieve a specified overall function. Geometric functions describe the functionality of specific geometric features, forms, or parts.

Muogboh <sup>[29]</sup> developed a functionality model to support computer-aided conceptual design (CACD). The functionality model provides a description of the product in the form of functionality relations and constraints imposed on the physical resources used in the realization of the given task. New modeling concepts in the form of operands and coupling bonds are introduced in the modeling of product functionality. The resources required to accomplish a function (material and energy) are described in the form of functionality operands. The relations and constraints are defined in terms of coupling bonds. A generic model of product functionality was developed to describe functionality in a mathematical form. Muogboh's functionality model (especially the functionality operand concept) is adapted in this research with some modifications.

### **2.2.5 Function-to-Form**

The central task of the product design process is to decompose overall functional requirements into appropriate sub-functions and to synthesize reliable components to achieve those sub-functions. The next task is to embody these components in a workable economical and manufacturable design <sup>[23]</sup>. While some researchers studied functionality representation and

modeling, others developed methods that map function to appropriate form or appropriate physical element [8,9,27,28,30,31,32]. Roy et al. [27] proposed a design synthesis process for the evolution of a product from its product specification. This method is mainly a mapping process from functional requirements to artifacts, with multi-stage, constrained optimization during the stages of design evolution. Physical and structural details of an artifact are captured as abstract sketches. The design of an artifact is represented as:

$D = \{ \langle PS \rangle \langle Art\_Tree \rangle \}$ , where:

$\langle PS \rangle$ : product specification,

$\langle Art\_Tree \rangle$ : the artifact tree (a tree structured list of artifacts).

Initially the artifact tree is empty. Subsequently, when suitable artifacts are mapped to perform a desired functionality, these artifacts are added to the artifact tree. This iterative design process generates stages of partial solutions. At each stage, the attributes in the product specification and the constraints that have not been fully satisfied are transformed to the next stage, until a feasible design solution has been devised.

Mukherjee et al. [28] used data structures called “function-form matrices” to connect functionality with relating geometry. The decomposition of a function into sub-functions reaches a stage at which the designer begins to map the lower level functions with part geometry. The geometry at this point is not the entire geometry of the part; it is the functionally critical parts of the entire part geometry. The remaining geometry is abstracted using a set of linkages to create the sketching abstraction. The concept of sketching abstraction is to represent functional features using wireframe geometry located in partially defined planar regions and connect these functionality critical pieces of geometry with a linkage mechanism.

Gorti et al. <sup>[9]</sup> developed a symbolic evolution approach called “symbol-form mapping,” in which an evolving symbolic description of a design is mapped into a geometric description. The symbolic evolution of a design is guided by a considerable amount of knowledge: knowledge about the components used during the design, the process knowledge used in a particular task, the current context conditions, and user decisions and preferences. They called this part of design “function-symbol mapping.” At this stage, the specific functional requirements coupled with the design context lead to: (a) a selection of components, (b) establishment of functional relationships between components, and (c) establishment of important parametric constraints. On the other hand, in symbol-form mapping, the symbolic aspects must somehow be mapped into a geometric description in order to physically realize the design. Gorti et al. used an object-oriented paradigm in geometry representation to provide a solution that allows both the natural hierarchical decomposition of domain shape knowledge and the procedural computations necessitated by geometric algorithms.

Gardan et al. <sup>[8]</sup> introduced three methods to translate function specifications to shape. The first is an expert system method founded on the expert’s knowledge. All the mathematical and empirical rules used by the expert are implemented to arrive at a single correct solution. For the foundry mould, they used mechanical and metallurgical rules, which cannot be modified, and know-how based rules that can be brought into question with new experimentation. They pointed out that this method carries some drawbacks. The suggested solution is perhaps a good one, but is certainly not the best one. Moreover, the system only imitates the expert and is not innovative.

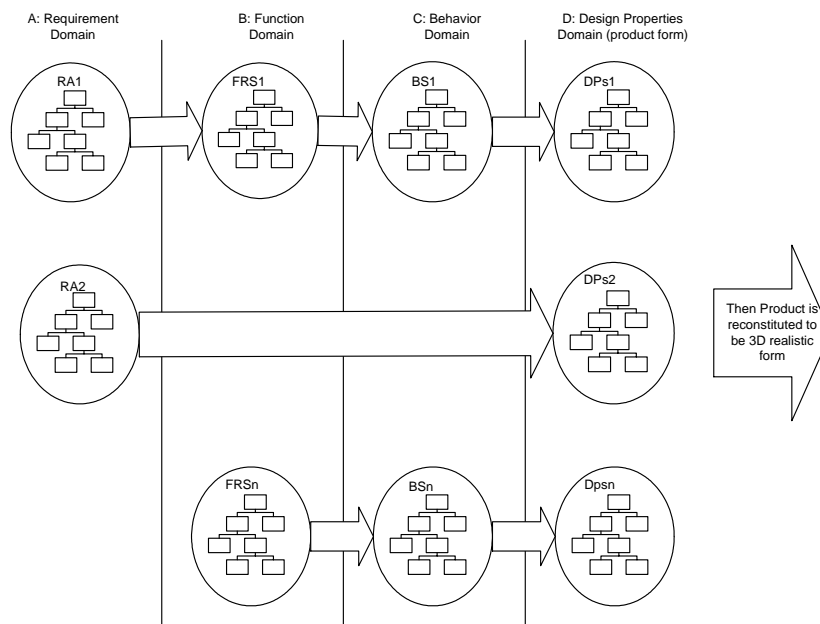
The second method proposed by Gardan et al. is a theoretical method. In this method, they assumed that the product’s various components can be identified after the functional decomposition. Each component is described by a set of constraints on physical parameters,

called “intermediate specifications.” Parameters used in the intermediate specifications are called “intermediate parameters.” In this method, they focused on translating the constraints on the intermediate parameters, not the functions, into shapes. Intermediate parameters are defined as quantifiable and measurable entities referring to the physical world. A component is produced from its own parameters. Each component is a rigid, finite, and homogenous solid. A set of shapes, called “solution space,” is generated for a component from primitive shapes. They supposed that a library of primitive shapes is known by the system. Each of primitive shapes is defined by a set of parameters called “terminal parameters.” For example, a sphere is defined by a radius, a box is defined by three lengths, and so on. Translating the intermediate constraints into constraints on terminal parameters creates the solution space. Due to the huge solution space of that will be produced by this method, Gardan et al. introduced a third mixed method. This consisted of applying the theoretical method and integrating some expert knowledge in order to automatically obtain a reasonably wide solution space.

Xu et al. <sup>[30]</sup> introduced a function-oriented, axiom-based generalized mapping scheme, including generalized mapping among requirements domain, function domain, and form domain to support product life-cycle design. Their mapping scheme has the characteristics of many-to-many and multilevel, bestriding characters (see Figure 2.6). The aim of this strategy is to transform the traditional, highly random design practice into an axiomatic, rationalized way. This generalized mapping strategy falls into four main modules:

1. A functional element library is based on the mechanism transmission rules, theories, and conventions; the mechanical element is dragged from the library and dropped on different 3-D orientation planes to construct a conceptual mechanism layout.

2. A function-to-function carrier-to-form mapping paradigm is used to finish the conceptual embodiment design, wherein each functional element is mapped to a certain abstract feature of the functional requirement.
3. The assembly model is represented as a network, wherein each node is an information unit called a “functional carrier,” and the arc is a constraint.
4. The function carrier is stored in each part in a random order.



**Figure 2.6 Process of generalized mapping**

The information generated from the above mapping process is usually provisional and inconsistent, and the geometric information is always in a high degree of uncertainty. To cope with this problem, data structure and algorithm of conceptual geometric reconstitution were explored. They represented each functional carrier by a new data structure called “Zero solid.” This Zero solid contains three parts: container feature and inner and outer features. Then they

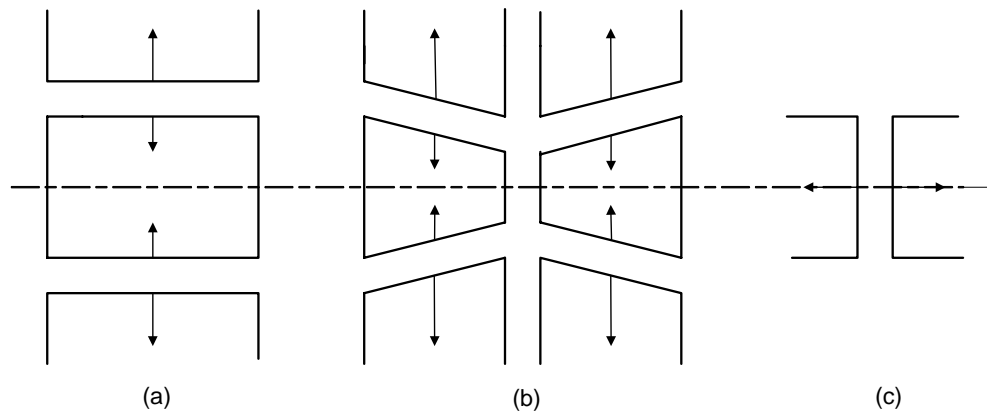
represented the conceptual part by a network of Zero solids. They used a default geometric reasoning (DGR) scheme for the recognition and reconstitution of product geometric and topologic information. The DGR means that, when any of geometric elements in a solid (e.g., vertex, edge, face) are incomplete or indefinite, the missing geometric elements must be supplemented and the relationship of the geometric elements must be reconstituted to construct unambiguous information that is consistent with the object under default logic.

Schulte et al. <sup>[31]</sup> used functional features that describe sets of primary functional faces (active surfaces) to help in functional reasoning, not only on the logic level of the design process, but also on the geometric level. They considered the physical effects and their primary functional faces as the basis for the definition of functional features, which makes it possible to document the fundamental ideas of the designer in terms of geometry. A physical effect is laid down with regard to a specific function or sub-function, and at the same time is usually characterized by plotting a certain arrangement and interrelationship of primary functional faces. According to their point of view, this offers the first direct relation between the functional and the geometric characteristics of a technical product. To obtain the required preliminary layout of the product, further surfaces have to be added. They called these surfaces “filling faces,” which do not have a functional meaning and may be modified without changing the product’s behavior.

Figure 2.7 shows an example of a simple functional feature, called “frictional connection,” that is related to a design function, “input or output of torque/speed.” As the underlying physical effect, “dry friction” can only be realized by at least one pair of functional faces, and the functional feature consists of two or four functional faces. The necessary separation of functional faces, indicated in the figure by material vectors, will automatically result in separate modules and separate assembly units at later design stages. The three different



types of the functional features “frictional connection,” shown in the figure, could lead to some kind of a flat belt drive, vee-belt drive, or disk-type clutch.



**Figure 2.7 Functional feature “frictional connection” and its variation**

Feng et al. <sup>[32]</sup> used graph theory and matrices to represent the function-feature relations, as well as their interrelations. Three basic types of feature-related functions were defined: performance-related functions, process-related functions, and ergonomics-related functions. The performance-related functions are obtained directly or indirectly from the functional requirements. The functional requirements of a mechanical product can be divided into mechanical, structural, thermal, fluid, and tribological functions. Further decomposition for each one is possible.

A n-ary tree was employed by Feng’s group to store the functional information in a hierarchical way. The root of the tree corresponds to a very abstract representation, whereas the leaves correspond to a more concrete representation. The correspondence between the functions and the shape is made on the level of the leaves of this structure, which are directly associated with geometric entities (e.g., form features).

Shapiro et al. <sup>[33]</sup> introduced a new view for modeling mechanical functions in terms of energy exchanges. A mechanical system interacts with its environment by exchanging energy through its physical boundary. The subsets of the physical boundary over which such exchanges occur are called “energy ports.” Energy ports are the key portions of geometry on a mechanical part and exist at contact surfaces (e.g., the internal surface of a bearing bore). Their shape is usually directed by the nature of the contact (e.g., once a specific bearing has been selected, the dimensions of the bore are implied). Thus, the geometries of the energy ports are fully specified and can be considered to be the functional features of the part to be designed.

Rosenman <sup>[34]</sup> employed genetic algorithms to automatically generate 2-D plans for house designs. The idea of a genetic algorithm consists of simulating the model of biologic organism evolution. The shape of biologic organism (the phenotype) is represented by genetic information called the “genotype,” which contains both the rules of development for the organism and the development process itself. A genotype algorithm is characterized by an initial population on which different mechanisms of evolution are applied, such as the crossing (mixture of two genotypes), the mutation (randomly modification of a genotypes), and so on. The phenotype in Rosenman’s work is a surface composed of square elements. It is represented by a sequence of horizontal or vertical unit vectors. The obtained shapes are automatically valued with objective fitness functions (e.g., the ratio perimeter of the area or the number of angles) and are manually valued with subjective functions (e.g., aesthetics).

Some researchers have focused on evaluating different design alternatives based on functionality representation <sup>[35, 36]</sup>. Iyengar et al. <sup>[35]</sup> developed a generic method of evaluating or comparing different systems or designs that perform the same function. “Different” refers to physical domain, size, connectivity, components, complexity, and so on. First, a functional

technique is used to come up with a common representation of different kinds of components and systems. Second, quantifying evaluation criteria (e.g., performance, reliability, serviceability) are defined. Finally, an upward recombination technique is used to collapse the expanded functional network representing the design, based on the evaluation criteria. The general method involves retracing the path followed in the functional decomposition process. Gardan et al. <sup>[36]</sup> proposed a way to automatically check whether a shape satisfies the function specifications by using a weighted average formula called the “satisfaction degree.”

Huang et al. <sup>[37]</sup> used a morphological evaluation chart to develop their “concept assessor,” which is responsible for concept evaluation or assessment in the conceptual design phase. The input to this concept assessor is the list of solution alternatives obtained from the concept generator, and the output is a rank order of the input solution alternatives to be forwarded for further development. The conversion from the input to the output is based on the set of criteria against which alternative solutions are evaluated.

### **2.2.6 Computer-Aided Conceptual Design**

Computer tools have been used extensively in the detailed design phase, but relatively few applications exist for the conceptual phase. This is because knowledge of the design requirements and constraints during this early phase of the product life cycle is usually imprecise and incomplete, making it difficult to utilize computer-based systems or prototypes <sup>[38]</sup>.

Current research on CACD has produced several prototype systems. Wallace et al. <sup>[39]</sup> developed an experimental conceptual design tool for industrial designers that is capable of generating alternative design suggestions based on user input. The purpose of the tool is to provide a means for designers to quickly generate and adapt alternate design concepts. Anthony

et al. <sup>[40]</sup> described a 3-D modeling tool for conceptual understanding and prototyping (CUP). CUP allows a user to specify a spatial layout of components and sub-assemblies, as well as structural, behavioral, and functional (SBF) information about components and sub-assemblies. Also, CUP possesses mechanisms for capturing textural information about the designer's intent and preferences.

To support design activities adequately, Al-Hakim et al. <sup>[41]</sup> proposed the incorporation of reliability with functional perspectives at the conceptual design stage. They used graph theory to represent a product and the relationships between its components. With this representation, it is easy to visualize energy flow between components, and thus trace any loss of functionality. Brunetti et al. <sup>[42]</sup> proposed the use of features to model the relationships between the requirements, functional descriptions, and physical solutions of a product. These features serve as information carriers to the downstream applications. In particular, the researchers focused on the support for early feature-based prototyping of different views to the overall product model.

Qin et al. <sup>[43]</sup> developed a fuzzy knowledge-based system that can capture the user's sketching intentions and automatically generate the corresponding geometric primitives. Most designers still prefer to express their creative design ideas through 2-D sketches; therefore, it is important for CACD systems to allow sketched input. Qin et al. built a prototype system that allows 2-D sketched input, interprets the input sketch into more geometrically exact 2-D vision objects, and when needed, projects the 2-D objects into 3-D models.

Conceptual design activity is not complete unless the design concepts are verified and satisfy the functional requirements. Deng et al. <sup>[44]</sup> investigated in automatic design verification through the use of a constraint-based approach. Design verification is typically done either by calculating the attributes of interest directly or by simulating the behavior of a system. Deng et

al. developed an extension to the first method through the use of constraint propagation and dynamic design verification, based on graphs. The input is the functional information. Based on the input, they developed a framework that allows for backward reasoning to the causes of the system behavior. Design verification is performed by identifying input and output design variables, developing a variable dependency graph, propagating constraints over the graph, and checking the values of the design variables against these constraints.

## **2.3 FEATURES**

The use of features is a recent approach to the old problem of attempting to link CAD with computer-aided manufacturing (CAM). The features approach constrains the designer/process planner to work with a set of features that have significance for design, analysis, or manufacturing. Instead of using a model consisting of graphic primitives (e.g., lines, circles, and points), the designer is asked to use a set of features (e.g., holes, pockets, slots) from which manufacturing operations can be derived. When both the designer and the process planner have finished the design and process plan, more information has been entered on a traditional CAD system. However, the designer and planner have entered information not at the geometric level, but at a higher level <sup>[45]</sup>.

### **2.3.1 Definitions of Features**

Features do not have a formal mathematical or generic definition. A feature is usually defined with respect to its significance for a particular application (e.g., design, engineering analysis, or manufacturing) for which it captures accumulated experience-based knowledge.

Each researcher working in the features arena has his or her own definition of features, and these definitions differ.

In the work of Shah and Rogers <sup>[46]</sup>, the term “feature” was defined as a set of information related to an object’s description. The description could be for design, manufacturing, or even administration purposes. Thus, the nature of the information sets can be different. These researchers classified features into sets related to product engineering as follows:

- Form features (nominal geometry): functional, aesthetic, and assembly aids
- Material features (material composition and condition): properties/specifications and treatment applied to the material and surfaces
- Precision features (allowable deviations from nominal geometry): tolerances and surface finishes
- Technological features (information related to the object performance and operation): performance parameters, operating variables, and design constraints

Shah <sup>[47,48]</sup> later redefined features as geometric forms that engineers associate with certain properties or attributes and with useful knowledge for reasoning processes related to the product; in other words, the features can be seen as primitive forms of engineering. McGinnis and Ulman <sup>[49]</sup> defined a feature as any particular or specific characteristic of a design object that contain related information about that object. A feature is verbally represented in the form of a noun or noun clause. Brunetti and Golob <sup>[42]</sup> defined a feature as an information unit representing a region of interest within a product. It is described as an aggregation of properties of a product and the description contains the relevant properties, including their values and relations (i.e., structure and constraints). Furthermore, it is defined as the scope of a specific view of the product, with respect to the classes of properties and phases of the product life cycle. According

to Salomons <sup>[50]</sup>, features can be treated as design objects, belonging to a general class that inherits properties of other classes. Liu and Nnaji <sup>[51]</sup> defined features as a set of geometric entities (i.e., surfaces, edges, and vertices) with specifications for the bounding relationship between them, which implies an engineering function for the object. Rembold, Nnaji, and Storr <sup>[52]</sup> classified features according to three levels: generic, application, and product. Their classification scheme is hierarchical and the result from any high level classification applies to the lower levels.

### **2.3.2 Feature-Based Design**

Because form features can represent the design intent and functionality of a part, form-feature methodologies have recently been employed in the development of mechanical design environments. This approach is known as “feature-based design” or “design with features.”

In applications of feature technology to CAM and computer-aided process planning (CAPP), design features are defined as primary features, while feature of downstream activities (e.g., manufacturing, process planning) are considered application features. Currently, three primary approaches exist with respect to how to obtain application features from a product model <sup>[50]</sup>. One approach involves a design with features, or a feature-based design. A product model can be built using design features. Features are functional elements for designers. However, design features often differ from application features. The second approach is feature recognition. In this approach, features are automatically or interactively recognized from a model of the object under consideration. The third approach is interactive feature definition. In this approach, features are defined by human assistance or interactively.

In comparison with the other two approaches, feature-based design has the advantage of offering relevant information for applications during the design process, as well as the possibility of considering manufacturing and assembly concerns early in the design process. This would not be possible using either feature-recognition or interactive feature definition alone. Thus, feature-based design is a promising means of achieving better CAD/CAPP integration <sup>[50]</sup>.

Research in the area of features has resulted in many promising techniques for combining engineering data and design knowledge with geometric information. Almost all the research on features has been in the domain of mechanical design. This is because the primary goal of mechanical CAD systems has been to provide concise, accurate representations of mechanical parts, along with their corresponding manufacturing processes. In feature-based design, part geometries are constructed, edited, and manipulated with form features (e.g., holes, slots, ribs, webs). These are high-level geometric entities with attributes that identify functionality and information related to design and to manufacturing knowledge. Furthermore, the need for integrated design environments and for automated information processing techniques for complex mechanical designs has prompted extensive research for better representations of the geometry and topology of both completed and in-progress designed objects <sup>[53]</sup>.

### **2.3.3 Features as Integrating Keys Linking Design and Manufacturing**

The automation of process planning requires that product data be automatically extracted from the product model. However, CAD product representations for product modelers usually differ from the type of information required for CAPP (e.g., manufacturing features). Although most of this information is generated during the design process, it is lost because only the results of the design process are stored in the CAD model. Feature-based design could, at least partly,



overcome this problem. Recently, designers have become more aware of the necessity of considering not only the design function and form of a design object, but also manufacturability.

As discussed above, features can be viewed as information sets that refer to aspects of form or other attributes of a product, so that these sets can be used in reasoning about the design, in performance, or in the manufacture of the part or its assemblies. Two primary benefits are associated with feature-based design <sup>[54]</sup>. First, it provides users with high-level, specialized modeling primitives that facilitate a top-down approach in the design process. Second, they contain a various forms of design information that enable application programs to reason about the characteristics of a design object.

## **2.4 PRODUCT MODELING**

A product model is a computer representation of the data that describe a product throughout its life cycle <sup>[55]</sup>. A product model should contain sufficient detail to allow engineering applications to have access to the data they use and to be a repository for the data they create. Therefore, a product model must have a structure so that originators of applications know where to find and deposit data. This structure is provided by a product data model that describes the form and content of the data being represented.

### **2.4.1 Types of Product Model**

Product models have been used over the past 20 years, sometimes without explicit naming of a product model. After the introduction of geometric models for various CAD/CAM applications, engineers and researchers quickly realized it was necessary to store additional information and to extend the modeling capability to capture such information. They also

recognized the need to cope with large-scale and complicated products (e.g., automobiles, ships, and engineering plants), to integrate different kinds of product-related information, and to have a structured representation for these products. In the following sections, different types of product models are explained <sup>[56]</sup>.

### Structure-oriented product models

The product structure refers to a description of the product's breakdown and is the kernel of structure-oriented product models. To represent the structure of products, several types of structures (e.g., different bill-of-material structure types, classification structures, structures to describe versions and variants of a product) can be used.

### Geometry-oriented product models

Geometry-oriented product models (e.g., wire frame, surface, solid, and hybrid models) can be defined as computer internal models, with the primary purpose of representing the shape of a specific product. This type of product model is typically used as part of basic CAD systems and provides the basis for such applications as FEM or NC programming. Since the data structures for geometrical models are especially designed to represent geometry, their extension to non-geometric data is limited. For supporting-product geometric design, it is necessary not only to represent final shapes, but also to represent some auxiliary shapes (i.e., auxiliary geometry) and attach a symbolic description for other attributes or relations. More generally, product-related information needs to be associated with geometry.

### Feature-oriented product models

As an extension of geometry-oriented product models, feature-oriented product models provide the ability to represent frequently used shape patterns as coherent geometric items, called “form features.” Form features are application independent because they do not carry any specific non-geometric semantics. Most form features represent shape patterns and have a specific semantic meaning related to the design or manufacturing process. To better support design and manufacturing tasks, it is necessary to extend the concept of form features by the explicit representation of their semantics. As a result, features represent the integration of form features and application-dependent semantics. In product modeling, two main classes of features can be distinguished: design and manufacturing features.

Design features support the designer in communicating with a design system easily and according to their design intent. During conceptual design, features related to the primitive geometric elements or symbols can be used to describe the functions of the desired product. Manufacturing features are defined as the interpretation and, most importantly, the combination of form features from the viewpoint of manufacturing, assembly, and inspection. Like design features, they are constructed by combining shapes, depending on their position in the piece of work.

### Knowledge-oriented product models

Knowledge-oriented product models are characterized by the use of artificial intelligence (AI) techniques (e.g., object-oriented programming, rule-based reasoning, constraints and truth maintenance systems). Through the employment of these AI techniques, it becomes possible to store human expertise as well as experience concerning products, processes, and factory

environments. One important characteristic of knowledge-oriented models is their ability to build abstract taxonomies of products or processes as objects and to store knowledge about former designs, possible alternative parts in an assembly, and the abilities and validity of processes used for a specific class of products.

### Integrated product models

An integrated product model incorporates the abilities of geometry-, feature-, structure-, and knowledge-oriented models. All types of product information can be stored in the integrated product model. Generic product knowledge comprises the product history, development principles, model of customers, technological requirements, and failure models. The representation of generic product knowledge takes into consideration the different stages of the product life cycle. In this way, integrated product models provide integrated support for product development over the whole product life cycle.

#### **2.4.2 Model Standardization Using STEP**

One of the most significant approaches toward the implementation of integrated product models is the development of the ISO Standard 10303 “Standard for the Exchange of Product Model Data” (STEP). STEP is an international standard that provides an unambiguous representation and exchange mechanism for computer-interpretable product information throughout the life cycle of a product. In addition, it provides a consistent data exchange format and application interfaces between different application systems (e.g., CAD/ CAM software). The present implementation methods for information transfer include file exchange, an application programming interface, and database sharing <sup>[57]</sup>.

### 2.4.3 Product Modeling to Support the Design Process

Product models, which are required to store and communicate the ideas of the designer, are an essential tool for supporting the drive toward concurrent engineering and improvement of the product development process<sup>[58]</sup>. Product modeling has been regarded as a key technique to develop conceptual and detailed design support systems. Therefore, product modeling has received attention from many researchers.

Bradley et al.<sup>[58]</sup> proposed a relation-based product model for computer-supported early design assessments. It is an aggregate product model in which the concept of relations between features is used to allow the extension of the model from the conceptual to detailed design stages. The approach the major difference between this model and most feature-based models to is the definition of feature geometry. The feature's primitive classes are not defined with a fixed and limited set of geometry information and must be specified in order to store the feature within the model. Instead, the system seeks to allow the user as much flexibility as possible in the definition of geometry.

Arai et al.<sup>[59]</sup> dealt with the development of a product modeling system in the conceptual design phase. They divided the conceptual design phase into two steps: the design specifications processing step and the initial geometric modeling step. Product modeling is executed through these two steps. In the design specifications processing step, the requirements are developed into the design specification by utilizing a database in which the designers' experiences are stored. In the next step, non-manifold geometric modeling is developed and combined with a solid modeling system and technological information/attribute modeling system. At the same time, the designers' intention, which is the most important information generated in the conceptual design

phase, is documented with the developed design process description language (DPDL) and connected with the geometric model in the developed product modeling system.

Tay et al. <sup>[60]</sup> described a function-based product model for conceptual design support. Their model represents and organizes product information in both the functional and physical domains in a multilevel and object-oriented manner. They used function-form relations to map the two domains. All basic product information (e.g., function requirements and mechanical drawings) are stored and managed by a relational database system that provides the facilities for tasks such as data consistency and integrity control. Different object instances that represent different views of the actual products are then constructed. These objects contain pointers that refer to the corresponding data in the relational database instead of the actual data, thus avoiding data redundancy and improving data integrity. Multiple layers of objects can be defined so that an object can be expressed in terms of other objects and can be shared by other objects.

Al-Hakim et al. <sup>[41]</sup> employed graph theory to represent a product and define the relationships between its parts. They used the graph theory concepts of the tree and forest to represent a functional design artifact and idle conditions, respectively. In this approach, the nodes of the graph are the connections that allow the flow of energy, and the edges are the components. By examining the flow of energy during the idle and operating conditions of the product, we can determine the redundant components of a product at its early design stage. Also, the approach improves visualization of the energy flow between the components and enables the designer to identify and examine various components' configurations.

Baxter et al. <sup>[55]</sup> proposed a functional data model that allows the functions of a product to be represented in a form that can be used by a number of engineering applications. This functional model is built based on the structure of the product data model defined using

EXPRESS-G. In this model, the product representation is combined with function representation. A product is represented as a set of parts. A part can be a component or an assembly. An assembly is a set of two or more parts, and a component is an object that cannot be subdivided further. Components are defined in terms of combinations of features. Also, a three-stage life cycle is represented: specification, definition, and actuals. Specification represents what is required, definition represents the design that shows how the specification is to be met, and; actuals represents the details of how the definition was actually realized. At the same time, functions have five attributes: named, inputs, outputs, has-need-of, and performed-by. Through functional decomposition, this strategy may be used to validate whether a product model fully satisfies its functional requirements.

Brunetti et al. <sup>[42]</sup> introduced an approach to a feature-based integrated product model that incorporates a feature-based representation scheme for capturing product semantics handled in the conceptual design phase. They claimed their model would improve concurrent engineering and top-down design by supporting early feature-based prototyping of the different views of the overall product model. Their model structures data into four levels: an assembly model, a part model, a feature model, and a generic model. Starting with the assemblies, every representation level is mapped on a model of the subsequent level (i.e., elements of a higher level are modeled by elements of the lower levels). A feature model is built on top of the generic representation, which stores application-independent product properties within a generic model. The data in this generic model specifies the form and parameters of the parts, as well as the geometric and topological constraints defined by them.

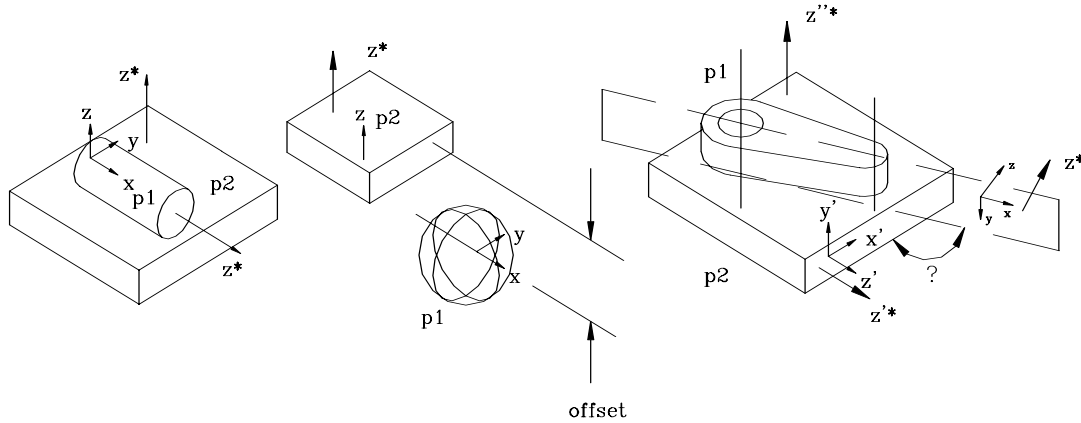
## 2.5 PRODUCT DESIGN AND SPATIAL RELATIONSHIPS

Spatial relationships were proposed by Ambler and Popplestone<sup>[61]</sup> to describe the relative positions of product components in their final state by specifying the feature relationships among them. The spatial relationships are: against, coplanar, fits, parax, lin, rot, and fix. These spatial relationships were firstly used for the configuration of a product. Liu and Nnaji<sup>[51]</sup> introduced a revised form of the spatial relationships that not only can be generally applied to assembly, but is also capable of accepting the design specifications and the designer's intent. They introduced six types of spatial relationships (see Figure 2.8):

- *Against*: The mating faces touch at some point. It is the most basic relationship and applies to any parts assembly.
- *Parallel-offset*: The parallel relation holds between planar faces and cylindrical and spherical features. In two parallel planar faces, the two outward normals are pointing in the same direction. Two features with an offset distance have no physical contact.
- *Parax-offset*: This relationship is similar to that of parallel-offset, but the outward normals of the parallel planar faces are in the opposite direction.
- *Aligned*: This relationship exists if the center lines of the two features are collinear.
- *Incline-offset*: The inclination relation holds for an angle between two planar faces. The offset describes the distance from a planar face of a part to the intersection line of the two faces, which makes the inclined angle.
- *Include-angle*: This relationship is similar to that of incline-offset. An include angle between two planar faces is in the positive normal direction. The rotation is clockwise to a normal of a picking faces. The rotational axis has to be parallel to the normals above two planar faces.



A spatial relationship can be interpreted as a constraint imposed on the degrees of freedom between relative mating and interfacing features. Any allowable motions for parts have to follow a path along the directions specified by the degrees of freedom to maintain their spatial relationship.



Degrees of freedom based on geometry:

p1: {plane\_z::rot\_x,rot\_z}  
p2: {cyl\_x::lin\_y,rot\_z}

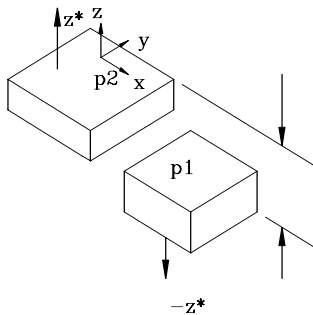
(a)

p1: {plane\_z::rot\_x,rot\_y,rot\_z}  
p2: {sph::plan\_z,rot\_z}

(b)

p1: {plane\_z::lin\_z,rot\_z}  
p2: {plane\_z'::lin\_z',rot\_z'}

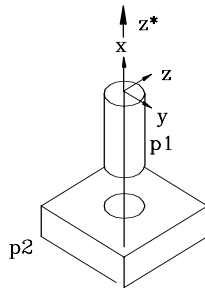
(c)



Degrees of freedom based on geometry:

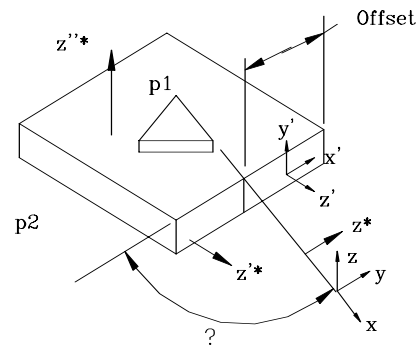
p1: {plane\_z::rot\_z}  
p2: {plane\_z::rot\_z}

(d)



p1: {lin\_x::rot\_x}  
p2: {lin\_x::rot\_x}

(e)



p1: {plane\_z::rot\_z}  
p2: {plane\_z'::rot\_z'}

(f)

**Figure 2.8 Types of spatial relationships: (a) against, (b) parallel-offset, (c) include-angle, (d) parax-offset, (e) aligned, (f) incline-offset (adapted from Muogboh [29])**

### **3.0 OVERVIEW OF FUNCTION-TO-CONCEPTUAL FORM TRANSLATION MODEL**

From the literature review chapter, we can see that many researchers have identified the importance and relevance of function modeling and function-to-form mapping to engineering design. At the same time, however, a well defined and widely accepted function-to-form translation method does not yet exist. In this research, a model for translating function specifications into conceptual forms is developed. The concepts of functionality operation, operand, relationship, function server, and conceptual product model are utilized to aid functionality-based design and to realize a product's conceptual form. The approach of this research allows for both the functional and the conceptual physical description of a mechanical device.

The function specifications-to-conceptual form translation tool (hereafter, simply “translation tool”) developed in this research aims at translating an abstract representation of a product into a more concrete representation. During the conceptual design phase, the designer adds information to a design that has very low information content. At this very abstract stage, the design team has only a primary objective function and a handful of constraints and criteria. Information must be added to the design incrementally, making it more concrete with each step. One typical approach is to decompose the objective function into sub-functions. Then, the design must cross the function-form boundary and enter the embodiment stage.

In order to achieve the above objectives, the following tasks were identified as essential:

- Functionality modeling to capture and maintain the designer's intent during the conceptual design phase and to help in searching for possible physical solutions.
- Function server modeling to represent and capture the information about the physical solutions. This founds the basic structure of the function driven database.
- Function-to-function server mapping to develop the set of relationships and mapping operations between the functional and the physical domains.
- Conceptual product model to document the mapping results and to represent and organize product information in both the functional and the physical domains.
- Data structure development to allow for the description of product functionality and product form in a transparent manner.
- Computational tools to implement the above concepts. This includes a functionality object model, a function server object model, data representation in XML, and a web-based user interface.

This chapter introduces the scope of and the assumptions related to this work. In addition, the components of the translation tool model are described.

### **3.1 SCOPE OF WORK**

The scope of this work is limited to mechanical devices governed by the well defined laws of physics. A mechanical device is a piece of equipment designed to serve a special purpose or perform a special function <sup>[72]</sup>. It generally consists of mechanical members connected by joints. These members transmit motions by moving upon each other as mechanisms or transmit

force without motion as structures. This limitation will increase the chance of producing a practical design. It is hoped that this work will be able to be extended to other classes of engineering systems.

Another limitation of this project is that the materials are restricted to only rigid bodies (solid material). A rigid body can be considered as a combination of a large number of particles, in which all the particles remain at a fixed distance from one another both before and after applying a load <sup>[73]</sup>. Deformable bodies and fluids are not considered in the functionality model. Moreover, energy is restricted to mechanical energies comprised of force and torque. The extension to all classes of material and energy can be done in future work.

## **3.2 FUNCTIONALITY MODEL**

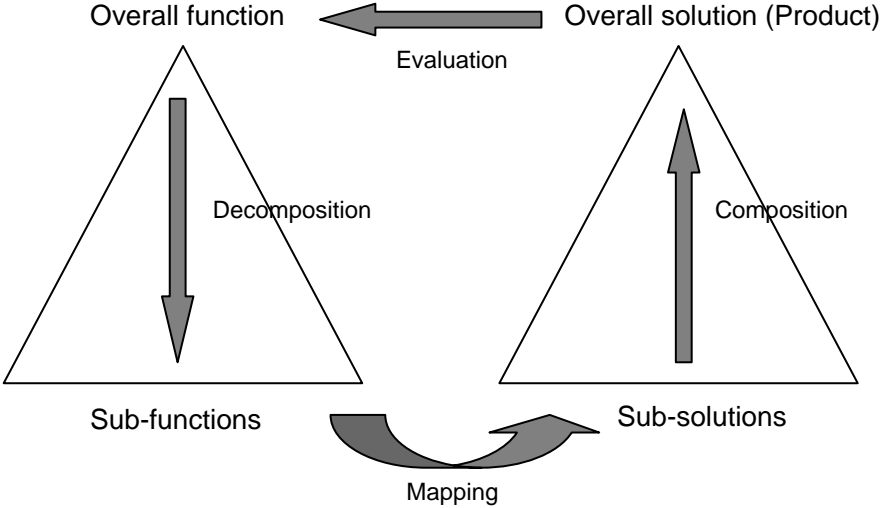
The functionality modeling of a mechanical product can be regarded as a process of establishing the functionality operations of a desired but physically indefinite product. This is different from the functional modeling of an existing physical device, because the object being modeled has not yet been physically defined. The major issues are how to represent and manipulate functions, so that they can be mapped into a description of a realizable physical structure. The functional data model developed in this research was built on the functionality model proposed by Muogboh <sup>[29]</sup>.

### **3.2.1 Functionality Operations**

The functionality of an entity (or a product) is the general task it performs. The entity may be either a physical artifact, such as the drive shaft of a car, or an invisible object, such as

the electro-magnetic forces used to relocate a ferro-magnetic object. The existence of an identifiable task that needs to be achieved is the essence of the functionality. This task should be independent of any predefined solutions.

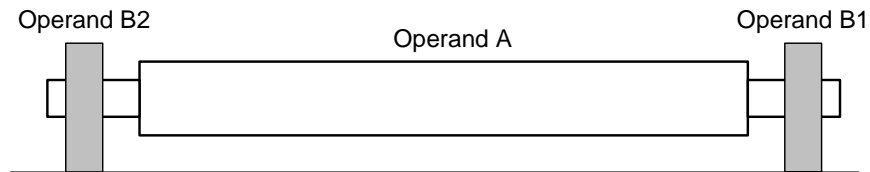
Design is function driven. In the usual design practice, a designer starts with an objective, which is a functional description of a product. This primary functional objective (overall function) is then decomposed into several sub-functions. Sub-solutions are then selected to perform the specified sub-functions. Finally, the overall solution (product) is constructed from a related set of sub-solutions. Figure 3.1 illustrates the function driven design cycle.



**Figure 3.1 Function driven design cycle**

A functionality operation is defined as the realization of a given task, where the task is the desired functionality of the entity. A functionality operation defines relationships and constraints on functional elements (i.e., operands and attributes). For example, if the task of supporting a shaft is considered, as shown in Figure 3.2, the functionality is the support of the

shaft. The functionality operation is the representation of the act of providing support to the shaft.



**Figure 3.2 Example of mechanical functionality (support of a shaft)**

The mechanical functionality is composed of two components: the operand and the relation. These two entities are defined as follows:

- An operand is a distinct element involved in the realization of a given functionality. Operands are the building blocks that come together to accomplish any mechanical functionality. In the example of supporting a shaft, the operands are the load exerted by the shaft (operand A) and the two objects that support the shaft (operand B1 and B2). If the shaft transmits torque, the operands will be the shaft, the torque, and the component that the torque transmitted to.
- A relation is the established relationship between operands in the performance of functionality operations.

Many researchers <sup>[2,20,27]</sup> have used the term “flow” to model functionality. The definition of functional flow assumes causality, which is a cause and effect relationship (i.e., between inputs and outputs). This assumption limits functionality modeling to the class of functions with identifiable inputs and outputs. In addition, the flow definition includes signals, which can be

derived from a combination of other basic functional elements. The basic functional elements are modeled in this work by excluding signals and the flow restriction on the nature of mechanical elements. Muogboh <sup>[29]</sup> borrowed the term operand from computer engineering, where it is used to represent the data component of a computer instruction code.

### 3.2.2 Generic Functionality Model

A functional model is composed of an aggregate of functionality operations. The functionality operation is defined by a set of operands with their corresponding attributes and the relations between these operands.

Given a set of functionality operands ( $\mathbf{O}_{ij}$ ), with their corresponding attributes ( $\mathbf{a}_{ijk}$ ), we define the generic functionality model,  $\mathbf{F}$ , of a functionality operation as:

$$\mathbf{F}_i = \{(\mathbf{o}, \mathbf{r}, \mathbf{s}) \mid \mathbf{o} \in \mathbf{O}_i, \mathbf{r} \in \mathbf{R}_i, \mathbf{s} \in \mathbf{S}_i \}$$

Where,

$\mathbf{i}$  = functionality operation index.

$\mathbf{O}_i$  = a set of functionality operands corresponding to operation  $\mathbf{i}$ .

$\mathbf{R}_i$  = a functionality relation, which defines a set of relationships between functionality operands.

$\mathbf{S}_i$  = a set of functionality states, which defines the state each operand (or operand attributes) can assume.

The functionality relation defines the relationship between the functionality operands. A relation can have one of three forms: spatial, physical aspect, or constraints. More details about functionality relations are presented in chapter 4.

The functionality relation,  $\mathbf{R}_i$ , is defined mathematically as:

$$\mathbf{R}_i = \{ \mathbf{r}_{ijk} (\mathbf{o}_{ij}, \mathbf{o}_{ik}) \mid \mathbf{o}_{ij} \mid \mathbf{o}_{ij} \in \mathbf{O}_i, \text{ and } \mathbf{o}_{ik} \in \mathbf{O}_i \}$$

Where,

$$\mathbf{O}_{iq} = \{ \mathbf{a}_{iqs} \mid \mathbf{a}_{iqs} \in \mathbf{A}_{iq} \} \quad ; \text{operand } q \text{ in functionality operation } i.$$

$\mathbf{A}_{iq}$  = attribute set for functionality operand  $\mathbf{O}_{iq}$ .

$\mathbf{r}_{ijk}$  = relation, between operands  $j$  and  $k$ .

$\mathbf{j}, \mathbf{k}, \mathbf{q}$  = functionality operand indices.

$\mathbf{i}$  = functionality operation index.

The state of the functionality operand represents the various possible values its time-varying attributes might assume. The state of functional operand  $q$  can be represented mathematically by a state set,  $\mathbf{S}_i$ , as:

$$\mathbf{S}_i = \{ \mathbf{s}_{iq} \mid \mathbf{s}_{iq} \in \mathbf{S}_i \}$$

Where,

$\mathbf{S}_{iq}$  = the state of operand  $q$  in functionality operation  $i$ .

$$\mathbf{S}_i = \{ \mathbf{a}_{iqs}, \mathbf{v}_{iqs} \mid \mathbf{a}_{iqs} \in \mathbf{A}_{iq}, \mathbf{v}_{iqs} \in \mathbf{V}_{\mathbf{a}_{iqs}} \}$$

Where,

$\mathbf{A}_{iq}$  = attribute set of functionality object  $\mathbf{O}_{iq}$ .

$\mathbf{V}_{\mathbf{a}_{iqs}}$  = set of possible values (range) of attribute  $\mathbf{a}_{iqs}$ .

The components of the functionality model are discussed in more detail in chapter 4.



### 3.3 FUNCTION SERVER MODEL

Defining a functionality model facilitates the discovery of solution principles, because it simplifies the general search for them and also because solutions to sub-functions can be elaborated separately. Individual sub-functions must then be replaced with more concrete statements (i.e., concepts). A concept is an idea that can be represented in a rough sketch or with notes, in other words, an abstraction of what might be a product or a component in a product. The generation of concepts from functions is, of course, an essential aspect of design. This dissertation develops the concept of a function server to aid in the conceptual form realization process. The functionality model represents the description of a product in the functional realm, while the function server model represents the description of a product in the physical realm.

A function server is defined as the conceptual physical element that is used in a product to serve or to satisfy a sub-function or sub-functions. Therefore, function servers are sub-solutions of the functionality requirements, which form, when arranged and combined, the conceptual form of the product (i.e., the overall solution). The primitive function servers are classified into three groups:

- Functional features: The elements that comprise a component (functional form feature) and possess functionality, such as holes or bosses.
- Functional standard parts: The parts manufactured in quantity for use in a number of products, such as bolts or springs.
- Functional module parts: The standard sub-assemblies or products, such as motors or pumps.

Each function server has its functional details, structured details, and the working conditional details, along with links to other function servers. The function server representation consists of information about: primary and secondary functions, abstract shape, material type and properties, manufacturing process and properties, interface with other function servers, and working conditions. According to this description, a function server is modeled as:

**FS:** {**FD**, **SH**, **M**, **MI**, **IN**, **WC**}

Where,

**FD:** The function description, which includes the primary and secondary functions that the function server can serve.

**SH:** The shape description, which includes the abstract shape and the base shape.

**M:** The material model, which includes the material type and material properties.

**MI:** Manufacturing information, which includes manufacturing process, surface properties, and alternative materials.

**IN:** Description of interface with other function servers. It includes interface type, degree of freedom, and spatial relationship.

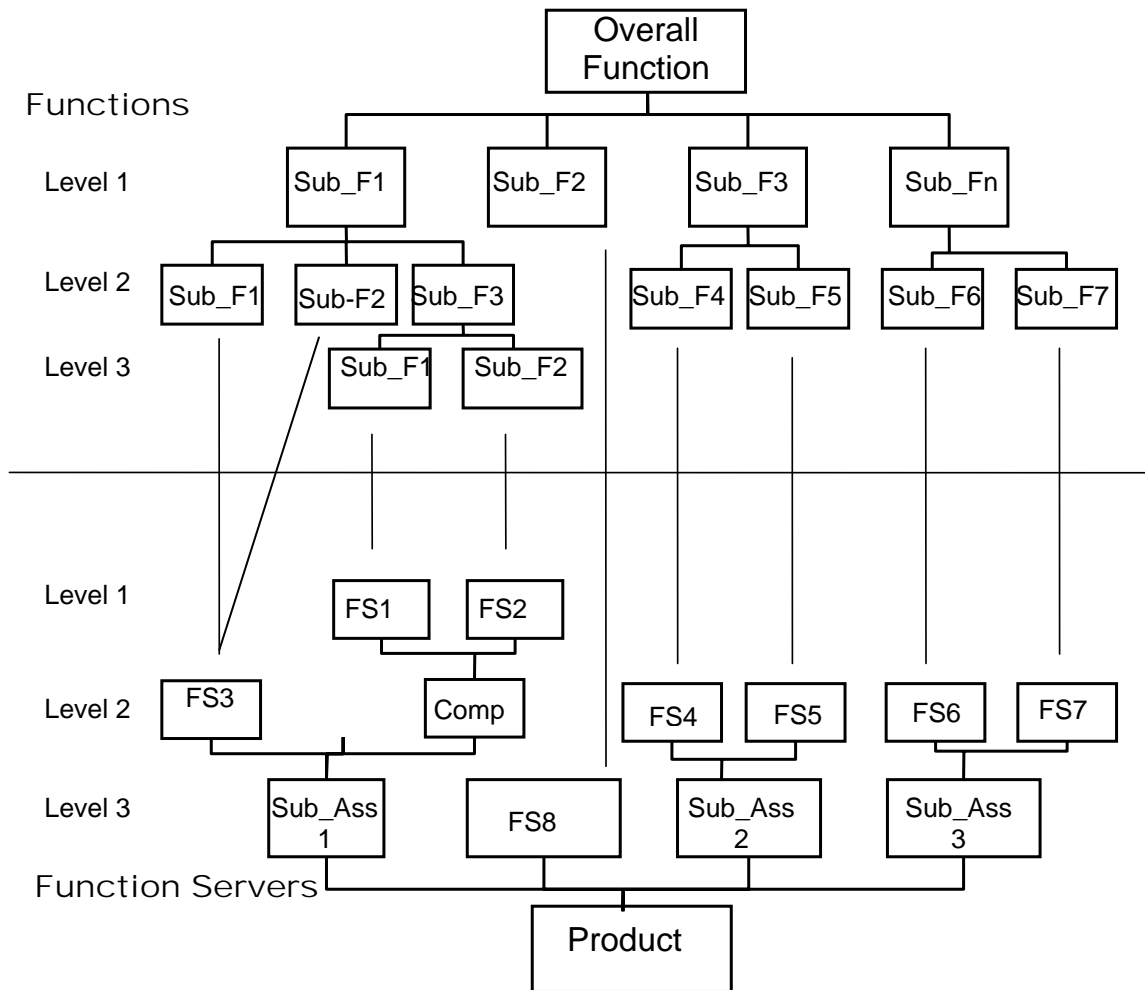
**WC:** Working conditions, such as physical laws, energy type, loading nature, and failure modes.

However, some of this information may not be known at the beginning of the design process. The model is however defined, and, as design progress, these attributes are defined. More details about function server representation and modeling are presented in chapter 5. The function server model is implemented in an object-oriented environment using Unified Modeling

Language (UML). The implementation details are presented in chapter 6. The information about function servers constructs a function driven database that is accessed to search for possible solutions. This database is comprised of a comprehensive information structure for maintaining detailed data information about function servers. A prototype function driven database is developed and implemented in this research project in order to demonstrate the use of the translation tool. More details about this aspect of the project are presented in chapter 6, which includes the implementation and the case study.

### **3.4 FUNCTION-TO-FUNCTION SERVER MAPPING**

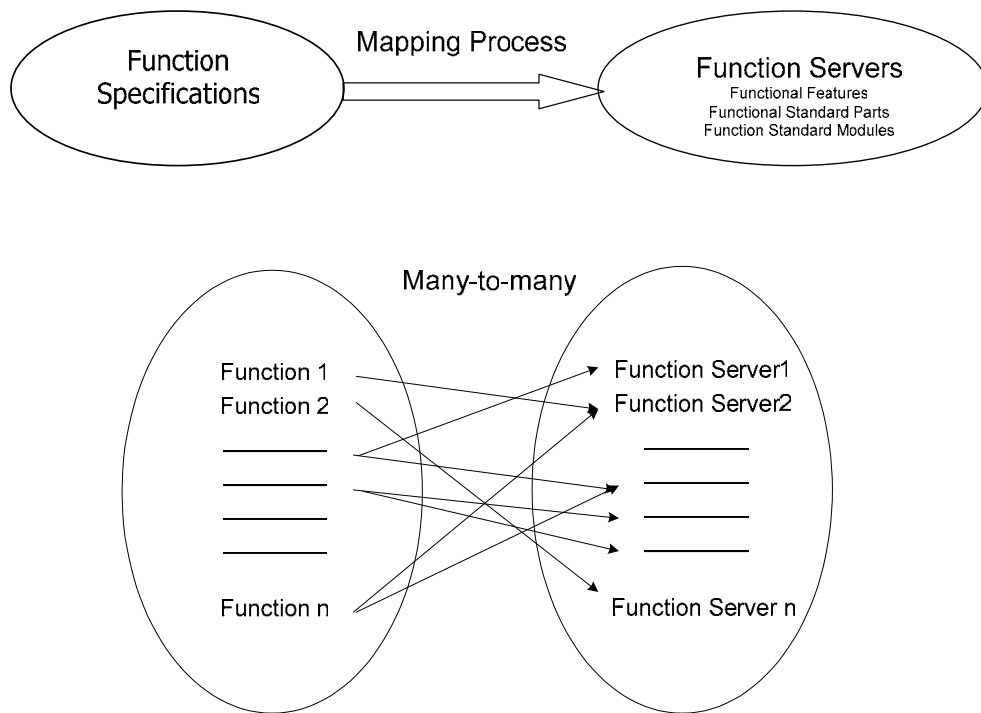
The functional decomposition model and the interaction with potential function servers, which satisfy the functions, are illustrated in Figure 3.3. This model expresses how functions are broken into sub-functions and into sub-sub-functions until the functional decomposition is fully defined. A complete function breakdown would consist of all functions, first without forms (i.e., function servers) and then with the corresponding form. Many iterations on the function structure are made from the initial functional breakdown, before any function server is applied to the final structure with all the levels developed. The function/function server model represents the product structure at a conceptual stage.



**Figure 3.3 Function/function server decomposition structure**

The mapping process between functions and function servers is not straight forward. The same function could be performed by several different function servers and a given function server could be used to perform different functions. The relationship between function and function servers has the nature of many-to-many, as shown in Figure 3.4. Therefore, the selection process should be based on an evolution of each solution to some criteria. The criteria can consist of any of the following: usage of function server, type of application, boundary

conditions, initial constraints on the system, introduced constraints due to the relationships with adjacent objects, or the actual functional problem under investigation. All of these criteria come under a product context that captures the application environment of that product. The following presents a formal definition of the function-to-function server mapping process that is adapted, with modifications, from Wang and Nnaji [74].



**Figure 3.4 Function-to-function server mapping process**

Formally  $F$  is defined as the set of functions to be satisfied, where  $f$  is an element of  $F$  ( $f \in F$ ). Also,  $FS$  is defined as the set of function servers, where  $fs$  is an element of  $FS$  ( $fs \in FS$ ). The mapping process from function-to-function servers, which is the process of function server retrieval, is defined as:

$$P: F \rightarrow FS$$

The mapping from function servers to functions, which is the process of function retrieval, is defined as:

$$P': FS \rightarrow F$$

Also,  $C$  is defined as the set of contexts, which captures the application environment, where  $c$  is an element of  $C$  ( $c \in C$ ). Then a restricted mapping that reduces the search space is:

$$P_C: F \times C \rightarrow FS$$

Similarly,

$$P'_C: FS \rightarrow F \times C$$

Function level is used also to reduce the search space. A function can be at an assembly/subassembly level, a part level, or a geometric level. Assembly level functions target functional module parts. Part level functions target functional standard parts. Geometric level functions target functional features. Formally,  $L$  is defined as the set of function levels, where  $l$  is an element of  $L$  ( $l \in L$ ). A more restricted mapping is then defined as:

$$P_{CL}: F \times C \times L \rightarrow FS$$

Similarly,

$$P'_{CL}: FS \rightarrow F \times C \times L$$

The function-function server relations are represented by the relation matrix  $R = [r_{ij}]$ , where:

$$r_{ij} = \begin{cases} 1 & \text{if there is a relation between server } i \text{ and function } j \\ 0 & \text{otherwise} \end{cases}$$

In a full matrix notation,  $R$  is given by:

$$\begin{array}{c}
 F_1 \\
 F_2 \\
 \cdot \\
 \cdot \\
 \cdot \\
 \cdot \\
 \cdot \\
 \cdot \\
 F_n
 \end{array}
 \begin{array}{cccccc}
 FS_1 & FS_2 & \cdot & \cdot & \cdot & FS_m \\
 \left[ \begin{array}{cccccc}
 r_{11} & r_{12} & \cdot & \cdot & \cdot & r_{1m} \\
 r_{21} & r_{22} & \cdot & \cdot & \cdot & r_{2m} \\
 \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 r_{1n} & r_{2n} & \cdot & \cdot & \cdot & r_{nm}
 \end{array} \right]
 \end{array}$$

Where,

$F_i$  ( $i=1, 2, \dots, n$ ) are the functions

$FS_j$  ( $j=1, 2, \dots, m$ ) are the function servers.

### 3.5 CONCEPTUAL PRODUCT MODEL

Primitive product information, which includes functional data and physical parts (i.e., function servers) information, are organized and represented by a data structure called Conceptual Product Model (CPM). CPM is developed in this work to document the product information in both the functional and the physical domains and to support the conceptual design phase. This model provides the tools for managing the huge amount of design information and for allowing the designer to access design information from different views. The representation of the conceptual product consists of information from both the functional data model and the

function server model, as well as the relationship between them. According to this description, CPM is defined as:

$$\mathbf{CPM} = \{\mathbf{F}, \mathbf{FS}, \mathbf{P}\}$$

Where,

**F:** functionality model.

**FS:** function server model.

**P:** mapping process between functions and function servers.

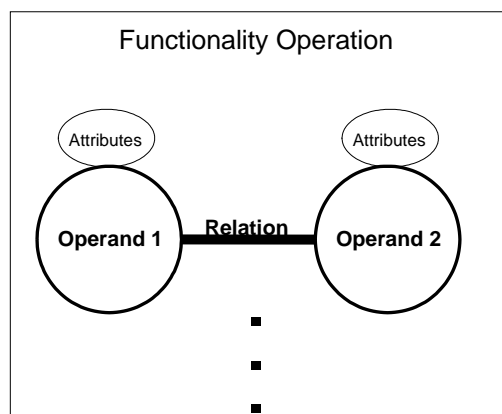
The CPM is implemented in an object-oriented environment using the Unified Modeling Language (UML). The implementation details are presented in chapter 6.



## 4.0 FUNCTIONAL DATA MODEL

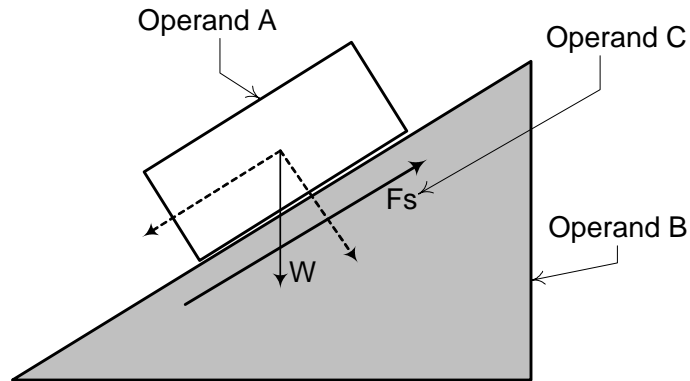
### 4.1 FUNCTIONALITY OPERANDS

An operand is a distinct element involved in the realization of a given functionality. Operands are the building blocks that come together to accomplish any mechanical functionality. They can be classified into two broad categories: material and energy operands. A material operand is any tangible physical entity that has some mass and volume. An energy operand, on the other hand, is the functionality component that affects some work outcome<sup>[29]</sup>. However, the realization of any functionality must contain, in addition to the operands, a description of the relation between the operands. The existence of an operand alone will not yield any meaningful functionality. Hence, operands are the active factors brought together by a relation for the actualization of a given task. Figure 4.1 illustrates the components of a functionality operation.



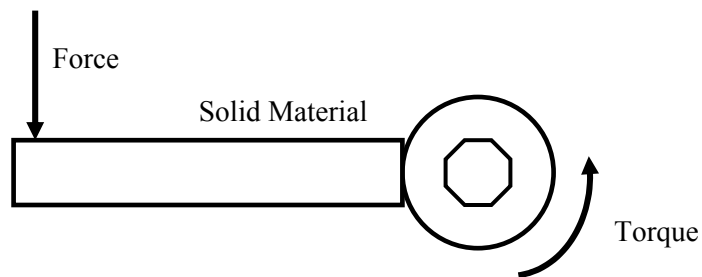
**Figure 4.1 Components of a functionality operation (operands and relations)**

To illustrate the interaction between functionality operands, consider the functionality operation shown in Figure 4.2, friction on the lower surface of a block by a wedge. Operands A, B, and C interact to realize the given functionality operation. Operands A and B are solid material operands, while operand C is a force operand.



**Figure 4.2 Friction functionality operations showing the interaction between functionality**

The functionality operation—transform force to torque—is illustrated in Figure 4.3. In this operation, three operands interact: force, torque, and the medium (solid material).



**Figure 4.3 Force-to-torque transformation operation**

#### 4.1.1 Solid material operand

A material operand is any tangible physical entity that has some mass and volume. There are three classes of materials: solids, liquids, and gases. In this research, only solid material operands are considered. The principal components of solid material operands are: functional markers, physical properties, degree of freedom, mass property, material type, and operand role. In accordance with this, solid material operands are modeled as:

**SM:** {**FM**, **PyC**, **DoF**, **MP**, **MT**, **OR**}

Where,

**FM:** the set of functional position markers.

**PyC:** the set of physical characteristics of the operands.

**DoF:** the required degree of freedom of the operands.

**MP:** the set of mass properties on the operands.

**MT:** the material type.

**OR:** operand role.

The following paragraphs explore these solid material attributes.

##### Functional position marker

Functional position markers are the regions/points of contact between solid operands. They are geometric points, locations, or regions on solid operands. Each functional position marker is described by two types of characteristics:

- Intra-functional marker characteristics: This describes the geometric shape and size, location, orientation, tolerance, and dimension.

- Inter-functional marker characteristics: This describes the dependences between the functional markers within a solid operand, such as position of functional point, orientation, feature adjacency, etc.

### Physical characteristics

Physical characteristics define the physical composition of the solid operand. This set of attributes includes material strength, elasticity, ductility, density, surface characteristics, thermal conductivity, thermal expansion, specific heat capacity, and electrical resistance. Surface characteristics are surface friction, surface roughness, and wear rate. The strength attribute measures the stress (load per unit area) the operand is able to withstand under operation conditions. The types of stress include tensile, compressive, torsion and bending stress.

### Degree of freedom

The degree of freedom of a solid operand defines the motion the operand may undertake during the functionality operation. The relative position of one solid operand to another is defined by spatial relationships. Therefore, each spatial relationship can be interpreted as a constraint imposed on the degrees of freedom between relative mating or interacting features.

The types of degrees of freedom are classified as follows <sup>[51, 75]</sup>:

- Lin-n: linear translation along n axis.
- rot-n: rotation about n axis.
- cir-n: translation along a circle with n axis.
- Plan-n, syl-n, sph: translating along a planar, cylindrical, and spherical, respectively.
- rol-lin-n: rolling along a corner.

The degrees of freedom of a solid material operand are expressed as: {degrees of freedom of moving along the functional feature of relative coupling operand: degrees of freedom of moving the operand with respect to itself}, where the relative coupling operand is fixed. More details about degree of freedom can be found in Liu <sup>[51]</sup>.

### Material type

Engineering materials can be broadly classified into five groups: metals, plastics (polymers), ceramics, composites, and wood. The material performance requirements of a solid operand are essential for choosing a potential material type. Material performance requirements include: functional requirements, processability requirements, cost, and reliability. Figure 4.4 illustrates these requirements.

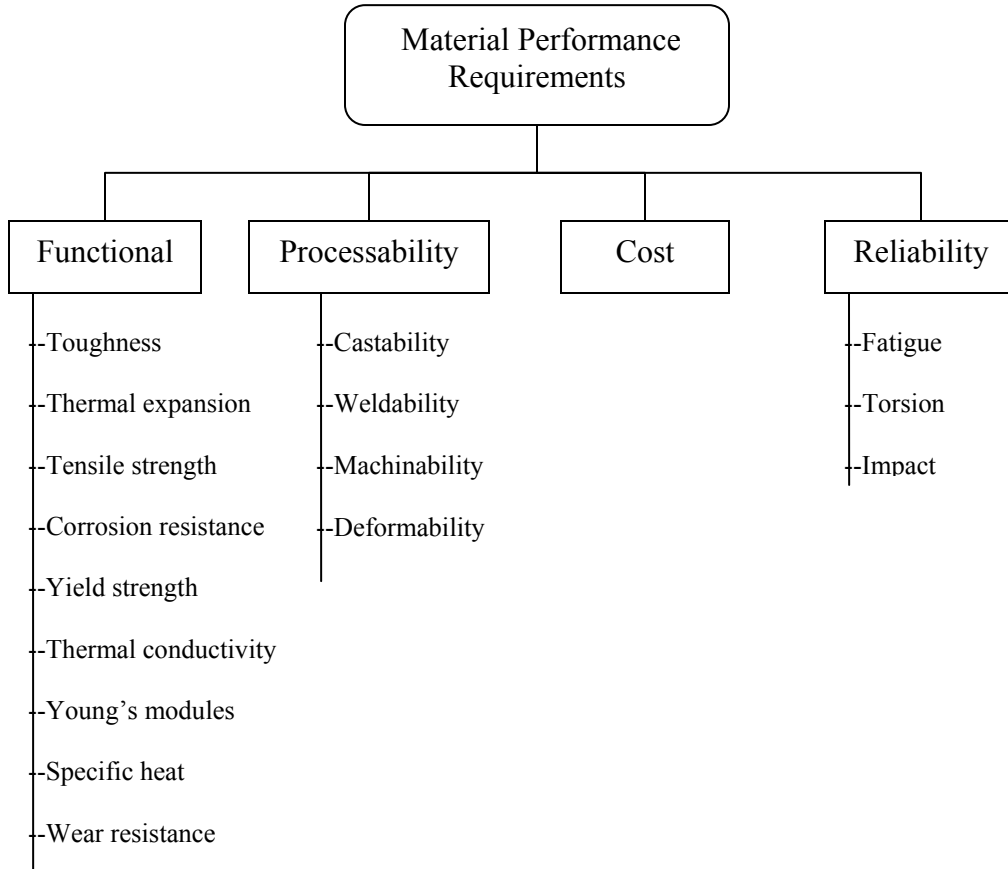
Functional requirements are the characteristics of the product required to do its expected work. The processability requirement is the material's ability to be worked and shaped by different manufacturing processes. The reliability requirement is the probability that the new product will work without failure during its expected life.

### Mass properties

Mass properties of a solid material include its volume, mass, area, and moment of inertia.

### Role

A solid operand can be a function performer, which introduces the effect of the function, or a function receiver, which receives the effect of the function. In our friction example above, the wedge is the function performer and the block (operand A) is the function receiver.



**Figure 4.4 Material performance requirements**

#### 4.1.2 Mechanical energy operand

Energy is the functionality operand that affects some work outcome. Energy can be in different forms: mechanical, thermal, electrical, magnetic, acoustic, radioactive, chemical, biological, optical, hydraulic, or pneumatic. In this dissertation, only the mechanical energy form is considered. Mechanical energy is always associated with the displacement of a material operand or with strain energy associated with loading. Two forms of energy operands, force and torque operands, represent the mechanical energy in the operation of mechanical systems. The

force operand is the primary mechanical energy source in a mechanical system. The torque operand, on the other hand, is secondary, as it is derived from the force operand.

In mechanical systems, a variety of forces are always acting on material operands. Interaction forces are the individual forces acting on an object, when other objects are involved in this interaction. In this research, the net force on an object (i.e., the sum of all interacting forces) is computed for modeling purposes. The modeling of a force operand requires information about the force source, nature, kind, magnitude, and point of application. The force operand is defined as <sup>[29]</sup>:

**FORCE :{ < source | kind > < mag, angle, point > < nature > }**

#### Kind of force

Kind of a force defines the contact between the object exerting the force and the object receiving it. Therefore, force is classified into two kinds: contact and action-at-a-distance (or non-contact) force. Contact force applies when there is direct contact between the objects—for example, the force of a man’s hand on a box when he pushes it. Non-contact (or action-at-a-distance) force applies when the object exerts a force without appearing to contact the other object. This includes the use of gravitational, electrical, and magnetic force.

#### Magnitude, angle, and point of application

Magnitude, angle, and point of application define the quantitative composition of the force operand. They introduce the effect of the force of other operands involved in a given functionality. Magnitude represents the quantitative resultant of all interacting forces. Angle

represents the direction of the force. Point of application represents the exact spatial relationship between the force operand and the solid material operand.

### Nature

Nature represents the force form with respect to time. According to nature, forces are classified as:

- Static force: Force is gradually applied and equilibrium is reached in relatively short time.
- Sustained force: Force is constant over a long time.
- Impact force: Force is applied and removed rapidly.
- Cyclical force: Force changes in magnitude or direction with time.

Torque energy is the energy that results from the rotation or torsion of a solid material operand. Torque, which is a force analogy for rotational motion, is the effectiveness of a force in bringing about changes in rotational motion. Torque depends on the applied force and how far it is applied from the rotation axis. The representation of the torque operand is similar to that of the force operand. The torque operand is modeled as <sup>[29]</sup>:

**TORQUE: {<source | kind> < mag, angle, axis > < nature >}**

Source attributes define the source of the force that produces this torque. Kind attribute specifies whether the source force is contact or non-contact. The angle attribute is the rotation from the Cartesian axis, and the axis is the axis of rotation.



## 4.2 FUNCTIONALITY RELATIONS AND STATES

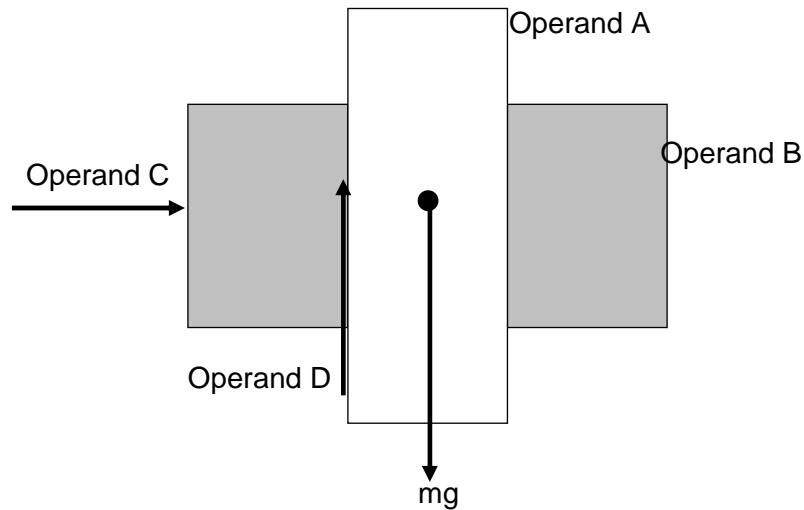
The functionality relation defines the relationship between functionality operands in the performance of the functionality operation. Defining functionality operands alone will not realize the functionality operation. Both operands and relations are required to accomplish and completely realize a functionality operation. After each operand has been defined with its associated attributes, relations are built between those attributes. As a result, the functionality relation defines how the attributes of functionality operands are related.

For example, consider the following functionality operation: hold a block (solid material) between two jaws (solid materials) of a machine's vise. The frictional coefficient, stress, and surface wear rate are some of the relations that relate the attributes of the three solid operands in this example. The solid operands are the block and the two jaws (operands A and B, if we consider the jaws as one piece), as shown in figure 4.5. The force operand is the applied holding force (operand C) and the friction force (operand D). The functional regions of interest are the contact surfaces. The other attributes that are relevant in this interaction are the surface characteristics (which determine the coefficient of friction,  $\mu$ ), the magnitude and nature of the applied force C, and the physical characteristics of the block A (such as strength, elasticity, and ductility). The friction force D is given by:

$$F_D = \mu \cdot F_C$$

Where  $\mu$  is the coefficient of surface friction and  $F_C$  the magnitude of applied force C, which represents the normal force of the contact surface. The relative position of the surfaces in contact is controlled by the spatial relationship. The two surfaces of the block must maintain a

relationship with the two surfaces of the vise. Hence, both solid operands A and B are fixed with no degree of freedom.



**Figure 4.5 Example of hold functionality operation**

Relations in a functionality operation are defined by the following components: the coupling operands that have a relationship and their attributes, functional relations, constraints, and the degree of freedom. Therefore, relations can be modeled as:

**Relation: {<coupling operands, attributes> <functional relation> <constrains> <DoF>}**

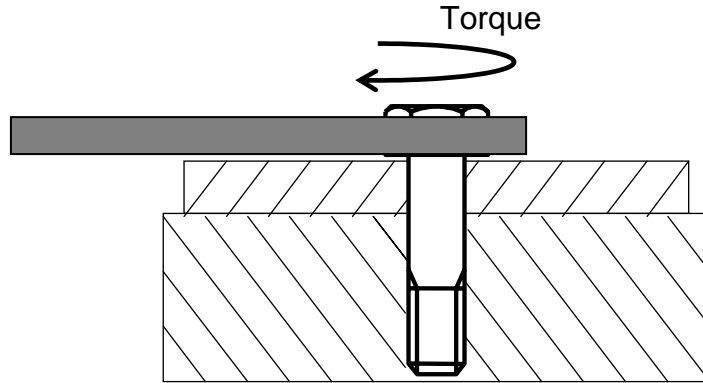
In this model, functional relations capture the relation between operands (or attributes of operands) according to physical laws or spatial relationships. These relations must be maintained between attributes of the interacting operands to satisfy the given functionality. Newton's law, wear rate, stress, and strain, are examples of physical laws that govern the relation between

operands. A spatial relationship represents the relative location of objects in space. Six types of spatial relationships are introduced by Liu <sup>[51]</sup>: against, parallel-offset, include-angle, parax-offset, aligned, and incline-offset.

Constraints are the limits imposed on the operand attributes or on the relations between operands. For example, the maximum stress applied on a solid operand could be required to be less than 3000 N/m<sup>2</sup>. The desired degree of freedom to be maintained by the involved operand can be one of the following types: lin-n, rot-n, cir-n, plan-n, cyl-n, sph, and rol-lin-n.

During any functionality operation, functionality operands can move from one state to another. The transition can be determined by computing the change in the operand's attributes. Therefore, a state can be defined as an instant manifestation of the functionality operands' condition during the satisfaction of a functionality task. The transition may be either continuous or discrete. In this work, discrete approximation is assumed to reduce the complexity of some continuous systems. As a result, only the upper and lower values for each operands' attributes are considered.

For an example of the functionality states in a functionality operation, take the unfasten operation shown in Figure 4.6. Here, the torque applied by a wrench on a bolt changes from Max-Torsion condition at the beginning of the operation to Min-Torsion or No-Torsion condition at the end of the operation. Hence, the torque magnitude is defined by two states: Max-value state and Min-value state. In addition, the bolt position is transmitted from initial-position state to final-position state.



**Figure 4.6 Example of unfasten a bolt**

### **4.3 MECHANICAL FUNCTION CLASSIFICATION BASED ON OPERANDS INTERACTION**

The classification of mechanical functions has been attempted by some researchers [2, 5, 22, 23]. In their work, the categorization of mechanical functions was constructed in an inductive fashion. These researchers tried to comprehensively describe all functions used in designed space by grouping them into classes and sub-classes. Each class or sub-class contained functions with some similarities in their effects. However, this inductive method makes it difficult to argue that the list is complete. Moreover, it is impossible to count *a priori* all possible functions in design space. As a result, this dissertation introduces a new classification scheme for mechanical functions. This classification scheme is based on the interaction of the functionality operands (i.e. material and energy). As stated before, the functionality operands in any functionality operation interact with each other to achieve the desired task. This interaction of functionality operands leads to a small yet complete taxonomy of basic mechanical functions. Table 4.1 shows the basic function classes and the corresponding functions of each class based on the interaction

between solid material, force, and torque. This classification scheme can be extended to include all possible interactions between other types of material and energy operands. The corresponding functions list can be extended too.

**Table 4.1 Classification of basic mechanical functions**

Function class	Examples
Class I : [Solid materials interaction]	Position, enclose, contain, block, space, align, seal, strengthen, mate.
Class II: [Solid material & Force interaction]	Support, transmit force, transmit material, store, control, friction, eject, hold, join, allow passage, guide, mount.
Class III: [Solid material & Torque interaction]	Support, transmit torque, transmit material, store, control, hold.
Class IV: [Solid material, Force & Torque interaction]	Transform, friction.

Class I includes the functions that arise when solid materials operands interact. The solid materials in this interaction are assumed to have negligible weight. For instance, the function of container, cover, and housing is to contain and enclosed other solid materials. The function of key, pin, slot, and hole is to locate and position other solid materials.

Class II includes the functions that arise when solid materials and forces interact. Most mechanical functions come under this class. Forces can have many types of effects on solid materials, such as transmit, hold, join, friction, or guide. On the other hand, solid materials can support, transmit, or store forces. For example, if a claw hammer is used to pull nail out of a piece of wood, the human force is first transmitted though the hammer (solid material) to the head of the nail. This force will then cause the nail to move outward. Bearing, roller, wheel, wall, rod, and pin are example of solid materials that support force. Beam, rod, and cable are examples

of solid materials that transmit force. In addition, forces can be stored in a spring, which, if released, can be transferred to other solid materials.

Class III includes the functions that arise when solid materials and torques interact. Similar to class II, torque can transmit, control, or hold solid materials. On the other hand, solid materials can support, transmit, or store torque. For example, the torsion of a wrench causes a bolt to move outward and the pipe to rotate. Pump spindle transmits torque, while a fly wheel stores torque. Single journal bearing, single thrust bearing, single smooth pin, and single hinge all support torque.

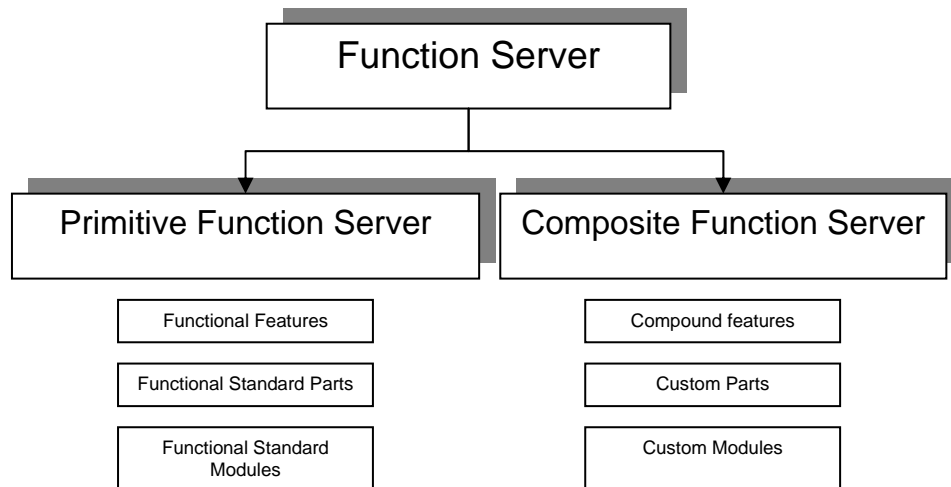
Class V includes the functions that arise when there is an interaction between solid materials, force, and torque. Wrenches, gears, and shafts are examples of solid materials in which force is transformed into torque. In addition, the rotation of a cylinder on a flat solid material can cause a friction force on both surfaces.

The advantage of this classification scheme is that it offers the designer a convenient guide for constructing the function composition and for determining the stopping point during decomposition. Also, the result of the function decomposition will effectively reflect the functionality operations that compose the overall function of the mechanical product. Furthermore, basic functions may have a list of associated function servers that can help the designer create the mapping between them.

## 5.0 FUNCTION SERVER REPRESENTATION AND MODELING

### 5.1 FUNCTION SERVER CLASSIFICATION

Function servers are the conceptual physical elements that possess functionality. Functional specifications can be satisfied by individual or by combined function servers. Function servers are then the sub-solutions of the considered sub-functions, which, when arranged and combined, form the conceptual form of the product (i.e., the overall solution). Function servers are classified into two groups: primitive function servers and composite function servers. Figure 5.1 illustrates this classification.



**Figure 5.1 Function server types**

Primitive function servers are the basic physical elements located at the lowest level of abstraction in the hierarchical composition of the product's conceptual form. Each primitive function server can perform one or more functions that contribute directly or indirectly to the achievement of the overall product function. The composite function servers are those physical elements that are constructed by the combination of two or more primitive function servers and that possess functionality. Their functionality derives from the contribution of the primitives.

The primitive function servers are classified into three groups as shown in Table 5.1. These groups are as follows:

- **Functional features:** The most primitive design structure that can comprise a component (i.e., functional form features). They can possess a variety of functionalities. They are classified to five sub-classes: walls, additives, subtractive, beside-walls, and solid elements. A wall is an area or surface of a pre-existing shape and can be of various kinds, such as flat or curve. An additive feature is a feature that extends outward from the shape; examples are bosses, ribs, tabs, and flanges. A subtractive feature is a feature that extends inward from the shape or that goes through the shape; examples are holes, slots, grooves, pockets, and steps. A beside-wall feature is a feature that connects two elements of a pre-existing shape; examples are chamfers, fillets, ribs, undercuts, and bridges. A solid element is a solid feature; examples are blocks, cylinders, spheres, and disks.
- **Functional standard parts:** The parts manufactured in quantity for use in a number of products. These possess some specific functionality. Examples are screws, bolts, nuts, springs, rackets, beams, and shafts.



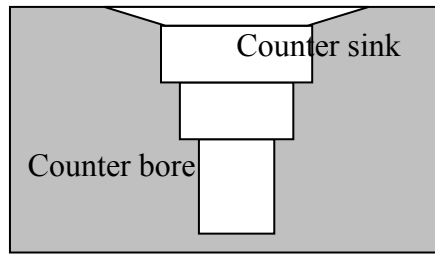
- Functional modules parts: The assemblies or sub-assemblies that are manufactured in quantity for use in a number of products and possess some specific functionality. Examples include motors, pumps, and clutches.

**Table 5.1 Classification of primitive function servers**

<b>Primitive Function Servers</b>		
<b>Class</b>	<b>Sub-class</b>	<b>Examples</b>
Functional Features	Walls	Flat , curve, surface
	Additives	Boss, web, gusset, rib, protrusion, tab, flange.
	Subtractives	Hole, slot, pocket, groove, indent, depression, step, corner.
	Beside walls	Chamfer, fillet, groove, undercut, rib, web, bridge.
	Solid elements	Block, cylinder, rod, tube, sphere, disk, ring.
Functional Standard Parts		Screws, bolts, nuts, springs, brackets, beams, shafts.
Functional Module Parts		Motors, clutches, switches, pumps, bearings.

Composite function servers are also classified into three groups:

- Compound features: A union of one or more features to create a more complex feature definition. For example, a compound feature can be defined by the combination of a counter bore hole and a counter sink hole as illustrated in Figure 5.2.
- Custom parts: Parts that are designed and manufactured especially for use in a particular product. They are built from the combination of one or more functional features.
- Custom modules: Assemblies or sub-assemblies that are designed and manufactured especially for use in a particular product or to serve some new functionality description.



**Figure 5.2 Example of compound feature**

## **5.2 FUNCTION SERVER REPRESENTATION**

The function server representation model consists of six basic components: functional description, shape description, material model, manufacturing information model, interface representation, and working conditions model. The function server model possesses a comprehensive information structure for maintaining detailed data information about function server attributes for use in the conceptual design phase and in the detailed design phase. Each function server maintains information as to type, functionality, abstract shape, application domain, and manufacturing process domain. In addition to this information, a description or analysis of the designer's reasoning and intent through the satisfied functionality operation can be added and preserved. This information can include the considered functionality operation, interfaces with other function servers, material type, manufacturing process by which it should be made, working environment, and constraints imposed on it. As stated earlier in chapter 3, the Function Server (**FS**) is modeled as:

**FS: {FD, SH, M, MI, IN, WC}**

Where,

**FD:** functionality description.

**SH:** shape description.

**M:** material model.

**MI:** manufacturing information.

**IN:** interface description.

**WC:** working condition.

### Functionality description

A function description includes a listing of all primary and secondary functions that a function server can satisfy. For example, the function of a motor is to convert electrical energy to rotating mechanical energy. A gear's function is to change speed of rotation and transmit torque. A wall can block a motion, support a load, or position an object.

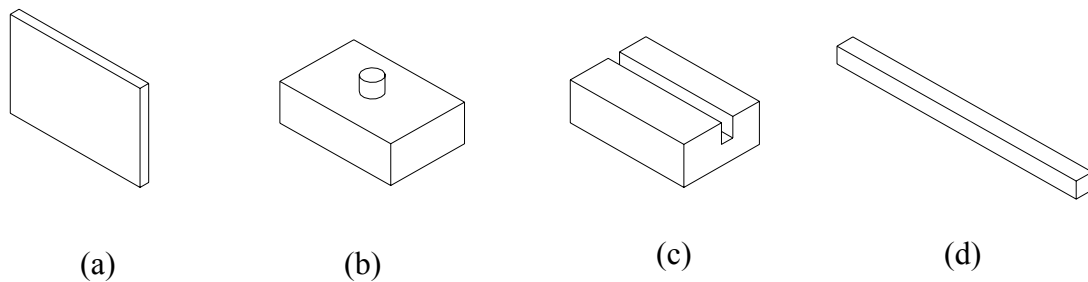
Beside the primary and secondary functions, the functionality description includes a listing of all function server participants, which are the other function servers that are required for the function server to successfully complete its task. For example, a bolt's participant list includes a hole, a nut, and a washer.

### Shape description

Shape description includes the base shape and the abstract shape of the function server. The base shape is the initial shape of the material before machining. It represents the general shape form. This base shape can assume any of the following representations:

- Block-base-shape: Initial shape of the material is a rectangular cross section.
- Cylindrical-base-shape: Initial shape of the material is circular cross section.
- N-polygon-base-shape: Initial shape of the material is a polygon with n number of sides.

The abstract shape is the conceptual schematic representation of the actual shape of the function server. Abstract shapes help the designer to build the conceptual form of the product. Detailed shape representation that includes dimensions and tolerances is completed in the detailed design stage. Figure 5.3 shows the abstract shapes of some functional features. Appendix A contains the abstract shapes of all the functional features considered in this study.

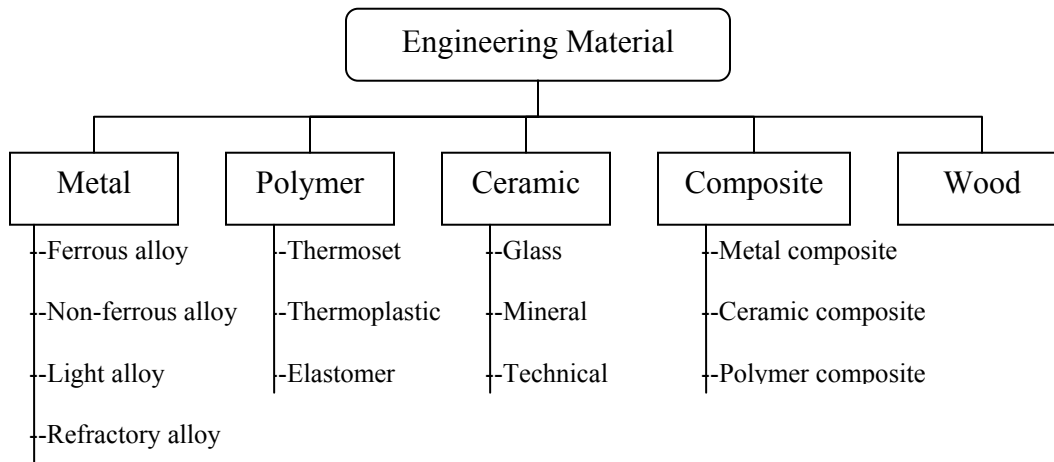


**Figure 5.3 Example of abstract shapes of some functional features: (a) wall, (b) boss, (c) slot, (d) rod**

### Material model

The material model includes information about material class or sub-class and the material properties for the function server. Materials are broadly classified into five groups: metals, plastics (polymers), ceramics, composites, and wood. Figure 5.4 shows this classification. Material properties are divided to mechanical properties, physical properties, and

chemical properties. Mechanical properties include: strength, ductility, elasticity, fatigue resistance, creep resistance, corrosion resistance, hardness, toughness, and fracture. Physical and chemical properties include: density, specific heat, melting point, thermal conductivity, thermal expansion, oxidation, and magnetic properties.

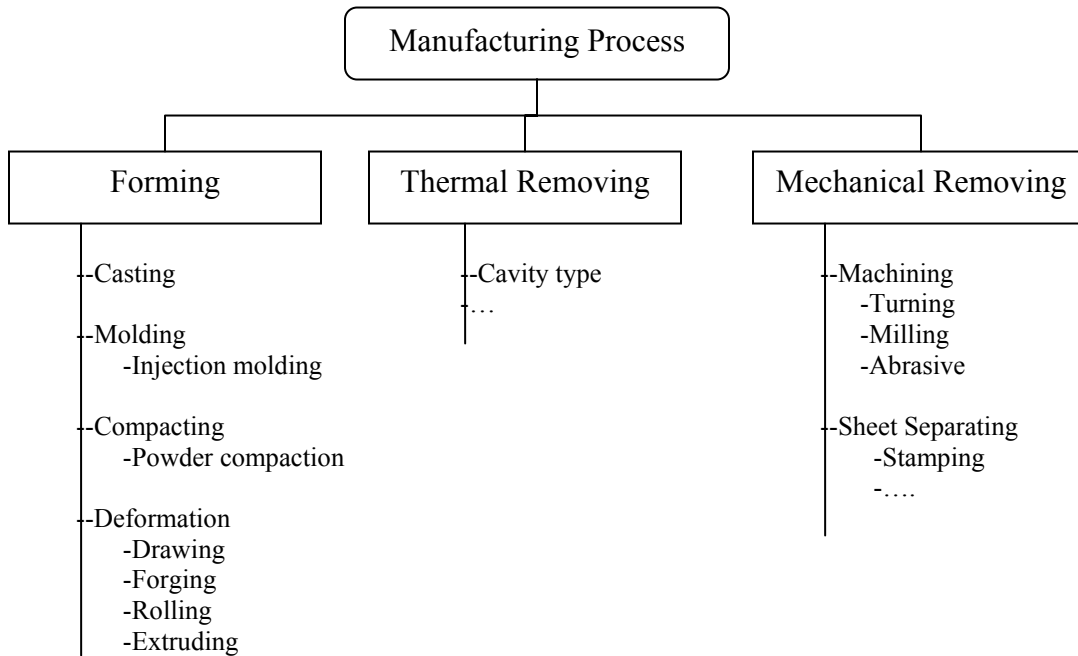


**Figure 5.4 Engineering materials**

Manufacturing Information

The manufacturing information module includes information about manufacturing processes, surface properties, material properties, and alternate materials. Manufacturing processes are the operations for manufacturing or changing the function server. Manufacturing processes are divided into three types: mechanical removing, thermal removing, and forming (see Figure 5.5). Surface properties are the characteristics of the function server surface

including roughness, flatness, texture, and surface finish. Alternate materials are allowable secondary materials that may be used if the primary material is not available.



**Figure 5.5 Manufacturing processes**

Interface Description

The interface description includes information about the type of interface, special relationships, and the degree of freedom. The type of interface can be rigid or flexible. A rigid interface implies that the interacting function servers are fixed relative to each other with zero degrees of freedom. A rigid interface can be permanent, such as a riveted joint, or temporary, such as two pieces of sheet metal connected with a bolt and nut. A flexible interface allows one

or more degrees of freedom for at least one of the interacting function servers. For example, simple contact is a flexible interface.

Spatial relationships represent the relative location and orientation of objects in space. Six types of spatial relationships are introduced by Liu <sup>[51]</sup>: against, parallel-offset, include-angle, parax-offset, aligned, and incline-offset. These relationships can be interpreted as constraints imposed on the degree of freedom between interacting function servers. The degree of freedom can assume any of the following types: lin-n, rot-n, cir-n, plan-n, cyl-n, sph, or roll-lin-n. A function server's degree(s) of freedom are expressed as: {Degrees of freedom of moving along the functional feature of relative coupling function server: degrees of freedom of moving the function server with respect to itself}, where the relative coupling of function server is fixed.

### Working conditions

Working conditions represents the service environment and the working requirements that enable the function server to achieve its intended function. The working conditions include physical phenomena, energy type, load nature, and failure modes. The physical phenomena represent the physical effect or the physical law that manifests the required function. These physical laws include Newton's laws and conservation laws, such as conservation of mass or conservation of energy. Examples of physical effects include friction, stress-strain, creep, fatigue, thermal expansion, absorption, diffusion, and buoyancy effects. More information about physical laws and effects can be found in Hix <sup>[76]</sup>.

Energy type represents the energy domain by which the function sever is able to affect some work outcome. The energy type can be mechanical (rotation or translation), electrical, thermal, magnetic, acoustic, radioactive, chemical, biological, optical, hydraulic, or pneumatic.

In this research, only the mechanical energy type is considered. Load nature represents the nature of the force applied to the function server. The load nature can be a static, sustain, impact, or cyclic force.

Failure modes are the physical processes that produce a function server failure during its operation. This information is commonly collected from past experiences and previous designs. Mechanical failures may be defined as any change in the size, shape, or material properties of a structure, machine, or machine component that renders it incapable of performing its function [77]. The main failure mode set includes, in part, elastic deformation, yielding, fatigue, corrosion, and wear. The complete list can be found in Collins [77].



## **6.0 IMPLEMENTATION AND CASE STUDY**

The implementation of the function specifications-to-conceptual form translation tool consists of a number of computer models, data structures, and computer tools. These items are developed to validate and demonstrate the concepts proposed in this work. The implementation must support all of the activities included in the translation tool. This includes functionality modeling, function server modeling, a function driven database structure, a searching mechanism, and conceptual product modeling. Furthermore, the implementation should ultimately guide and help the designer to manage and display the information needed to translate function specifications into conceptual form. This requires an advanced graphical user interface to facilitate information displaying and manipulating.

For testing and validation, the implemented computer tools are applied to the design of a real world product, a toggle clamp. This case study illustrates how the designer would interact with the system when engaged in function driven design. The following sections clarify the implementation details.

### **6.1 DATA STRUCTURE OF THE TRANSLATION TOOL**

The structure of the translation tool's components is based in an object-oriented approach. Object-oriented programming is a rapidly maturing technology that possesses a number of advantages over traditional programming. An object-oriented system attempts to

model the real world <sup>[78]</sup>. This type of system uses a natural model, concurrent processes, and multiple controls. In addition, it presents many new techniques and algorithms for the development and implementation of current mechanical design methodology. The data in object-oriented programming is quantized into discrete, distinguishable entities called objects. Each object has its own inherent identity. Objects with the same data structure (attributes) and behavior (operation) are grouped into a class. Each class describes a possibly infinite set of individual objects. Each object is said to be an instance of a class. Each instance of a class has its own values for each attribute, but they share attribute names and operations with the other instances of the class.

This dissertation employs the object-oriented Unified Modeling Language (UML) to construct object models. These models are used to describe the static structure of the translation tool objects and their relationships. This includes the functionality object model, the function server object model, and the conceptual product object model. Each object model is graphically represented with an object diagram. An object diagram is a graph whose nodes are object classes and whose arcs are relationships among classes.

### **6.1.1 Functionality Object Model**

The class diagram of the functionality object model developed in this work is shown in Figure 6.1. The components of this functionality object model are: function operation, operand, relation, state, and constraint. These components are described below.

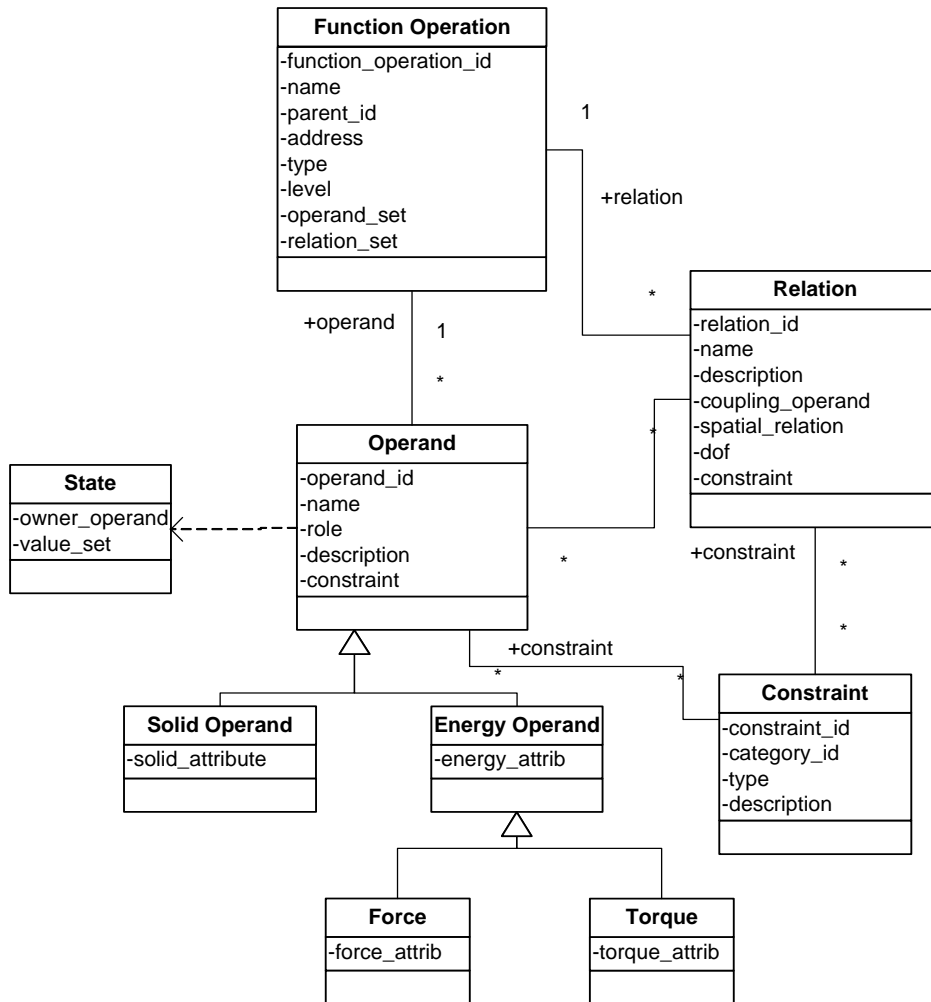


Figure 6.1 Data structure of functional model

### 6.1.1.1 Function Operation Class

The function operation class defines the function to be realized. It acts as a header that provides a reference point for other objects. The attributes of this class are: name, id, parent-id, type, address, and level.

#### ID and Name

The attribute id is a unique identifier for function operation. It is used to reference an operation by other objects. The name attribute is the name of the function to be satisfied. It could be user assigned based on some naming convention in an application domain. Previous researchers <sup>[19]</sup> proposed verb/noun pairs for the function name. This research follows this convention: the verb is the function name and the noun is the operand that is manipulated by the function. For instance, consider the function operation: guide the shaft motion. The function name is guide, and the operand is the shaft. This method of naming gives the designer more flexibility in using function names to build the required function operation.

#### Parent-ID

The parent id attribute is the unique identifier for the function parent in the function breakdown. It is used to capture and track the parent-child relation in the function breakdown structure.

#### Type

The type attribute represents the kind of task needed by this function operation. The types are divided into performance (design), assembly, manufacturing, safety, or aesthetic.

## Level

Mechanical functions can be performed by all the components of the product, by some of its components, or by certain geometric features of components. Therefore, the function level attribute denotes the level of the function according to the targeted function server. The function can belong to one of the following levels:

- Overall functions: These represent the overall functionality of a mechanical product. For example, the overall function of a car is to transport people from one place to another. Overall functions are defined at the most general level.
- Elementary functions: These represent the decomposition of overall functions (sub-functions) such that no direct mapping to physical elements can be done. As a result, further decomposition has to be undertaken. Some of the functions at this level can move to the embodiment level if a direct relationship to a physical element is found.
- Embodiment functions: These represent the lower level of the function decomposition. They are mapped to some physical embodiment. This level is divided to three sub-levels:
  - Assembly level functions: They target the function servers that can be standard modules or custom sub-assemblies.
  - Part level functions: They target the function servers that can be standard parts, custom parts, or solid elements.
  - Geometric level functions: They target the function servers at the geometric level (i.e., functional features).

### Address

The address attribute specifies the exact location of the function operation in the function decomposition structure. The address of a specific function is expressed as:

$$\text{Address} = (i,j)$$

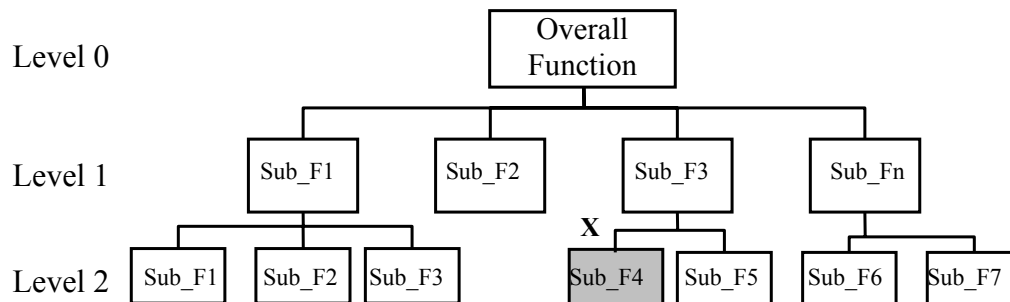
Where,

$i$  = level number 0, 1, 2...  $n$

$j$  = position in that level

$n$  = total number of levels

For example, the address of the function denoted by X (Sub-F4) in Figure 6.2 is: (2,4).



**Figure 6.2 Example of function decomposition to illustrate the function address attribute**

### **6.1.1.2 Operand Class**

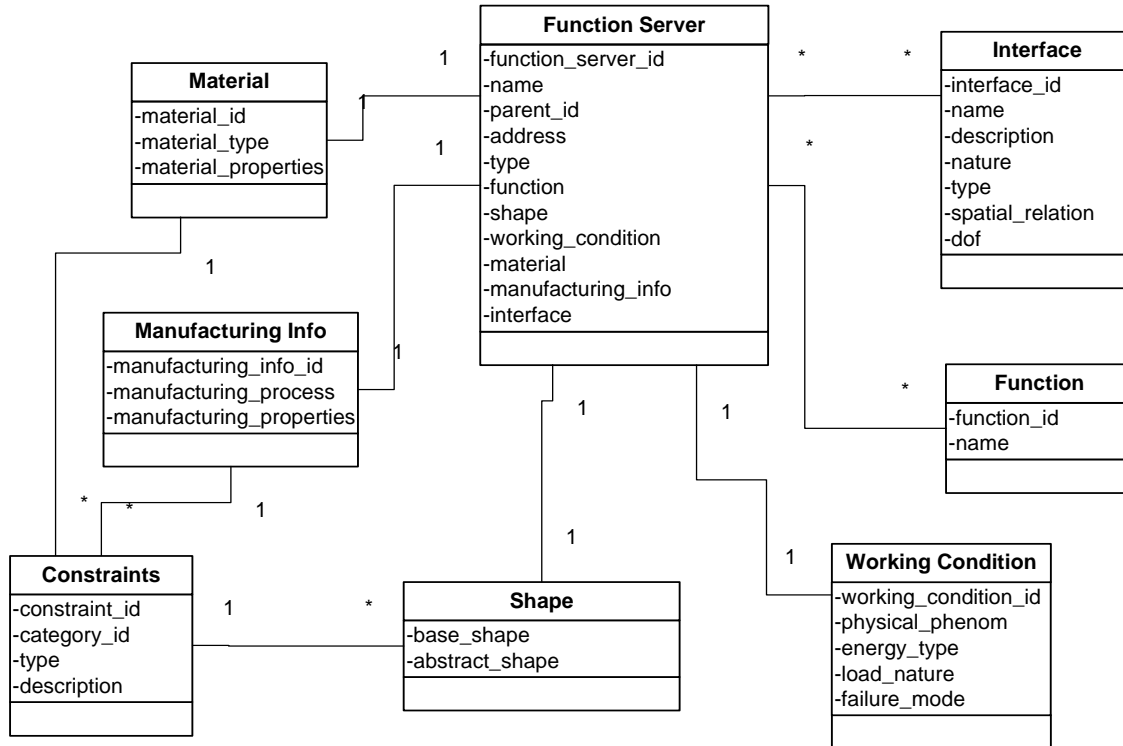
The operand class represents all of the functionality operands that interact to achieve the function operation. In this dissertation, operands are grouped into two categories: solid material and energy (force and torque). Each operand has its own id, name, and role description. The role attributes indicate if the operand is a function performer or a function receiver. Solid material, force, and torque operands have their own attributes. These attributes are explained in detail in chapter 4.

### **6.1.1.3 Relation Class**

The relation class represents all of the relationships and the constraints imposed on the functionality operands in order to realize the functionality operation. It includes information about the coupling operands, functional relation, constraints, and degree of freedom.

## **6.1.2 Function Server Object Model**

The class diagram of the function server object model developed in this work is shown in Figure 6.3. The components of this model are: function server, material, manufacturing information, shape, working condition, function, and interface.



**Figure 6.3 Data structure for function server**

Function server class defines function server characteristics and functionality. It acts as a header that provides a reference point for other classes. The attributes of this class are: name, id, parent-id, address, knowledge domain, and type.

The id attribute is a unique identifier for the function server. The name attribute can be a predefined generic name or a user assigned name. The parent-id attribute denotes the unique identifier for the function server parent. This attribute helps to capture and track the parent-child relationship in the function server tree structure. The address attribute is used to specify the exact location of the function server in the tree structure (i.e., function server composition). The



address attribute is expressed as: Address = (i,j), where i is the tree level and j is the position in that level.

The type attribute defines the category or class that the function server belongs to. Function servers are divided into two classes: primitive function servers and composite function servers. Primitive function servers are divided into three sub-classes: functional features, functional standard parts, and functional module parts. Composite function servers are also divided into three sub-classes: compound features, custom parts, and custom sub-assemblies. The knowledge domain attribute gives the manufacturing domain that the function server information or knowledge base belongs to. This can be machining, casting, inject molding, sheet metal, or generic.

Material class describes the material used in the manufacturing of this function server. This includes the type of material and the material properties. Manufacturing information class describes the manufacturing process and the manufacturing properties for the operation to produce the function server. Manufacturing properties include: surface properties, material properties, and alternative materials.

Shape class defines the base shape and the abstract shape of the function server. The base shape can be block, cylinder, or n-polygon. Abstract shape is the schematic representation of the actual conceptual shape of the function server. A link to the abstract shape is preserved in the shape class. Working condition class defines the working environment for the function server. This includes physical phenomena, energy type, load nature, and failure mode. Function class defines the list of functions that this function server can perform. Interface class defines all of the interfaces with other function servers. The attributes of the interface class are: id, name, description, nature, type, spatial relation, and degree of freedom. The function server model

represents the basic structure of the function driven database developed in this research. A more in depth description of function server components and attributes is presented in chapter 5.

### 6.1.3 Conceptual Product Object Model

The class diagram of the conceptual product model developed in this work is shown in Figure 6.4. This object diagram represents the information at the product level. The components of this model are the functional object model and the function server object model. The functional object model represents all the function operations that belong to the product under consideration with their sub-classes and attributes. The function server object model represents all the function servers that constitute the product under consideration with their sub-classes and attributes.

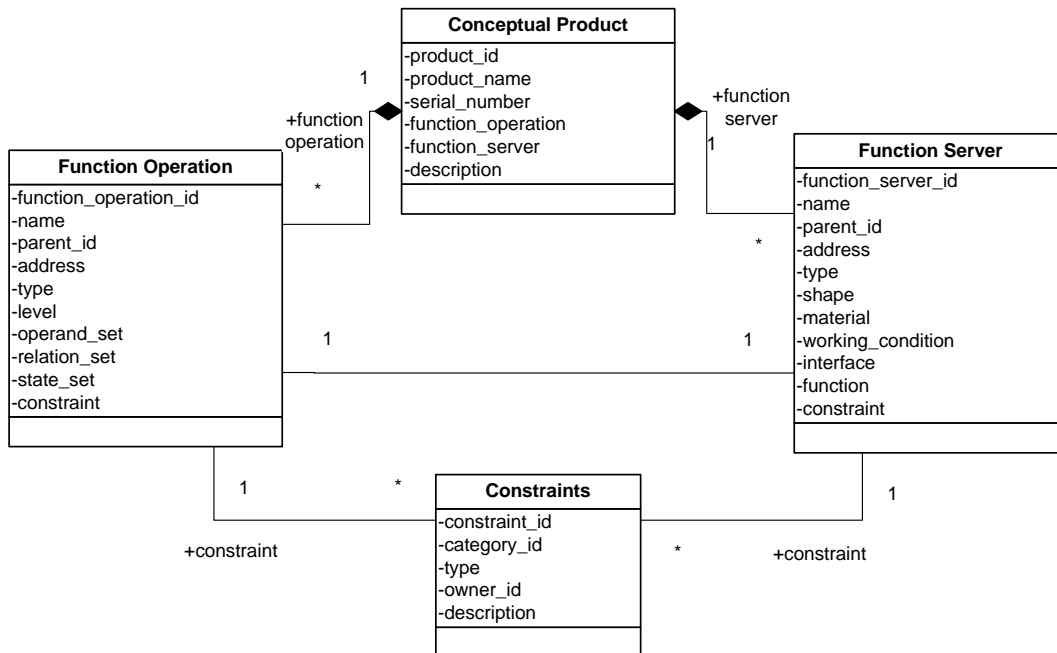


Figure 6.4 Data structure of the conceptual product model

## 6.2 TRANSLATION TOOL XML DATA FORMAT

This research uses Extensible Markup Language (XML) to structure, store, and exchange the translation tool information. XML was designed to carry data. It uses a self-descriptive Document Type Definition (DTD) to describe data as a plain text format. It is a cross-platform software and hardware independent tool for transmitting information. This makes it applicable to represent the translation tool information and means that it can be exchanged between different systems. Tags enclosed in “<” or “>” are used to define the structure and data elements of an XML text or string. In any new implementation, these tags must be defined, because they are not predefined in XML.

### 6.2.1 XML Syntax

XML documents use a self-describing and simple syntax. The first line is the XML declaration, which defines the XML version and the character encoding used in the document. For example, an XML description of a function server type is as follows:

```
<? xml version= "1.0" encoding= "ISO-8859-1"?>
<function-server>
  <info>
    <type>functional feature</type>
    .
    .
  </info>
</function-server>
```

In this example, the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set. The first tag in an XML document is the root tag. This defines the root or the head element of the document. All elements can have sub-elements that should be correctly nested within their parent element. All XML documents must contain a single tag pair to define the root element and must have a closing tag `</>`.

### 6.2.2 Functionality XML Data Format

The XML schema for the functional data model follows:

```
<?xml version= "1.0" ?>
<!-- Functional Data Definition !-->
<functionality-operation>
    <functionality-info></functionality-info>
    <functionality-operand-set></functionality-operand-set>
    <relations-set></relations-set>
    <state></state>
</functionality-operation>
```

The first line in the document (`<? xml version = "1.0"?>`) is the XML declaration. It defines the XML version used in the document. The root tag (`<functionality-operation>`) describes the root element of the document. The definition of an instance of the functional data model starts by this root tag. The end of the instance is defined by the last line, which shows the end of the root element: `</functionality-operation>`. The functionality operation data must be within the opening and closing tags.

The XML schema for functionality information, functionality operand set, relations set, and state are given in the following:

### The info tag

Functionality information XML schema:

```
<functionality-info>
  <function-id></function-id>
  <function-name></function-name>
  <function-type></function-type>
  <function-level></function-level>
  <parent-id></ parent-id >
  <address>
    <tree-level></tree-level>
    <position></position>
  </address>
  </auxi-function-set>
    <auxi-function>
      <id></id>
      <name></name>
    </auxi-function>
    .....
  <auxi-function-set>
  <description></description>
</Functionality-info>
```

### The operand set tag

Functionality operand set XML schema:

```
<functionality-operand-set>
  <operand>
    <operand-id></operand-id>
    <operand-name></operand-name>
    <operand-type></operand-type>
    <operand-role></operand-role>
    <operand-description></operand-description>
    <attrib-set>
      <attrib-name></attrib-name>
      <attrib-value></attrib-value>
    </attrib-set>
```

```

    </operand>
    .....
    <operand>
    </operand>
</functionality-operand-set>

```

Relations set tag

Relations set XML schema:

```

<relations-set>
  <relation>
    <id></id>
    <name></name>
    <description></description>
    <coupling-operands>
      <opand-1>
        <id></id>
        <name></name>
        <coupling-attrib-set>
          <attrib-name></attrib-name>
        </coupling-attrib-set>
      </opand-1>
      <opand-2>
        <id></id>
        <name></name>
        <coupling-attrib-set>
          <attrib-name></attrib-name>
        </coupling-attrib-set>
      </opand-2>
    </coupling-operands>
    <f-rel-set>
      <f-relation>
        <id></id>
        <factor-1></factor-1>
        <rel></rel>
        <factor-2></factor-2>
      </f-relation>
    </f-rel-set>
    <f-constr-set>
      <f-constr>
        <id></id>
        <factor></factor>
        <rel></rel>
        <const></const>
      </f-constr>
    </f-constr-set>
  </relation>
</relations-set>

```

```

        </f-constr>
      </f-constr-set>
    <dof-set>
      <dof>
        <id></id>
        <type></type>
        <ref-frame></ref-frame>
      </dof>
    </dof-set>
  </relation>
</relations-set>

```

### State tag

State XML schema:

```

<state>
  <operand-attrib>
    <operand-id></operand-id>
    <value-set>
      <lower> </lower>
      <upper> </upper>
    </value-set>
  </operand-attrib>
</state>

```

## 6.2.3 Function Server XML Data Format

The XML schema for the function server model is as follows:

```

<?xml version= "1.0" ?>
<!-- Function Server Definition !-->
<function-server>
  <info></info>
  <function></function>
  <shape></shape>
  <material></material>
  <manufac-info></manufac-info>
  <working-info></working-info>
  <interface-set></interface-set>
  <constraint-set></constraint-set>
</function-server>

```

The XML schema for the function server information, function, shape, material, manufacturing information, working conditions, interface, and constraints are given in the following.

#### The info tag

Function server information XML schema:

```
<info>
  <id></id>
  <name></name>
  <type></type>
  <domain></domain>
  <parent-id></parent-id>
  <participants></participants>
  <address>
    <tree-level></tree-level>
    <position></position>
  </address>
  <description></description>
</info>
```

#### Function tag

Function server-function XML schema:

```
<function>
  <primary-function></primary-function>
  <secondary-function1></secondary-function1>
  <secondary-function2></secondary-function2>
  <secondary-function3></secondary-function3>
</function>
```

#### Shape tag

Function server shape XML schema:

```
<shape>
  <base-shape></base-shape>
  <abstract-shape-index></abstract-shape-index>
</shape>
```



### Material tag

Function server material XML schema:

```
<material>
  <type></type>
  <property-set>
    <property-name></property-name>
    <property-type></property-type>
    <property-value></property-value>
  </property-set>
</material>
```

### Manufacturing info tag

Manufacturing information XML schema

```
<manufac-info>
  <manufac-process></manufac-process>
  <property-set>
    <property-name></property-name>
    <property-type></property-type>
    <property-value></property-value>
  </property-set>
  <alternate-material></alternate-material>
</manufac-info>
```

### Working condition tag

Working condition XML schema:

```
<working-info>
  <physical-phenom></physical-phenom>
  <energy-type></energy-type>
  <load-nature></load-nature>
  <failure-mode></failure-mode>
</working-info>
```

### Interface tag

Interface XML schema:

```
<Interface-set>
  <interface>
    <id></id>
    <name></name>
    <description></description>
    <interface-with>
      <object-id></object-id>
      <object-name></object-name>
    </interface-with>
    <interface-type></interface-type>
    <spatial-rel></spatial-rel>
    <dof>
      <type></type>
      <ref-frame></ref-frame>
    </dof>
  </interface>
</Interface-set>
```

### Constraint tag

Constraint XML schema:

```
<constraint-set>
  <constraint>
    <id></id>
    <type></type>
    <factor></factor>
    <rel></rel>
    <constant></constant>
  </constraint>
</constraint-set>
```

### 6.3 ARCHITECTURE OF THE FUNCTION TO CONCEPTUAL FORM TRANSLATION TOOL

The flow diagram for translating function specifications into conceptual form in a function driven design system is given in Figure 6.4. The inputs to the system include the design task and needs, given in the form of problem statements, preferences, and constraints. All the tasks and needs should be fully described at this stage, because anything neglected here cannot be captured later on in the functionality modeling stage. The design problem can be about either a new product (i.e., an original design) or about a task within a specific product (i.e., a variant design).

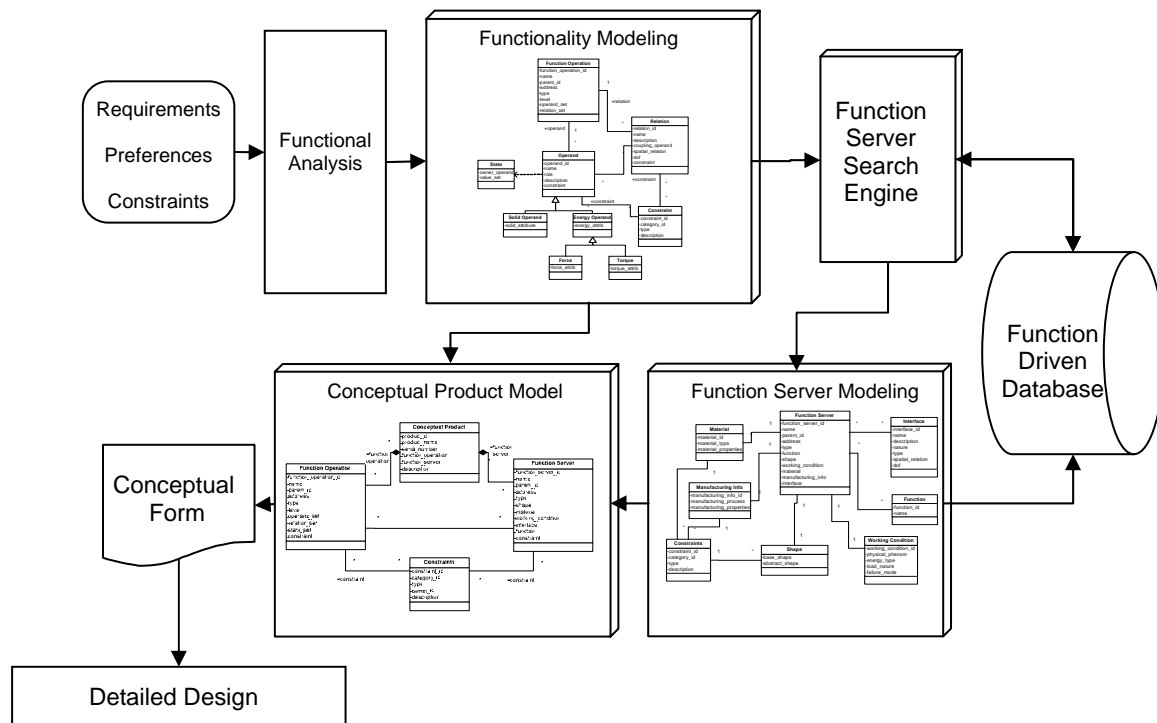


Figure 6.5 The translation tool architecture

Functional analysis includes identifying the functional requirements that the design product must have in order to perform successfully. The design tasks and needs are mapped onto functional requirements that include the function decomposition, the required functionality operands and their corresponding attributes, the relations and interactions between operands, and the functional constraints.

At the functionality modeling stage, the functional requirements are translated into object instances in the functionality model. This includes, for each function operation, the modeling operands and their attributes, the relations between operands, and the corresponding attributes and constraints. The functional data model captures and maintains the functional information of the design problem in the computer data structure (object model and XML schema). At the same time, it is used to search for potential solutions. The function information available in the functionality model is passed to the function server search engine. This engine searches the function driven database for matches and presents the function servers that may potentially satisfy the required function. The solution is then saved with all the related information, including the function information, in a new data structure, known as the function server object model. If the designer is not satisfied with the solution, or if there is no solution, the system will ask him or her to consider possible modifications to the functional requirements. The key point to note is that the solutions are accessed through their functionality rather than by their name. The advantage to this is that the designer does not need to know the solution before searching for a function server to satisfy the problem.

The function server object model captures information about the function servers that have already been chosen as potential solutions for the functional requirements. This information includes: shape, material manufacturing information, interface, working conditions, and

constraints. The function server model represents the realization of the conceptual form. This knowledge can be passed back to the function driven database and saved for reuse in the future.

The function driven database possesses a comprehensive information structure for maintaining function server data. Initially, only knowledge of primitive function servers, functional features, functional standard parts, and functional standard modules are available for the designer to use. However, when the designer starts to use the translation tool, the solutions he or she finds can be saved in the function driven database for reuse in later projects. In this dissertation, a prototype function driven database is built to demonstrate and validate the working of the translation tool. This database contains information on functional features only. However, the database could easily be extended to include other function server types.

Function data and function server information are then stored and organized by a data structure: the conceptual product object model. This model will provide the facilities for managing the huge amount of design information and for allowing the designer to access design information from different views, such as function decomposition, product tree, and constraints. The conceptual product model will be passed to the detailed design stage to complete the design process by providing a detailed description of function servers that can satisfy the predefined functions.

#### **6.4 WEB-BASED IMPLEMENTATION AND THE GRAPHIC USER INTERFACE**

The function-to-conceptual form translation tool is implemented in a client-server architecture with a web-based user-friendly graphical interface. Other services are implemented as dynamic web pages. In the dynamic web pages, the published contents are created “on the fly”

after a specific request. This system requires two layers. The first is the database layer, which will manage the search for solutions, and the second is a scripting layer, which allows the database and the web server to communicate. MySQL is used for the first layer and PHP for the second. Apache software is employed as the web server. This popular open software solution is very flexible, safe, and provides a lot of excellent online documentation.

The database of the translation tool is realized using MySQL, which is a relational database management system that is particularly suited for web-based applications on high-traffic sites that generate content on the fly. MySQL is widely regarded as faster than other databases. It is distributed in a relatively small package, not in hundreds of megabytes like some other database products. Moreover, MySQL understands Structured Query Language (SQL), the most common database language in use today. <sup>[79]</sup>

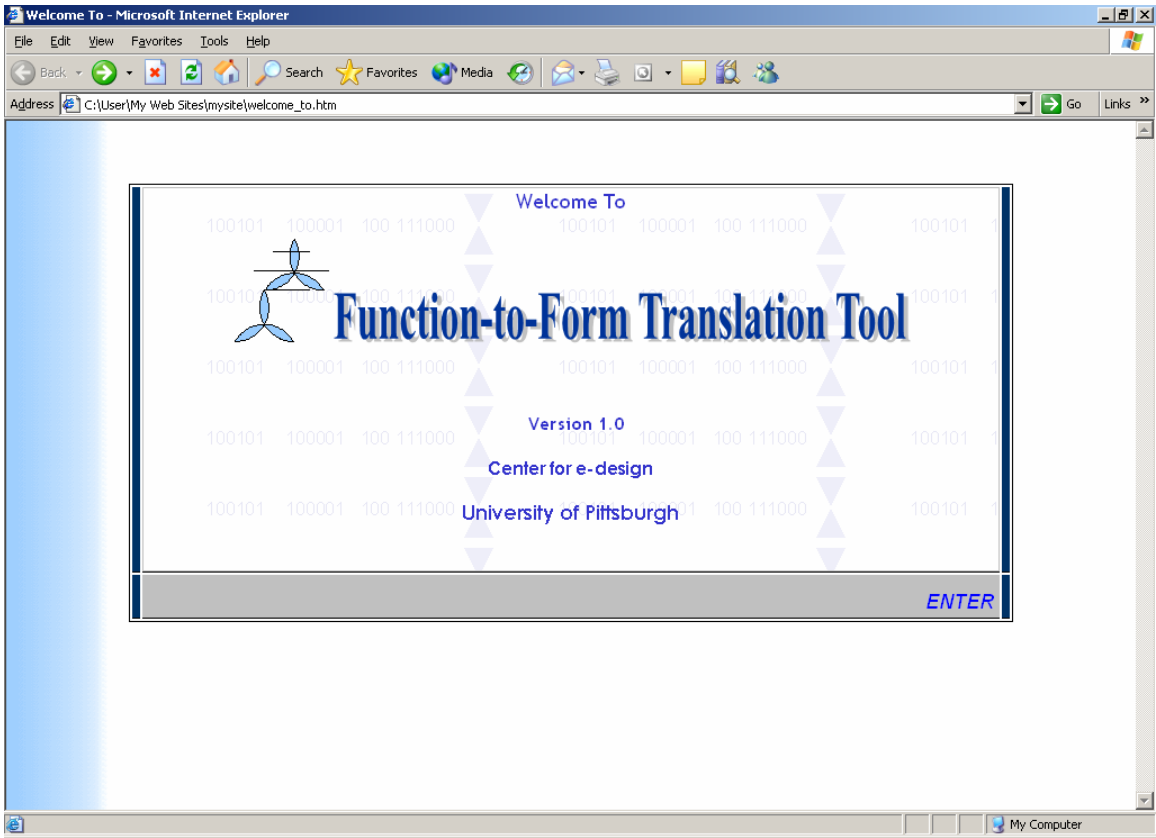
PHP, Hypertext Preprocessor, is a popular open-source programming language used primarily for developing server-side applications and dynamic web content. There are several advantages in using PHP. First, PHP has cross-platform operability and is suitable for today's heterogeneous network environments. As a result, PHP runs on almost any computer platform with only minor modifications to the code. Second, PHP is very fast and has a small memory footprint. Third, it is simple, easy-to-learn, and its intuitive interface allows programmers to embed PHP commands right into the HTML page. Finally, PHP offers many advanced features for the professional programmer <sup>[80]</sup>.

#### **6.4.1 Screen Presentation and General Usage**

A windows-type graphical user interface, designed using Microsoft Front Page, provides the designer with several menu and button options. Information in the database can be easily

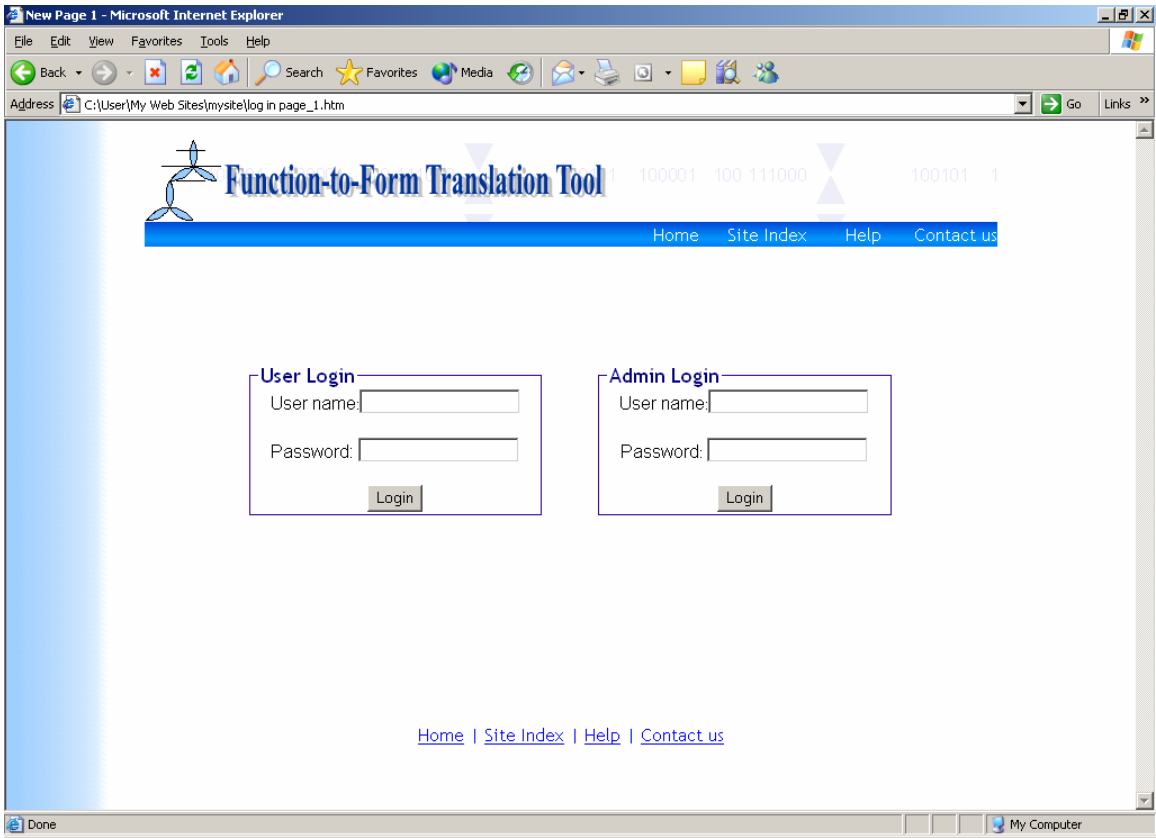
accessed, viewed, modified, added, or deleted. An overview of the run-time screens and general usage is presented in the following.

When a designer connects to the web server, he or she is first presented with the Beginning Screen, shown in Figure 6.6. Clicking on the Enter button brings up the Login screen, shown in Figure 6.7. Administrative personnel can login separately from regular users. Logging in as a regular user takes the designer to the Main Screen, as shown in Figure 6.8. This screen presents three options to the designer. The first option is to start a new project or to check/modify the available projects. This option can be selected by clicking on “Products” in the left frame. The second option is to view/modify the function listing that is currently contained within the database. This option can be selected by clicking on “Functions List” in the left frame. Figure 6.9 shows the screen after choosing this option. Appendix B shows the complete function listing and the corresponding synonyms. The third option is to view/modify the function server listing that constitutes the function driven database. A view after choosing this option is presented in Figure 6.10. Appendix C shows the function servers used, in this dissertation, as a knowledge base with their possible functions. This knowledge base was built based on data collected from surveys, literature, and handbooks.

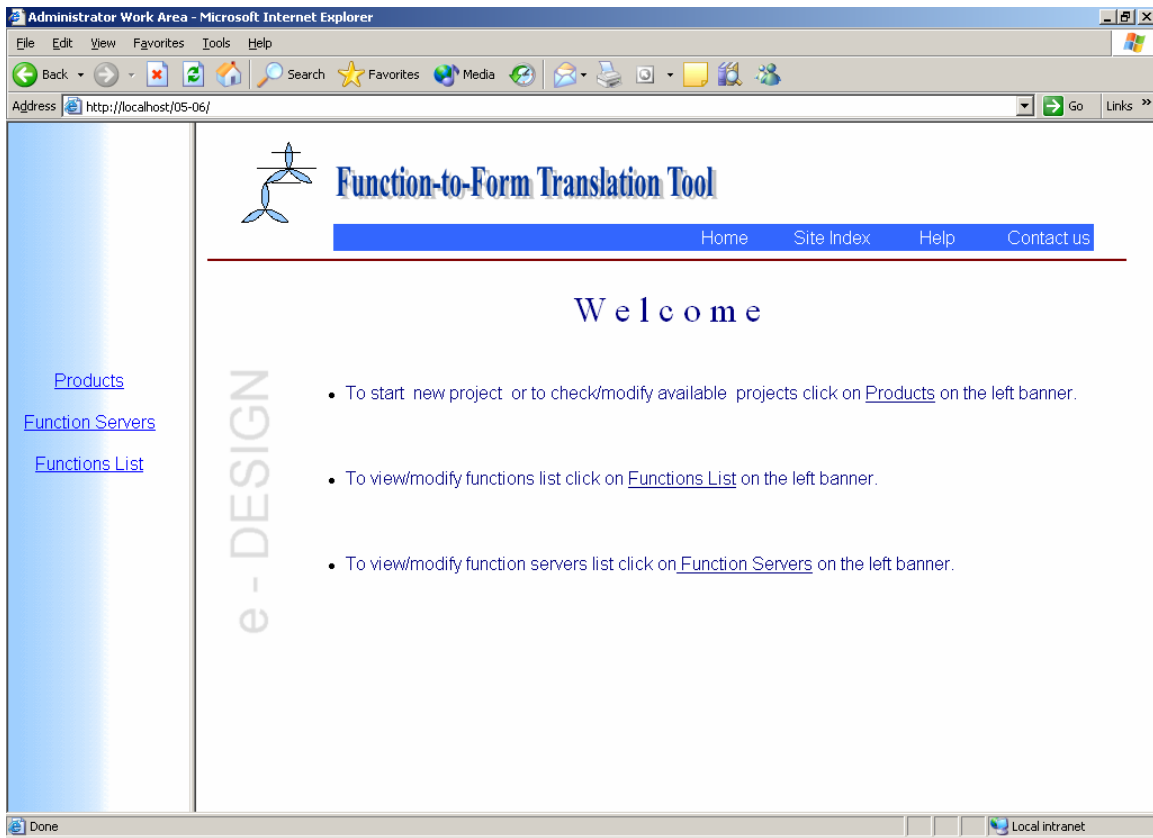


**Figure 6.6 Beginning Screen**





**Figure 6.7 Login Screen**



**Figure 6.8 Main Screen**

Administrator Work Area - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://localhost/05-06/ Go Links

### List of Functions

ID	Function Name	Synonyms	Delete?
4	<a href="#">support</a>	<a href="#">load bear, base, carry, hold, sustain, offset</a>	<a href="#">Yes</a>
6	<a href="#">access</a>	<a href="#">pass, enter</a>	<a href="#">Yes</a>
7	<a href="#">align</a>	<a href="#">straighten, adjust, line up, follow</a>	<a href="#">Yes</a>
8	<a href="#">enclose</a>	<a href="#">cover, contain, shield, protect, surround, include, guard, hide, wrap, insulate</a>	<a href="#">Yes</a>
10	<a href="#">allow</a>	<a href="#">permit, let, approve, grant</a>	<a href="#">Yes</a>
11	<a href="#">assemble</a>	<a href="#">fabricate, combine, bring together, gather, unite</a>	<a href="#">Yes</a>
12	<a href="#">assist</a>	<a href="#">Help, aid, work for, work with</a>	<a href="#">Yes</a>
13	<a href="#">balance</a>	<a href="#">Stabilize, steady, equal, adjust, level</a>	<a href="#">Yes</a>
14	<a href="#">block</a>	<a href="#">Obstruct, bar, stop</a>	<a href="#">Yes</a>
15	<a href="#">connect</a>	<a href="#">Attach, join, couple, link, fasten, mate, engage, mesh</a>	<a href="#">Yes</a>
16	<a href="#">control</a>	<a href="#">Regulate, limit, constrain, adjust, modify, restrict</a>	<a href="#">Yes</a>
17	<a href="#">eject</a>	<a href="#">Dispose, remove, discharge, export, bump</a>	<a href="#">Yes</a>
18	<a href="#">fit</a>	<a href="#">Agree, conform, match, meet, go together</a>	<a href="#">Yes</a>
19	<a href="#">friction</a>	<a href="#">Rub, resist</a>	<a href="#">Yes</a>
20	<a href="#">guide</a>	<a href="#">Direct, straighten, steer, control, regulate, lead</a>	<a href="#">Yes</a>
21	<a href="#">hold</a>	<a href="#">Stop, lock, remain, stay, secure</a>	<a href="#">Yes</a>
22	<a href="#">mount</a>	<a href="#">Secure, lock, fasten, hold, attach, fix</a>	<a href="#">Yes</a>
23	<a href="#">position</a>	<a href="#">Orient, locate, place, put, set, settle, stand, occupy</a>	<a href="#">Yes</a>
24	<a href="#">prevent</a>	<a href="#">Avoid, hinder, prohibit, restrict, inhibit</a>	<a href="#">Yes</a>
25	<a href="#">protect</a>	<a href="#">Guard, cover, care for, keep, insulate, keep safe</a>	<a href="#">Yes</a>
26	<a href="#">reduce</a>	<a href="#">Decrease, step down</a>	<a href="#">Yes</a>
27	<a href="#">reflect</a>	<a href="#">Bend back, mirror, reverse, throw back, return</a>	<a href="#">Yes</a>

Products  
Function Servers  
Functions List

Local intranet

Figure 6.9 Function list

The screenshot shows a web browser window titled 'Administrator Work Area - Microsoft Internet Explorer'. The address bar shows 'http://localhost/05-06/'. The main content area displays a table titled 'Function Servers'. The table has the following columns: ID, Function Server Name, Type ID, Subtype ID, Knowledge Domain, Participants, Description, Abstract Shape, and Delete?. The rows contain data for function servers with names like 'wall', 'boss', 'chamfer/fillet', 'rib', and 'flange'. Each row includes a 3D abstract shape icon and a 'Delete?' link with the value 'Yes'.

ID	Function Server Name	Type ID	Subtype ID	Knowledge Domain	Participants	Description	Abstract Shape	Delete?
54	<a href="#">wall</a>	<a href="#">0</a>	<a href="#">0</a>	<a href="#">generic</a>	-	-		<a href="#">Yes</a>
55	<a href="#">boss</a>	<a href="#">0</a>	<a href="#">0</a>	<a href="#">generic</a>	-	-		<a href="#">Yes</a>
66	<a href="#">chamfer/fillet</a>	<a href="#">0</a>	<a href="#">0</a>	<a href="#">generic</a>	-	-		<a href="#">Yes</a>
56	<a href="#">rib</a>	<a href="#">0</a>	<a href="#">0</a>	<a href="#">generic</a>	-	-		<a href="#">Yes</a>
57	<a href="#">flange</a>	<a href="#">0</a>	<a href="#">0</a>	<a href="#">generic</a>	-	-		<a href="#">Yes</a>

**Figure 6.10 Function driven database**

If the designer chooses to start a new project (or product) or to check/modify the available projects, a new screen will be presented, as shown in Figure 6.11. At this web page, the designer can add a new product by specifying its name, serial number, application, and description. In addition, he or she can update the information of old products. After adding/updating the information about a product, the designer can then proceed to build the function structure for this product. By clicking on the product in the list, the screen, shown in Figure 6.12, will be presented to the designer. At this screen, the designer can choose between

adding/modifying product functions or adding/modifying product function servers. By clicking on Add/Modify functions, the designer is taken to the screen shown in Figure 6.13. On this page, the function decomposition information for the product can be added.

To build the functionality model for each function in the list, the designer can choose any function in the function decomposition list. Clicking on it brings up a new screen, as shown in Figure 6.14. The Add/Modify operands button opens the screen shown in Figure 6.15. In this screen, the designer can add operands and attributes (see Figure 6.16). The Add/Modify relations button opens a new screen as shown in Figure 6.17. Here the designer can add relations between operands. The Add/Modify constraints button opens the screen shown in Figure 6.18 in which the designer can add the functionality constraints.

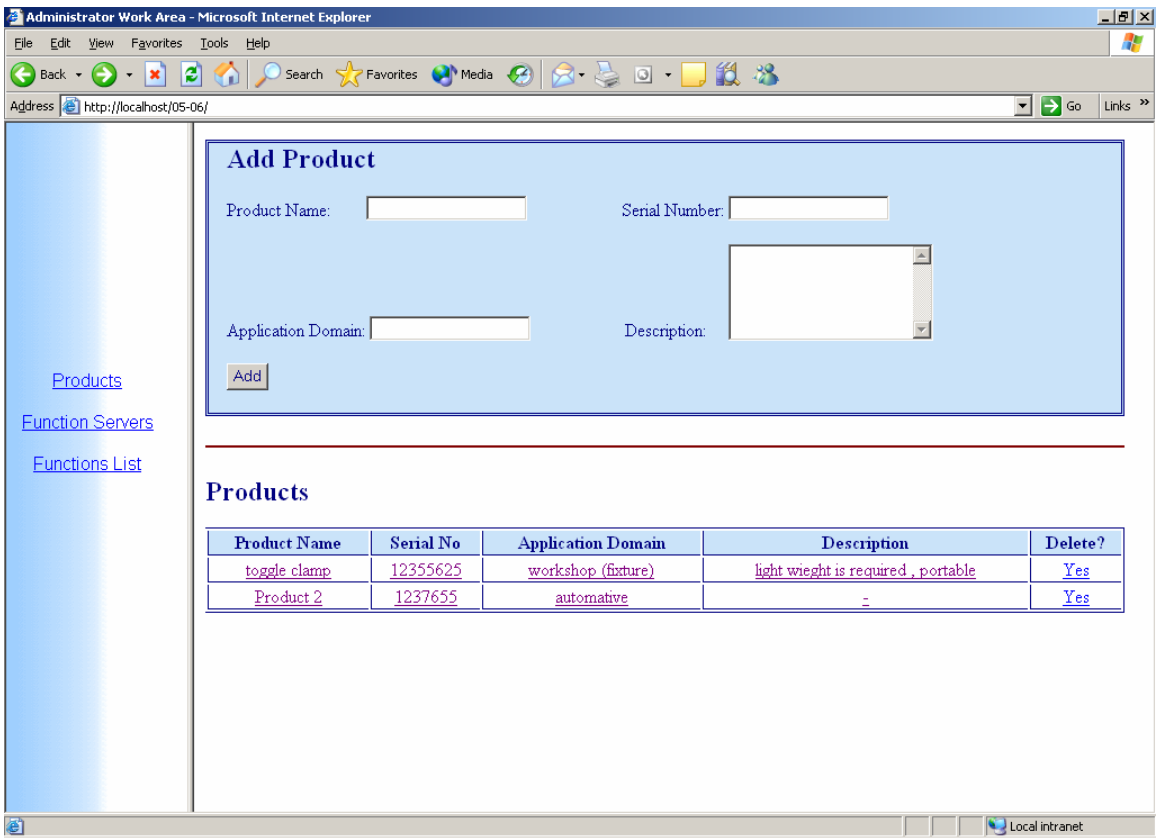
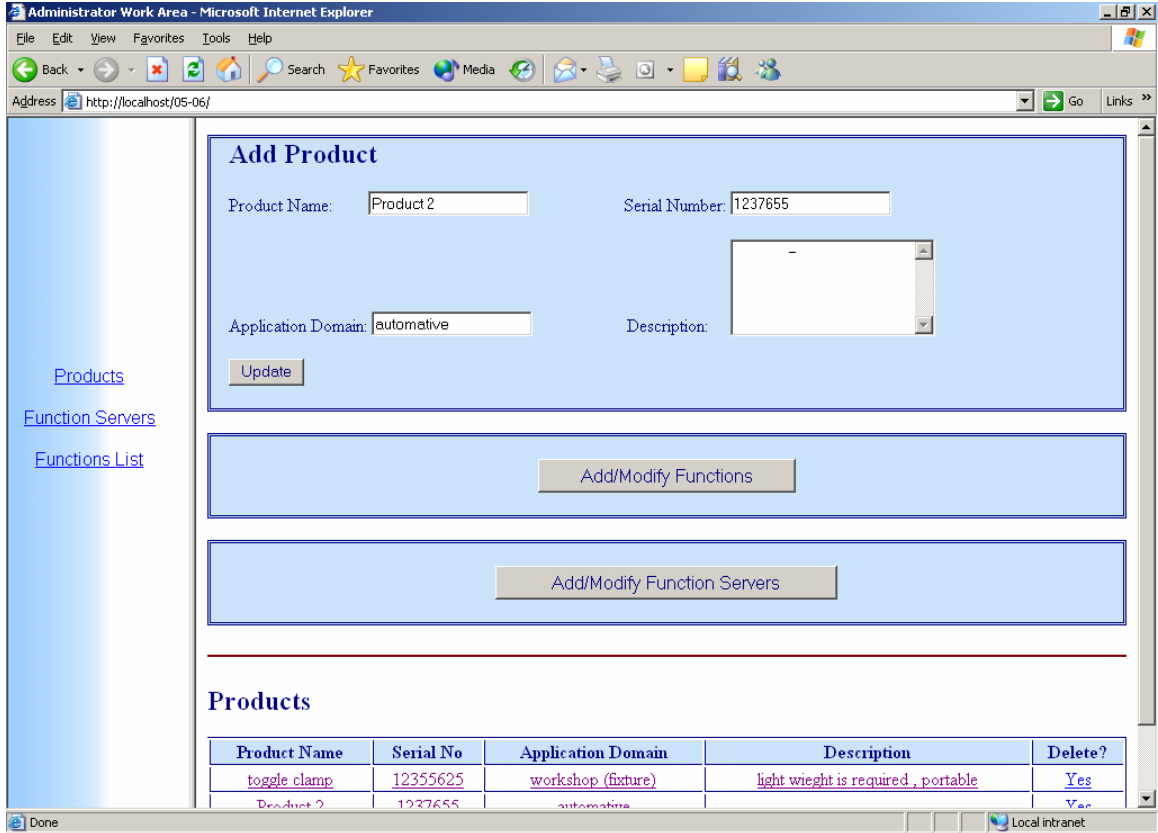


Figure 6.11 Add/Modify products



**Figure 6.12 Choosing add/modify product functions or add/modify product function servers**

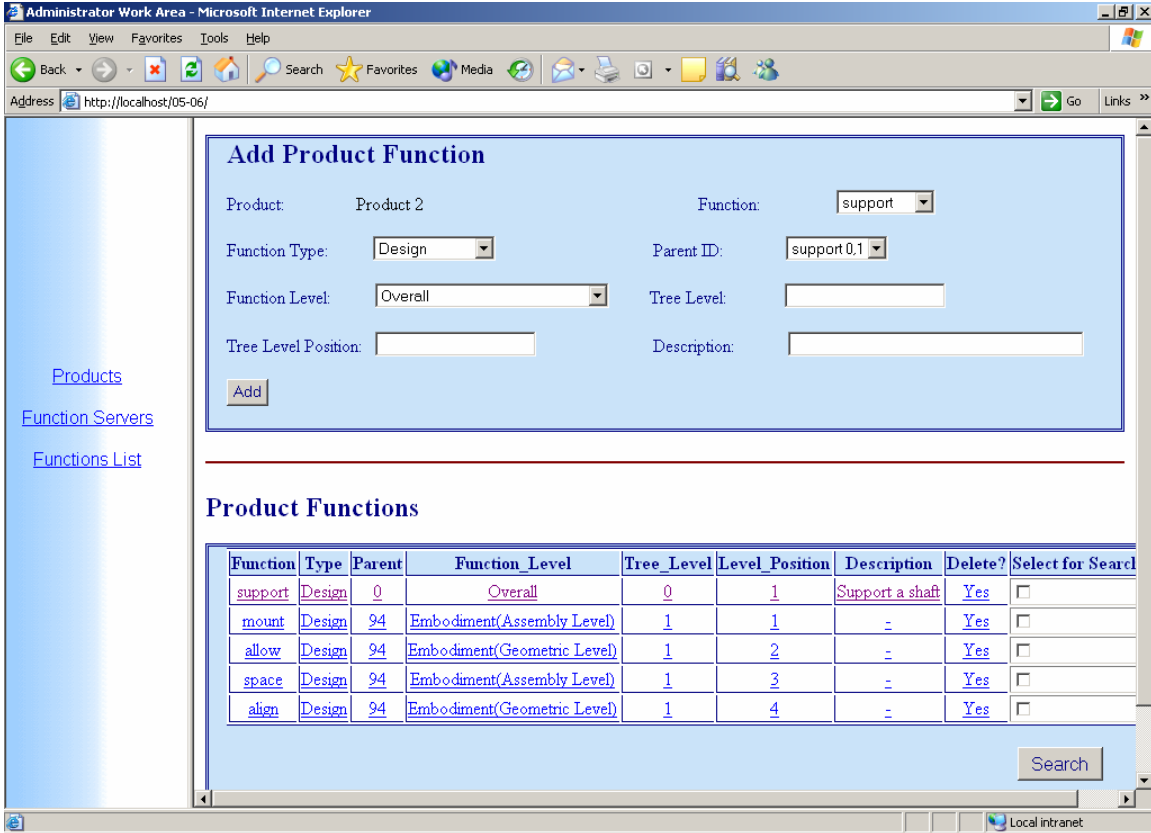
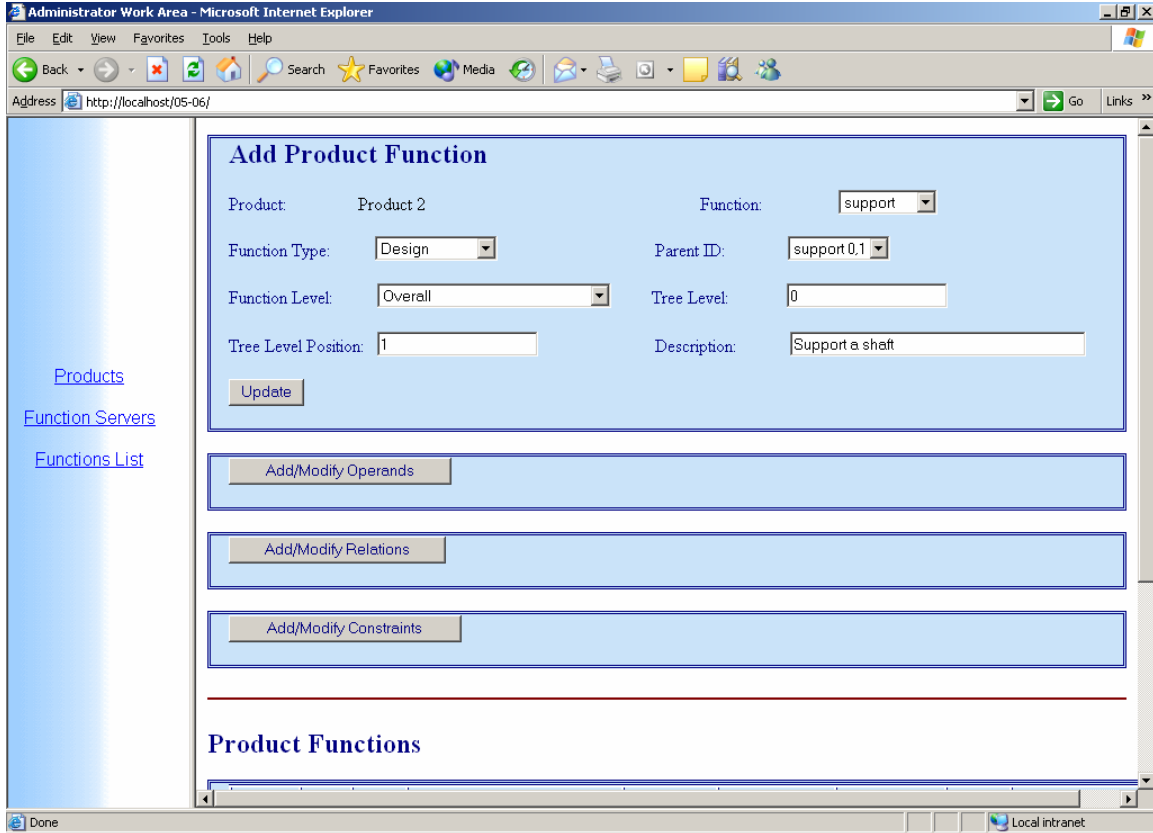


Figure 6.13 Add/Modify product functions (function decomposition)





**Figure 6.14 Add/Modify functionality model components (operands, relations, constraints)**

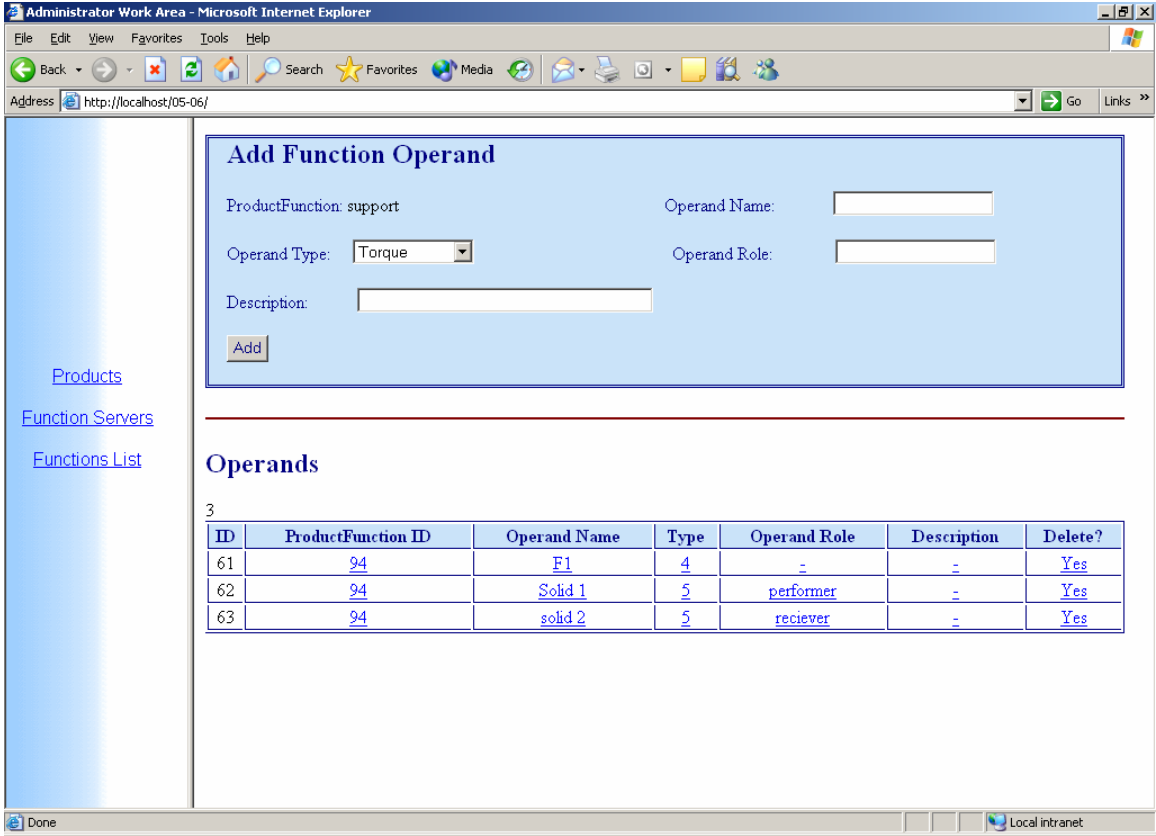


Figure 6.15 Add function operands

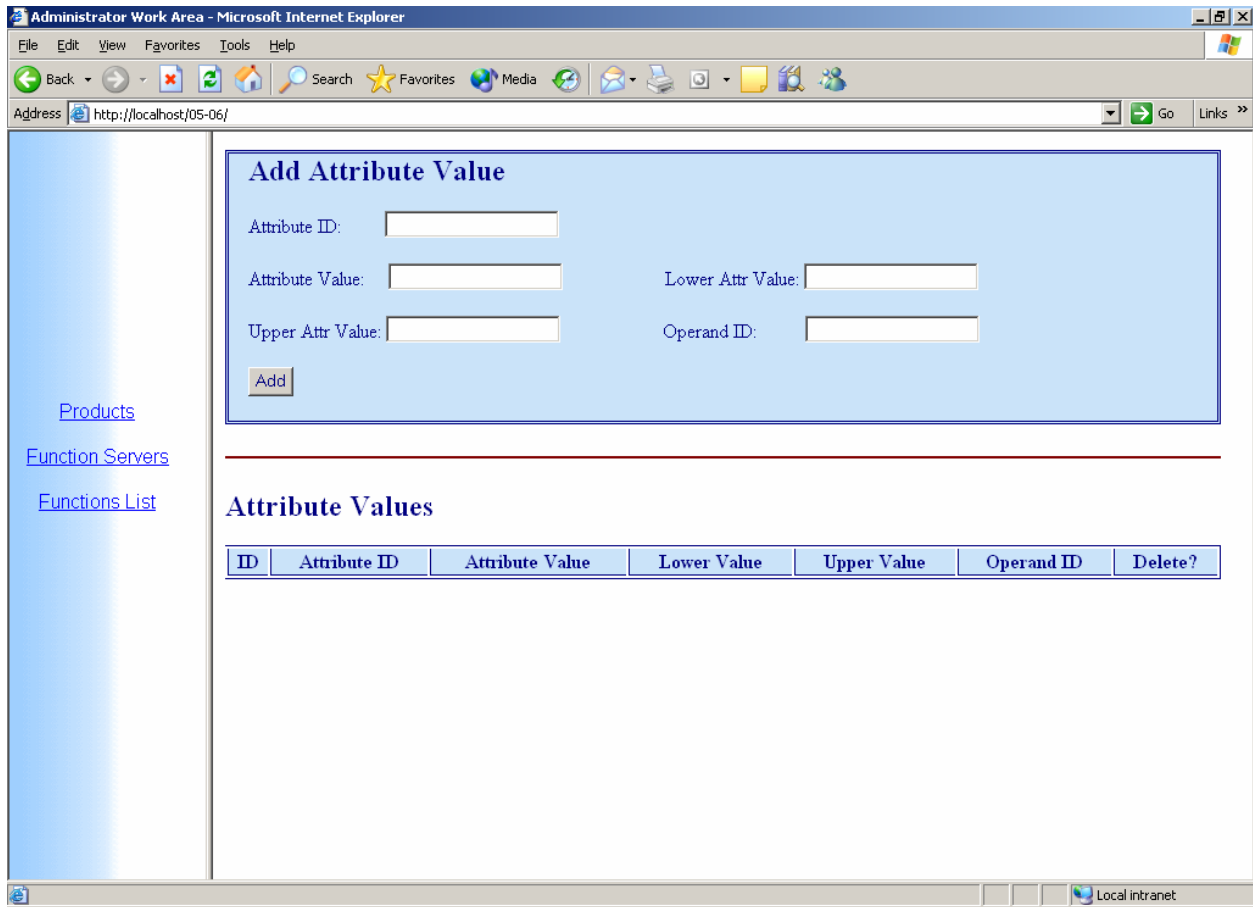


Figure 6.16 Operands attributes

Administrator Work Area - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media

Address http://localhost/05-06/ Go Links

### Add Function Relation

Relation Name:

FirstProductFunction ID:  First Operand ID:

SecondProductFunction ID:  Second Operand ID:

First DoF ID:  Second DoF ID:

Spatial Relation ID:  Description:

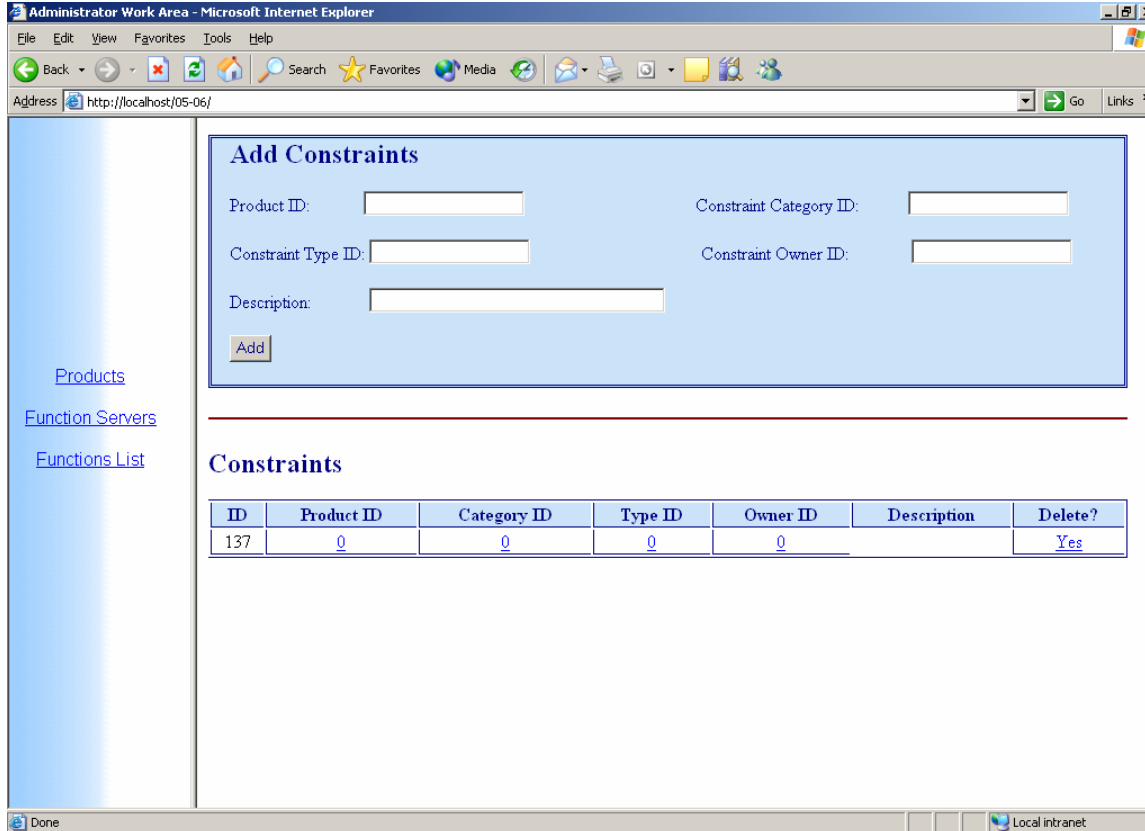
---

### Relations

ID	Relation Name	FirstProductFunction ID	First Operand ID	SecondProductFunction ID	Second Operand ID	First DoF ID	Second DoF ID	Spatial Relation ID	Description	Delete?
1	<a href="#">contact</a>	<a href="#">12</a>	<a href="#">3</a>	<a href="#">22</a>	<a href="#">34</a>	<a href="#">11</a>	<a href="#">13</a>	<a href="#">2</a>	non	<a href="#">Yes</a>

Done Local intranet

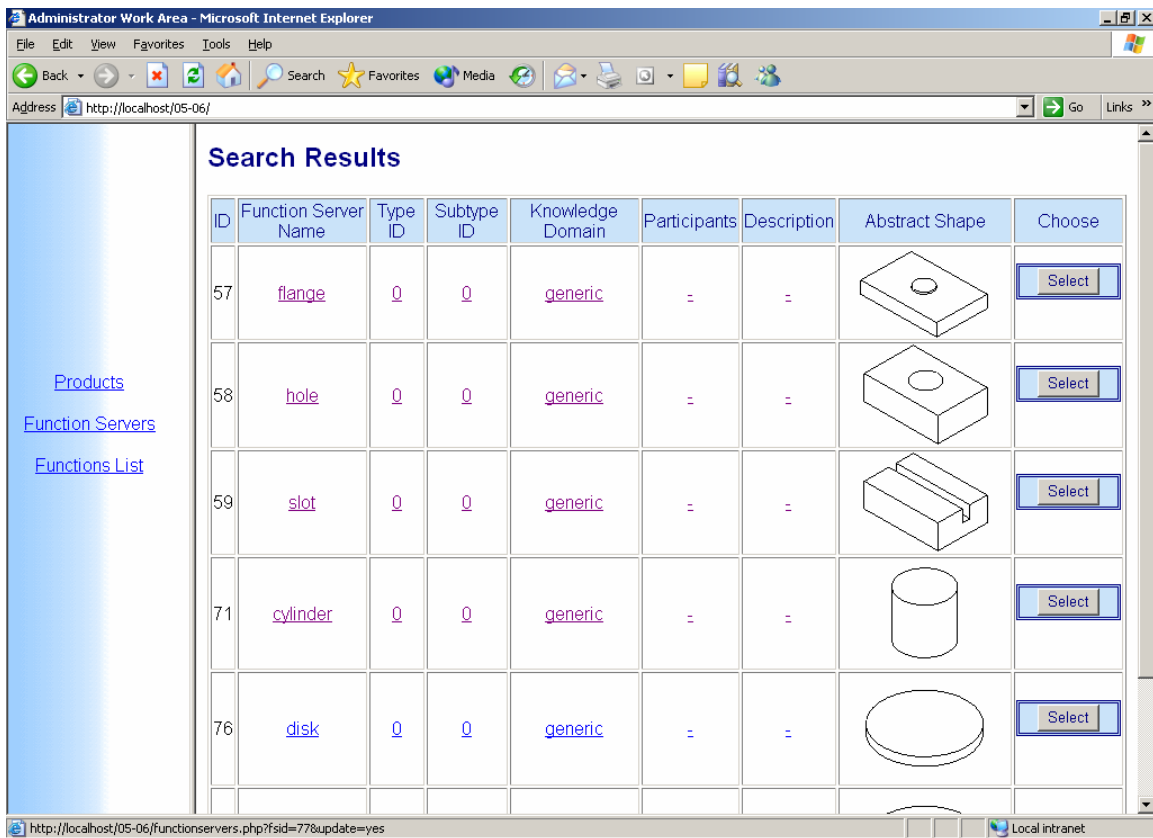
**Figure 6.17 Add function relations**



**Figure 6.18 Add function constraints**

Returning to the function decomposition screen (see Figure 6.12), the checkbox to the right of each function allows the designer to use this function to search the database. Multiple functions also can be used to search the database for a solution to satisfy these functions. For example, searching for solutions for the functions support and align gives the list of results seen in Figure 6.19. The designer can then choose any one of these solutions by clicking the select button. This will save the selected function server in the product function server list as shown in Figure 6.20.

Building the function server model for any product function server in the list (see Figure 6.20) can be done by clicking on that physical element. Information about product function server materials, manufacturing information, shape, working conditions, and interface can be added simply by clicking (see Figure 6.21). For example, if the material button is chosen, the information about material type can be added, as shown in Figure 6.22.



**Figure 6.19 Search results**

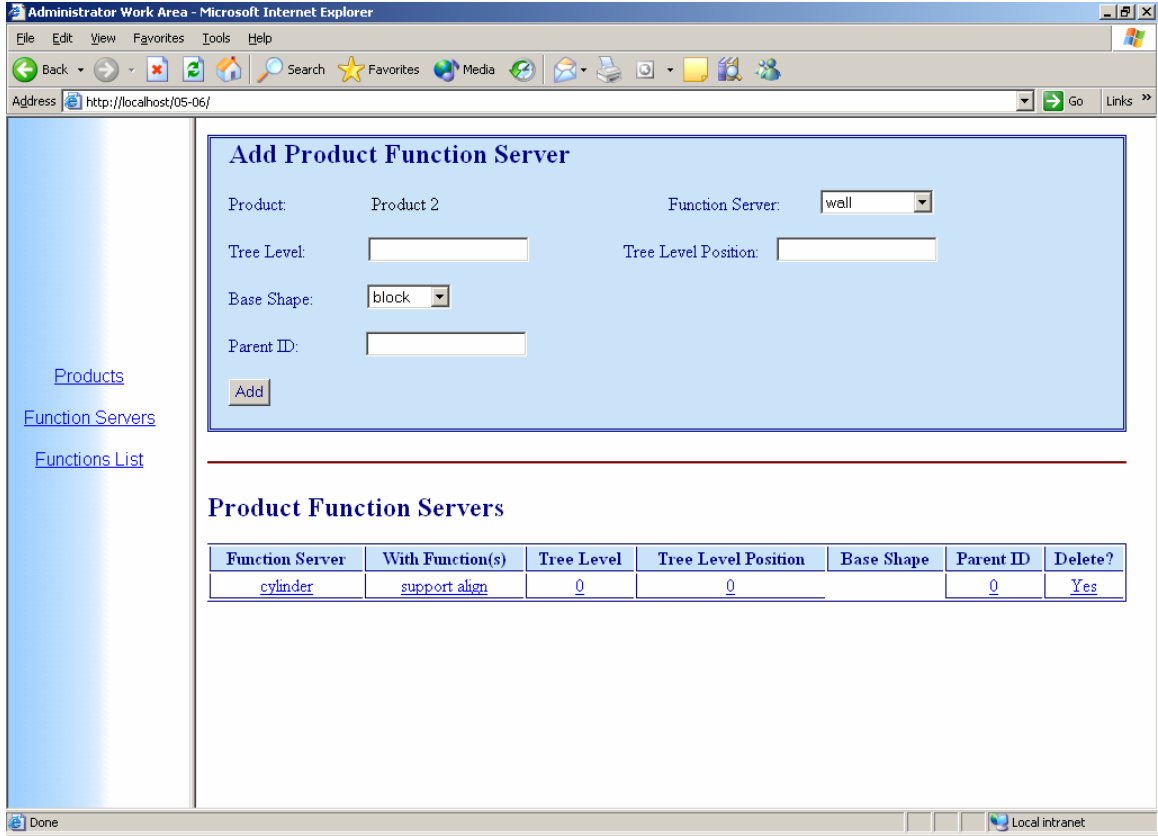
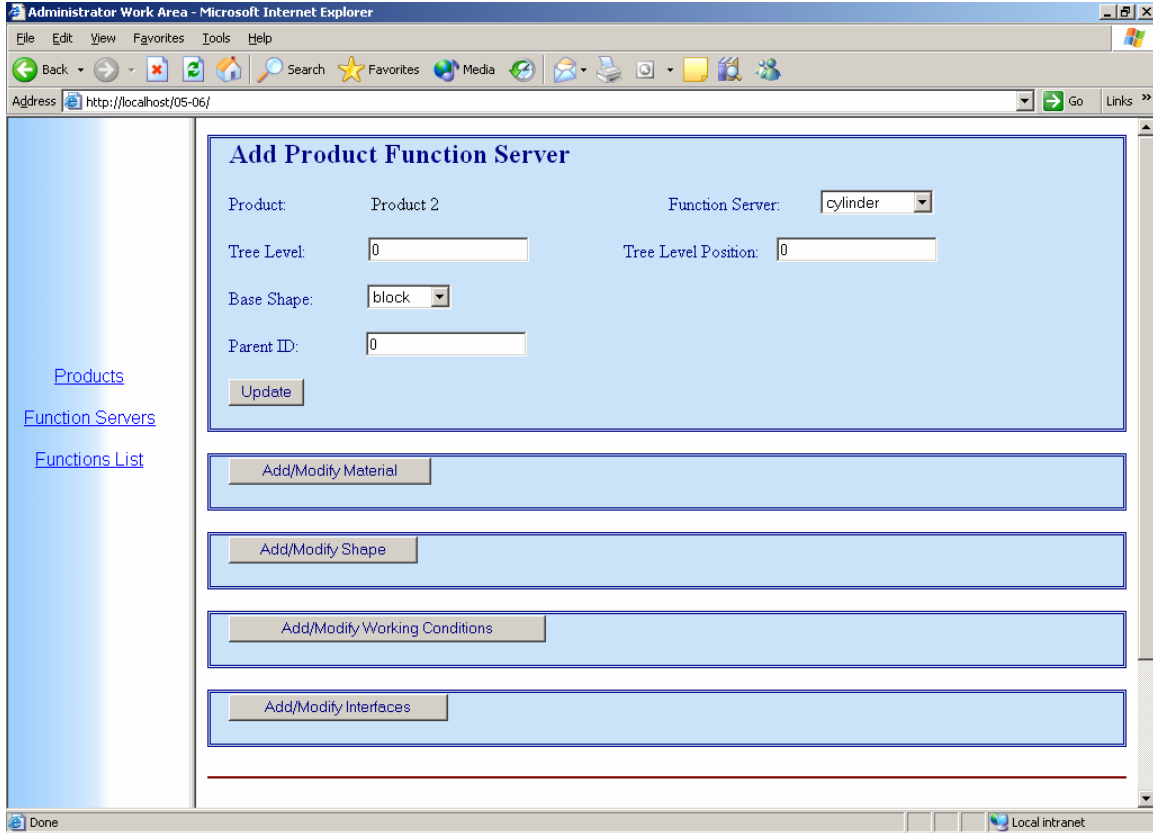
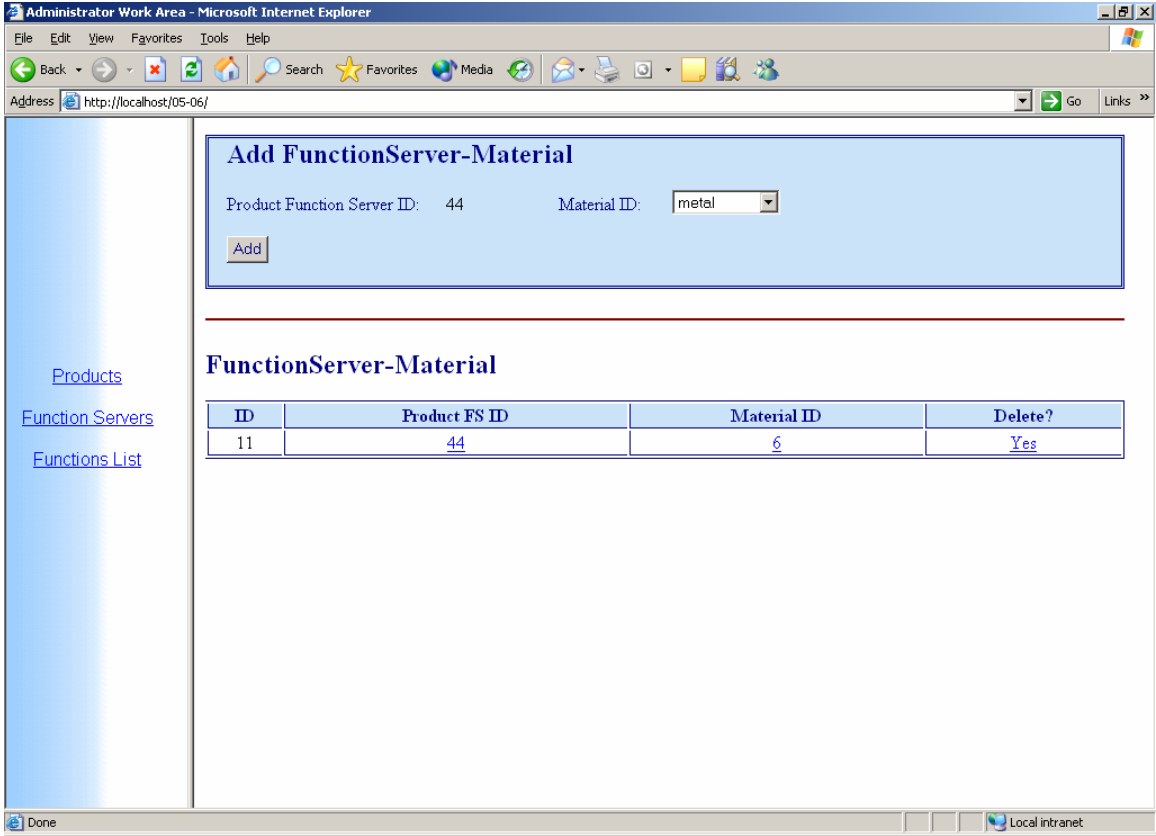


Figure 6.20 Product function servers (solutions)



**Figure 6.21 Components of product function server model (material, shape, manufacturing info, working condition, and interface)**





**Figure 6.22 Material type for product function server**

## 6.5 CASE STUDY AND VALIDATION

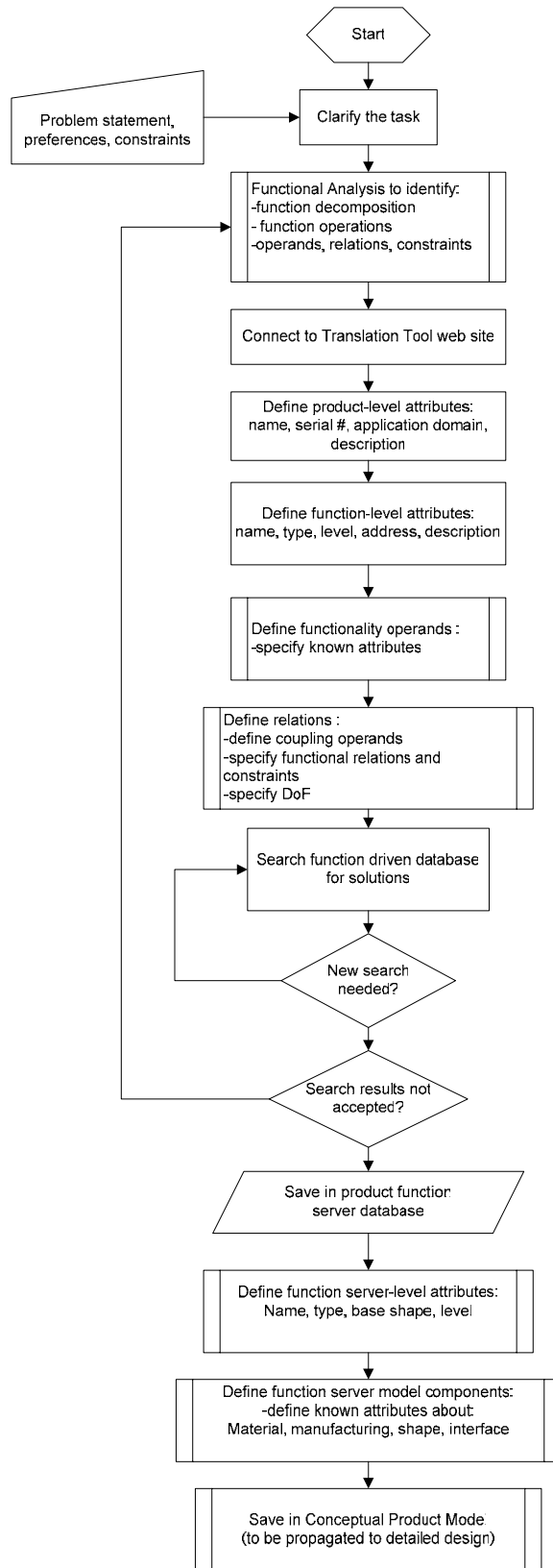
The translation tool developed in this work was tested and validated using the case study of a toggle clamp. By evaluating how efficiently the models capture and translate the functional requirements of the product into conceptual forms, the models developed in this work can be validated. The procedure of translating function-to-conceptual form is presented first. The toggle clamp example is then used to demonstrate and validate the methodology.

### 6.5.1 Procedure of Translating Function-to-Conceptual Form

The following steps illustrate the specific design actions that are involved when translating function-to-conceptual form. The process flow associated with these steps is given in Figure 6.23.

1. Clarify the needs and tasks by defining the design problem.
2. Analyze the functional requirements to identify:
  - Functional decomposition structure.
  - Individual functional operations.
  - Functionality operands and their corresponding attributes for each function operation.
  - Functionality relations and constraints for each function operation.
3. Connect to the translation tool website.
4. Define a new product with the corresponding attributes: name, serial number, application domain, and description.
5. Define function level attributes: name, type, level, and address.

6. Define new operands:
  - Specify known attributes.
  - Specify attribute value set if known.
7. Define relations between operands:
  - Select coupling operand pair.
  - Define functional relations and constraints.
  - Define DoF.
8. Search function driven database for solutions:
  - Specify functions for search.
9. If more solutions are needed, search again:
  - Repeat STEP 8.
10. Else, if the search result is not accepted:
  - Go to STEP 2.
11. Else, save the solution (function server) in the product function server model.
12. Define function server-level attributes: name, type, base shape, and level.
13. Define attributes of function server components:
  - Define material type and attributes.
  - Define manufacturing process and attributes.
  - Define working conditions.
  - Define interface and attributes.
14. Save function server information and functionality information in conceptual product model.



**Figure 6.23 Translating function-to- conceptual form flow diagram**

### 6.5.2 Case Study: Toggle Clamp

The case study used to evaluate the models developed in this work was a toggle clamp. A toggle clamp is a device used to hold or secure an object against a fixed surface to prevent it from moving. In this device, hand force is magnified and directed in order to plunge a spindle into an object so as to firmly hold it. The length and cross section of the spindle (plunger) depends on the holding capacity required.

#### STEP I:

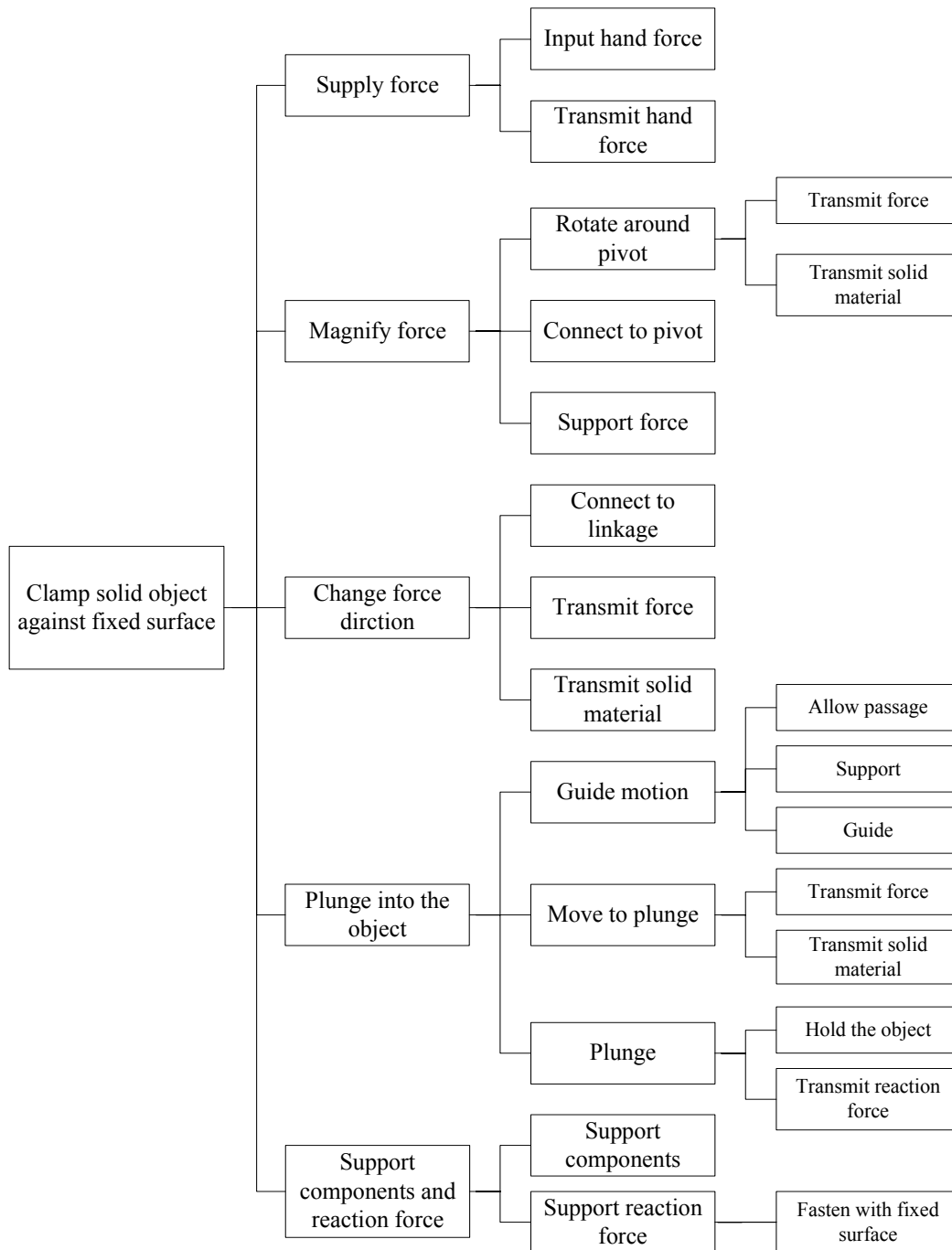
The first step in the design process is to identify the product's needs and to develop the functional specifications and constraints. The objective function of the device is: clamp solid object against fixed surface. The other requirements and constraints might include:

- Not heavy (total weight less than 5 Kg).
- Can be used in different places (portable).
- Can magnify hand force.
- Holding capacity = 1600 Kg.

#### STEP II:

The first step in functional analysis is to decompose the objective function into sub-functions. This decomposition is based on the possible interactions between energy types and materials. In this example, the objective function (clamp solid object against fixed surface) is broken down into five sub-functions: supply force, magnify force, change force direction, plunge into the object, and support components and reaction force. The sub-functions list fully describes the tasks required to achieve the overall objective function. However, they are still very general and require another level of decomposition. The final form of function decomposition is shown in Figure 6.24. The sub-functions at the lower level are embodiment functions, which means that

they can be mapped to physical elements. At the same time, it is not necessary for each individual sub-function in the lower level to have its own physical embodiment. Several sub-functions can be mapped onto one or more physical elements.



**Figure 6.24 Function Decomposition for toggle clamp**

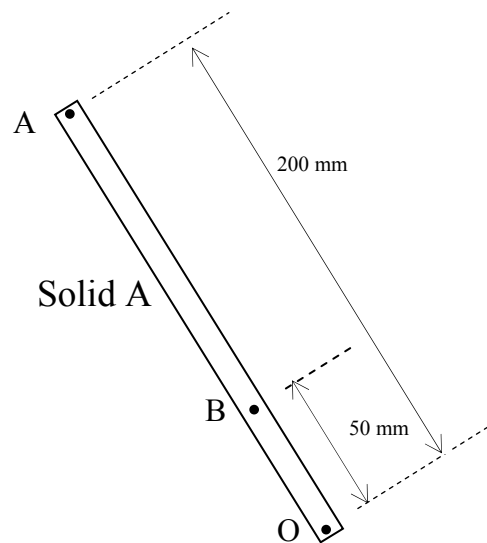
The next step in the functionality analysis is to identify the operands and relations. Given the nature of the task that needs to be performed, the function decomposition, and the context of the operation, the identifiable operands in this example are solid material and force energy. The solid operand consists of the actual physical components that will make up the product. The force is the set of external forces. To define the full set of operands and relations for this product, function operations must be investigated individually. A function operation can represent one or more sub-functions in the decomposition structure. To define a function operation, the designer must examine the function decomposition structure and try to figure out the interactions between material and energy. In our example, the sub-functions supply force and magnify force are considered first. In supply force function, a solid material (solid A) is required to receive hand force (input) and to transmit this force to another place. In magnify force function, the transmitted hand force is magnified by rotating the same solid material (solid A) around a pivot. Therefore, the combined function supply and magnify force can be considered as a function operation (function operation 1), in which a solid material operand (solid A) interacts with several force operands. The lower level sub-functions that constitute function operation 1 are: input hand force, transmit hand force, transmit force, transmit solid material, and connect to pivot.

*Material operands for function operation 1 (supply and magnify force)*

The context of operation of solid A, which performs function operation 1, requires this solid to be a slender component similar to that shown in Figure 6.25. This topology is dictated by the fact that the hand force applied at point A is magnified and transmitted to another component



at point B. This is achieved by the rotation of solid A, as a result of the hand force, around point O (lever effect). The distance AB must be at least three times greater than the distance BO, in order to magnify the hand force. Therefore, as can be seen in Figure 6.25, there are three key points that are functionally needed as the points of interaction with the other components of the toggle clamp. These points, A, B, and O, are fixed distances apart and are required to maintain this spatial relation. Since solid A rotates during the operation, the locations of points A, B, and O change from the initial coordinates  $(X_i, Y_i, Z_i)$  to the final coordinates  $(X_f, Y_f, Z_f)$ .

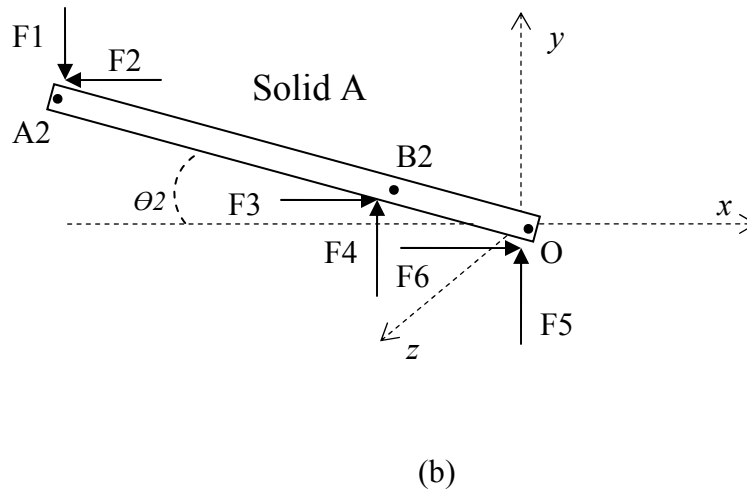
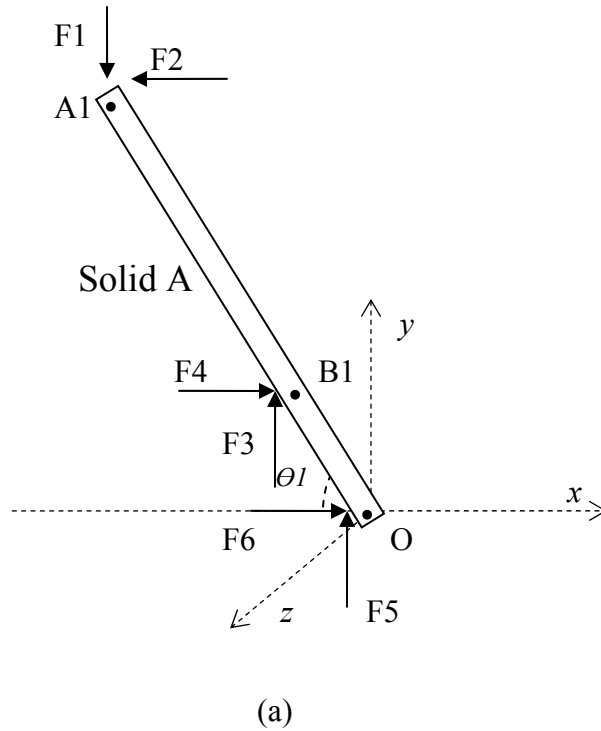


**Figure 6.25 Topology structure of solid operand (solid A) in function operation 1**

Energy operands for function operation 1 (supply and magnify force)

The second type of operand that is required for this operation is the force energy operand. The solid operand (solid A) is subjected to an external hand force at point A. In addition, there are reaction forces at points B and O (see Figure 6.26). The magnitude of forces, at points B and

O ( $F_3, F_4, F_5, F_6$ ), change during the operation from Min-load to Max-load. The magnitude of hand force at ( $F_1, F_2$ ) is assumed to be constant during operation. Moreover, the point of application of the forces ( $F_1, F_2, F_3, F_4$ ) changes due to the rotation of solid A around point O.



**Figure 6.26 Force operands ( $F_1, F_2, F_3, F_4, F_5, F_6$ ) interaction with solid A: (a) initial state. (b) final state**

### Stability and strength

Stability requirements ensure that the design yields a product that is stable during operation. This means that the forces acting on the product must be balanced. As a result, the strength of solid A must be sufficiently high to withstand the external forces. Also, the effect of all external forces must cancel out as a necessary condition for equilibrium.

### STEP III:

#### Functionality modeling for operation 1

In the functionality model, operands and relations are instantiated as design objects with their corresponding attributes. The generic functional model of the function operation 1 is defined as:

$$F_i = \{(o,r,s) \mid o \in O_i, r \in R_i \text{ and } s \in S_i\}$$

Where,

**i** : functionality operating index.

**O<sub>i</sub>**: a set of functionality operands.

**R<sub>i</sub>**: a set of functionality relations.

**S<sub>i</sub>**: a set of functionality states.

### Operands and attributes

For a set of operand **O<sub>i</sub>** in the functionality operation with an index **i**, each member of this operand set is given by:

$$O_{iq} = \{ a_{iqs} \mid a_{iqs} \in A_{iq} \}$$

Where,

**i**: functionality operation index.

**q**: functionality operand index.

**A<sub>iq</sub>**: attribute set for functionality operand **O<sub>iq</sub>**.

The functionality operands for function operation 1 are listed below in Table 6.1. A detailed description of these operands and their corresponding attributes is presented in Tables 6.2-6.8. In these tables, if the operand attribute is not defined at this stage, the word “Unknown” is displayed.

**Table 6.1 Operand list for function operation 1**

<b>j</b>	<b>Operand name</b>	<b>Operand type</b>
1	Solid A	Solid material
2	F1	Force
3	F2	Force
4	F3	Force
5	F4	Force
6	F5	Force
7	F6	Force

The strength of a solid operand depends on material type and geometry (cross-sectional area), as well as on the type of loading. In this example, uniform cross-sectional area is assumed for solid A. Therefore, the stress is given by  $F/A$ , where  $F$  is the applied force and  $A$  is the cross-sectional area. This stress (normal or shear) must not exceed the allowable stress limit for a given material. The allowable stress must be less than the yield stress ( $S_y$ ) by a safety factor. In general, the allowable stress is bounded by the following constraints:

- Allowable normal stress ( $N/m^2$ ):  $0.45 S_y \leq S_N \leq 0.6 S_y$

- Allowable shear stress ( $\text{N/m}^2$ ):  $0.4 S_y \leq S_s$

**Table 6.2 Functionality attributes of solid material operand; solid A**

Attribute	Description	Value	
		Initial state	Final state
Functional Markers	Point A	$(x_{A1}, y_{A1}, z_{A1})$	$(x_{A2}, y_{A2}, z_{A2})$
	Point B	$(x_{B1}, y_{B1}, z_{B1})$	$(x_{B2}, y_{B2}, z_{B2})$
	Point O	$(x_O, y_O, z_O)$	Same as initial
	$dist(A, O, d_1)$	200 mm	Same as initial
	$dist(B, O, d_2)$	50 mm	Same as initial
	Angle of <i>line</i> (AO) (from negative x direction)	$\Theta_1 (30^\circ)$	$\Theta_2 (2^\circ)$
Length	$dist(A, O, d_1)$	200 mm	Same as initial
Strength	Normal (tensile or compression)	$0.6 S_y$	Same as initial
	Shear	$0.4 S_y$	Same as initial
Material Type	High strength metal	STEEL	Same as initial
Mass Properties	Mass	Unknown	Unknown
	Area	Unknown	Unknown
	Volume	Unknown	Unknown
DoF	Degree of Freedom	<i>rot-z</i>	Same as initial
Role	Perform the function	Performer	Same as initial

**Table 6.3 Functionality attributes of force energy operand; F1**

Attribute	Description	Value	
		Initial state	Final state
Magnitude	$ F1 $	116 N	116 N
Direction	Negative y direction	y-axis	Same as initial
Point of application	Point A	$(x_{A1}, y_{A1}, z_{A1})$	$(x_{A2}, y_{A2}, z_{A2})$
Source	External hand force	External force	Same as initial
Kind	Contact force	Contact	Same as initial
Nature		Steady	Same as initial

**Table 6.4 Functionality attributes of force energy operand; F2**

Attribute	Description	Value	
		Initial state	Final state
Magnitude	F2	67 N	67 N
Direction	Negative x direction	x-axis	Same as initial
Point of application	Point A	( $x_{A1}, y_{A1}, z_{A1}$ )	( $x_{A2}, y_{A2}, z_{A2}$ )
Source	External hand force	External force	Same as initial
Kind	Contact force	Contact	Same as initial
Nature		Steady	Same as initial

**Table 6.5 Functionality attributes of force energy operand; F3**

Attribute	Description	Value	
		Initial state	Final state
Magnitude	F3	308.2 N	266.7 N
Direction	Positive y direction	y-axis	Same as initial
Point of application	Point B	( $x_{B1}, y_{B1}, z_{B1}$ )	( $x_{B2}, y_{B2}, z_{B2}$ )
Source	External reaction force	External force	Same as initial
Kind	Contact force	Contact	Same as initial
Nature		Steady	Same as initial

**Table 6.6 Functionality attributes of force energy operand; F4**

Attribute	Description	Value	
		Initial state	Final state
Magnitude	F4	534 N	7,646 N
Direction	Positive x direction	x-axis	Same as initial
Point of application	Point	( $x_{B1}, y_{B1}, z_{B1}$ )	( $x_{B2}, y_{B2}, z_{B2}$ )
Source	External reaction force	External force	Same as initial
Kind	Contact force	Contact	Same as initial
Nature		Steady	Same as initial

**Table 6.7 Functionality attributes of force energy operand; F5**

Attribute	Description	Value	
		Initial state	Final state
Magnitude	F5	192.6 N	151.3 N
Direction	Negative y direction	y-axis	Same as initial
Point of application	Point O	(x <sub>O</sub> , y <sub>O</sub> , z <sub>O</sub> )	Same as initial
Source	External reaction force	External force	Same as initial
Kind	Contact force	Contact	Same as initial
Nature		Steady	Same as initial

**Table 6.8 Functionality attributes of force energy operand; F6**

Attribute	Description	Value	
		Initial state	Final state
Magnitude	F6	467 N	7,579 N
Direction	Negative x direction	x-axis	Same as initial
Point of application	Point O	(x <sub>O</sub> , y <sub>O</sub> , z <sub>O</sub> )	Same as initial
Source	External reaction force	External force	Same as initial
Kind	Contact force	Contact	Same as initial
Nature		Steady	Same as initial

Relations

For the operand set  $O_i$ , the functionality relation  $R_i$  is defined as:

$$R_i = \{ r_{ijk} (o_{ij}, o_{ik}) \mid o_{ij} \in O_i \text{ and } o_{ik} \in O_i \}$$

Where,

$O_i$  = set of operands in functionality operation i.

$r_{ijk}$  = relation between operands j and k.

$j, k$  = functionality operand indices.

The relations in this functionality operation are listed below in Table 6.9. A detailed description of these relations is presented in Tables 6.10-6.16. The most important constraint is to ensure the satisfaction of the stress requirements. The instantaneous internal forces (as a result of the stress) along the solid operand must be greater than the applied load. For a given material of known yield stress ( $S_y$ ), the allowable normal stress  $\sigma_{allow} = 0.65 S_y$ , and the allowable shear stress  $\tau_{allow} = 0.4 S_y$ . Consequently, the maximum allowable internal force ( $F_{Sy}$ ) in the solid operand must be greater than the applied external loading ( $F_{applied}$ ), as shown in the following relations:

$$F_{Sy} = A \times \sigma_{allow} \geq F_{applied} \quad ,(\text{in normal stress case})$$

$$F_{Sy} = A \times \tau_{allow} \geq F_{applied} \quad ,(\text{in shear stress case})$$

Another important constraint is to ensure the stability and equilibrium. The equilibrium of the forces shall be maintained at all points in the product structure. The equilibrium conditions for the forces  $F_k$  in the coordinate directions and the moments  $M_k$  acting about these coordinates give:

$$\sum_{\forall x} F_k = 0; \sum_{\forall y} F_k = 0; \sum_{\forall x} M_k = 0; \sum_{\forall y} M_k = 0$$

Where,

$k (=1,2,3)$  are the nodal points corresponding to A, B, and O functional markers.



**Table 6.9 Relations in function operation 1**

Relation #	Interaction operands	Type of Operands	
1	Solid A – F1	SOLID-FORCE	Table 6.10
2	Solid A – F2	SOLID-FORCE	Table 6.11
3	Solid A – F3	SOLID-FORCE	Table 6.12
4	Solid A – F4	SOLID-FORCE	Table 6.13
5	Solid A – F5	SOLID-FORCE	Table 6.14
6	Solid A – F6	SOLID-FORCE	Table 6.15
7	F1, F2, F3, F4, F5, F6	FORCE-FORCE	Table 6.16

**Table 6.10 Description of the relation between solid A and F1 in function operation 1**

Coupling operands	Attributes	Relations
SOLID: solid A	$a_{111} = \langle \text{length} \rangle$ $a_{112} = \langle \text{normal strength} \rangle$ $a_{113} = \langle \text{shear strength} \rangle$ $a_{114} = \text{FM: } \langle \text{point } A \rangle$ $a_{115} = \text{angle of solid A with negative x direction : } \theta$	<u>Functional Relations:</u> $R_1 = \{  F_{Sy} _{\text{Normal}} = A \times \sigma_{\text{allow}} \}$ $R_2 = \{  F_{Sy} _{\text{Shear}} = A \times \tau_{\text{allow}} \}$ , A: cross-sectional area, $S_y$ : yield strength $\sigma_{\text{allow}} = 0.65 S_y$ , $\tau_{\text{allow}} = 0.4 S_y$ $R_3 = \{ \langle \text{application point} \rangle \text{ coincident point } A \}$
FORCE: F1	$a_{121} =  F1 $ $a_{122} = \text{dir}(F1)$ $a_{123} = \langle \text{nature} \rangle$ $a_{124} = \langle \text{application point} \rangle$	<u>Constraints:</u> $C_1 = \{  F_{Sy} _{\text{Normal}} \geq  F1  \cdot \sin\theta \}$ $C_2 = \{  F_{Sy} _{\text{Shear}} \geq  F1  \cdot \cos\theta \}$ <u>DoF:</u> $D1 = \{ \text{solid A: rot-z} \}$ $D2 = \{ \text{dir}(F1): \text{fix} \}$

**Table 6.11 Description of the relation between solid A and F2 in function operation 1**

Coupling operands	Attributes	Relations
SOLID: solid A	$a_{111} = \langle length \rangle$ $a_{112} = \langle normal\ strength \rangle$ $a_{113} = \langle shear\ strength \rangle$ $a_{114} = FM: \langle point\ A \rangle$ $a_{115} = \text{angle of solid A with negative x direction} : \theta$	<u>Functional Relations:</u> $R_1 = \{  F_{Sy} _{Normal} = A \times \sigma_{allow} \}$ $R_2 = \{  F_{Sy} _{Shear} = A \times \tau_{allow} \}$ , A: cross-sectional area, $S_y$ : yield strength $\sigma_{allow} = 0.65 S_y$ , $\tau_{allow} = 0.4 S_y$ $R_3 = \{ \langle application\ point \rangle \text{ coincident } point\ A \}$
FORCE: F2	$a_{131} =  F2 $ $a_{132} = dir(F2)$ $a_{133} = \langle nature \rangle$ $a_{134} = \langle application\ point \rangle$	<u>Constraints:</u> $C_1 = \{  F_{Sy} _{Normal} \geq  F2  \cdot \cos\theta \}$ $C_2 = \{  F_{Sy} _{Shear} \geq  F2  \cdot \sin\theta \}$  <u>DoF:</u> $D1 = \{ \text{solid A: } rot\text{-}z \}$ $D2 = \{ dir(F2): fix \}$

**Table 6.12 Description of the relation between solid A and F3 in function operation 1**

Coupling operands	Attributes	Relations
SOLID: solid A	$a_{111} = \langle length \rangle$ $a_{112} = \langle normal\ strength \rangle$ $a_{113} = \langle shear\ strength \rangle$ $a_{114} = FM: \langle point\ B \rangle$ $a_{115} = \text{angle of solid A with negative x direction} : \theta$	<u>Functional Relations:</u> $R_1 = \{  F_{Sy} _{Normal} = A \times \sigma_{allow} \}$ $R_2 = \{  F_{Sy} _{Shear} = A \times \tau_{allow} \}$ , A: cross-sectional area, $S_y$ : yield strength $\sigma_{allow} = 0.65 S_y$ , $\tau_{allow} = 0.4 S_y$ $R_3 = \{ \langle application\ point \rangle \text{ coincident } point\ B \}$
FORCE: F3	$a_{141} =  F3 $ $a_{142} = dir(F3)$ $a_{143} = \langle nature \rangle$ $a_{144} = \langle application\ point \rangle$	<u>Constraints:</u> $C_1 = \{  F_{Sy} _{Normal} \geq  F3  \cdot \sin\theta \}$ $C_2 = \{  F_{Sy} _{Shear} \geq  F3  \cdot \cos\theta \}$  <u>DoF:</u> $D1 = \{ \text{solid A: } rot\text{-}z \}$ $D2 = \{ dir(F3): fix \}$

**Table 6.13 Description of the relation between solid A and F4 in function operation 1**

Coupling operands	Attributes	Relations
SOLID: solid A	$a_{111} = \langle length \rangle$ $a_{112} = \langle normal\ strength \rangle$ $a_{113} = \langle shear\ strength \rangle$ $a_{114} = FM: \langle point\ B \rangle$ $a_{115} = \text{angle of solid A with negative x direction} : \Theta$	<u>Functional Relations:</u> $R_1 = \{  F_{Sy} _{Normal} = A \times \sigma_{allow} \}$ $R_2 = \{  F_{Sy} _{Shear} = A \times \tau_{allow} \}$ , A: cross-sectional area, $S_y$ : yield strength $\sigma_{allow} = 0.65 S_y$ , $\tau_{allow} = 0.4 S_y$ $R_3 = \{ \langle application\ point \rangle \text{ coincident } point\ B \}$
FORCE: F4	$a_{151} =  F4 $ $a_{152} = dir(F4)$ $a_{153} = \langle nature \rangle$ $a_{154} = \langle application\ point \rangle$	<u>Constraints:</u> $C_1 = \{  F_{Sy} _{Normal} \geq  F4  \cdot \cos\Theta \}$ $C_2 = \{  F_{Sy} _{Shear} \geq  F4  \cdot \sin\Theta \}$  <u>DoF:</u> $D1 = \{ \text{solid A: } rot\text{-}z \}$ $D2 = \{ dir(F4): fix \}$

**Table 6.14 Description of the relation between solid A and F5 in function operation 1**

Coupling operands	Attributes	Relations
SOLID: solid A	$a_{111} = \langle length \rangle$ $a_{112} = \langle normal\ strength \rangle$ $a_{113} = \langle shear\ strength \rangle$ $a_{114} = FM: \langle point\ O \rangle$ $a_{115} = \text{angle of solid A with negative x direction} : \Theta$	<u>Functional Relations:</u> $R_1 = \{  F_{Sy} _{Normal} = A \times \sigma_{allow} \}$ $R_2 = \{  F_{Sy} _{Shear} = A \times \tau_{allow} \}$ , A: cross-sectional area, $S_y$ : yield strength $\sigma_{allow} = 0.65 S_y$ , $\tau_{allow} = 0.4 S_y$ $R_3 = \{ \langle application\ point \rangle \text{ coincident } point\ O \}$
FORCE: F5	$a_{161} =  F5 $ $a_{162} = dir(F5)$ $a_{163} = \langle nature \rangle$ $a_{164} = \langle application\ point \rangle$	<u>Constraints:</u> $C_1 = \{  F_{Sy} _{Normal} \geq  F5  \cdot \sin\Theta \}$ $C_2 = \{  F_{Sy} _{Shear} \geq  F5  \cdot \sin\Theta \}$  <u>DoF:</u> $D1 = \{ \text{solid A: } rot\text{-}z \}$ $D2 = \{ dir(F5): fix \}$

**Table 6.15 Description of the relation between solid A and F6 in function operation 1**

Coupling operands	Attributes	Relations
SOLID: solid A	$a_{111} = \langle length \rangle$ $a_{112} = \langle normal\ strength \rangle$ $a_{113} = \langle shear\ strength \rangle$ $a_{114} = FM: \langle point\ O \rangle$ $a_{115} = \text{angle of solid A with negative x direction} : \theta$	<u>Functional Relations:</u> $R_1 = \{  F_{Sy} _{Normal} = A \times \sigma_{allow} \}$ $R_2 = \{  F_{Sy} _{Shear} = A \times \tau_{allow} \}$ , A: cross-sectional area, Sy : yield strength $\sigma_{allow} = 0.65 Sy$ , $\tau_{allow} = 0.4 Sy$ $R_3 = \{ \langle application\ point \rangle \text{ coincident } point\ O \}$
FORCE: F6	$a_{171} =  F6 $ $a_{172} = dir(F6)$ $a_{173} = \langle nature \rangle$ $a_{174} = \langle application\ point \rangle$	<u>Constraints:</u> $C_1 = \{  F_{Sy} _{Normal} \geq  F6  \cdot \cos\theta \}$ $C_2 = \{  F_{Sy} _{Shear} \geq  F6  \cdot \sin\theta \}$  <u>DoF:</u> $D1 = \{ \text{solid A: } rot-z \}$ $D2 = \{ dir(F6): fix \}$

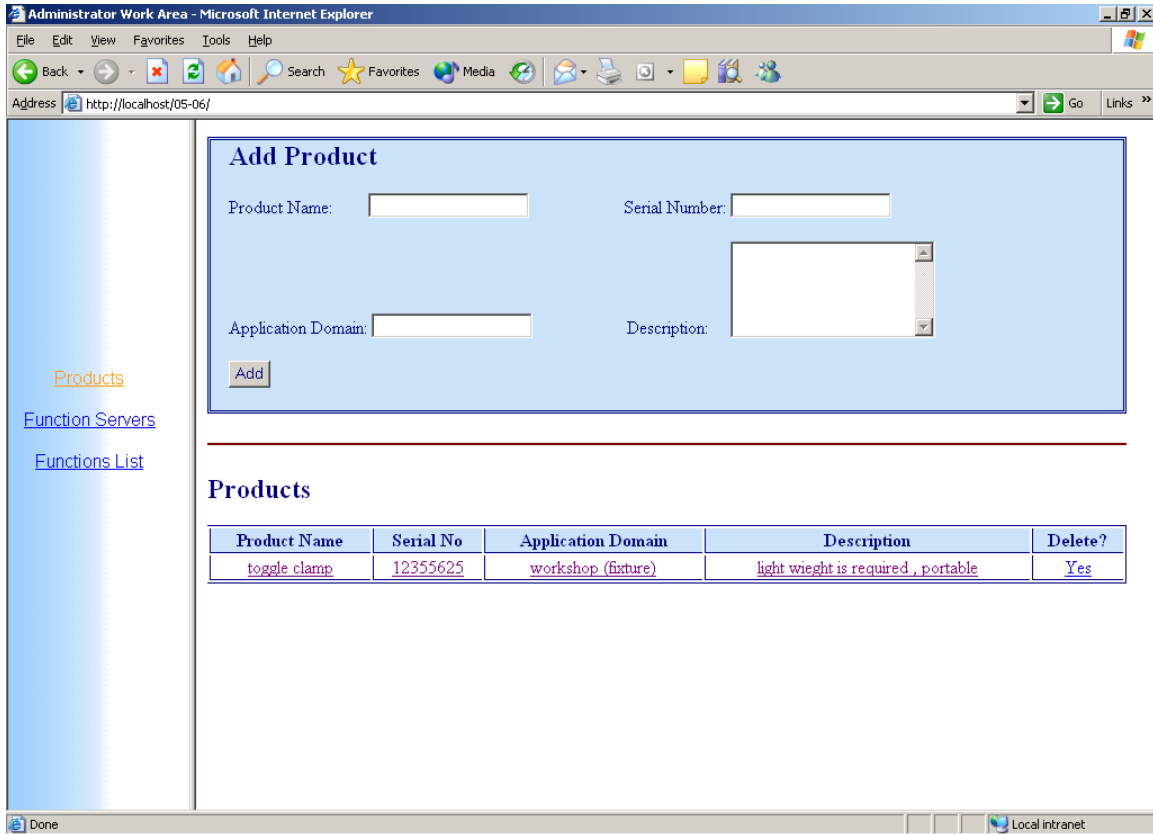
**Table 6.16 Equilibrium relation between force operands (F1, F2, F3, F4, F5, F6)**

Coupling operands	Attributes	Relations
FORCE Operands: F1, F2, F3, F4, F5, F6	$a_{1i1} =  Fi $ $a_{1i2} = dir(Fi)$ $a_{1i3} = \langle nature \rangle$ $a_{1i4} = \langle application\ point \rangle$	<u>Force Constraints:</u> $C_1 = \{ \sum F_x = 0 \mid F_2 + F_4 + F_6 = 0 \}$ $C_2 = \{ \sum F_y = 0 \mid F_1 + F_3 + F_5 = 0 \}$  <u>Moment Constraints:</u> $C_3 = \{ \sum_{A,B,O} M = 0 \}$

STEP IV: Search function driven database for solutions

In this step, the function driven database is accessed by the designer to search for potential solutions. The translation tool website is first opened. Then, the information about product name, serial number, application domain, and description are entered. After that, the

functions and sub-functions from the toggle clamp decomposition are entered and saved in the system. The operands, relations, and their attributes are also entered in the system. Screen captures of these steps are shown in Figures 6.27-6.29.



**Figure 6.27 Adding toggle clamp as a new product**

Administrator Work Area - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media

Address http://localhost/05-06/ Go Links

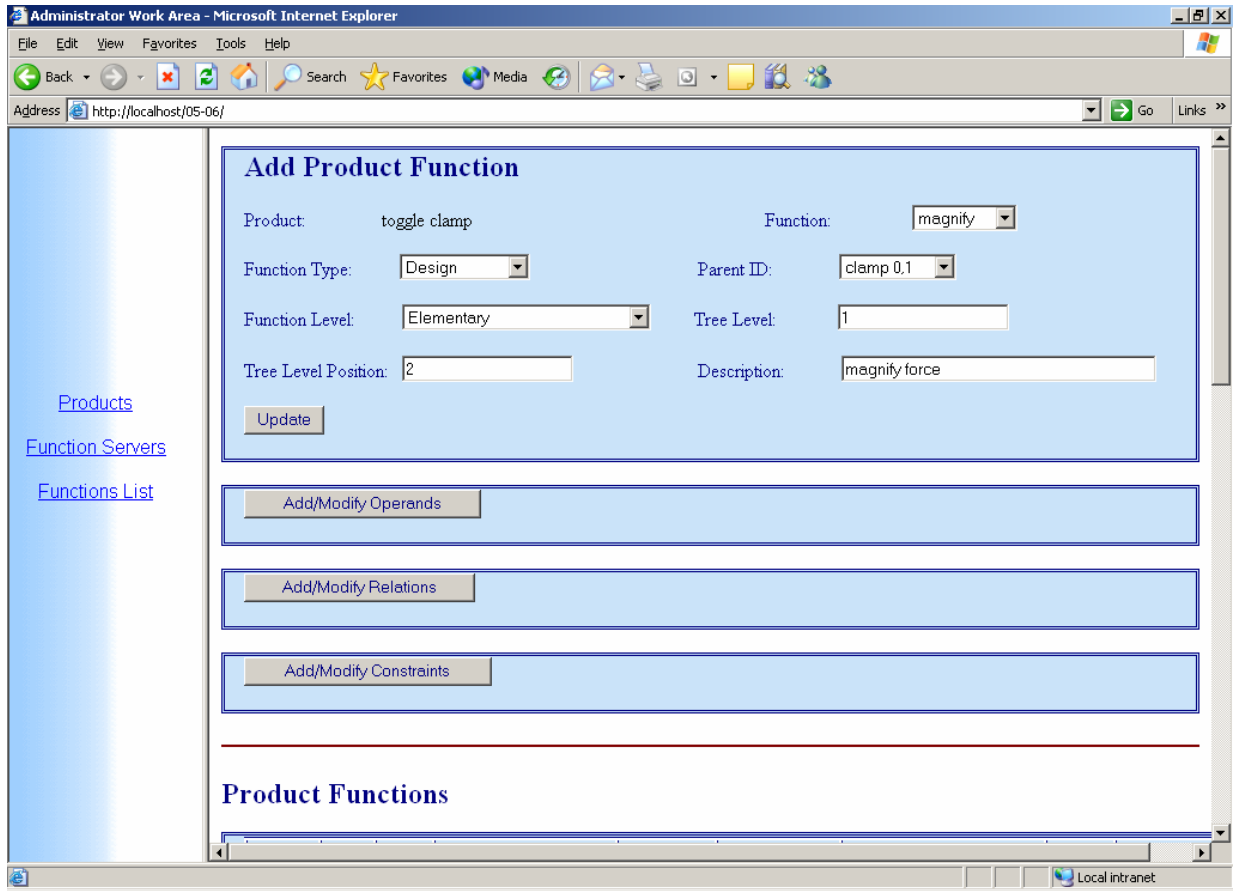
### Product Functions

Function	Type	Parent	Function_Level	Tree_Level	Level_Position	Description	Delete?	Select for Search
<a href="#">clamp</a>	<a href="#">Design</a>	0	<a href="#">Overall</a>	0	1	<a href="#">Clamp solid object against fixed surface</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">supply</a>	<a href="#">Design</a>	65	<a href="#">Elementary</a>	1	1	<a href="#">supply force</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">magnify</a>	<a href="#">Design</a>	65	<a href="#">Elementary</a>	1	2	<a href="#">magnify force</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">change</a>	<a href="#">Design</a>	65	<a href="#">Elementary</a>	1	3	<a href="#">change force direction</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">plunge</a>	<a href="#">Design</a>	65	<a href="#">Elementary</a>	1	4	<a href="#">plunge into the object</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">support</a>	<a href="#">Design</a>	65	<a href="#">Elementary</a>	1	5	<a href="#">support components and reaction force</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">input</a>	<a href="#">Design</a>	66	<a href="#">Embodiment(Part Level)</a>	2	1	<a href="#">input hand force</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">transmit</a>	<a href="#">Design</a>	66	<a href="#">Embodiment(Part Level)</a>	2	2	<a href="#">transmit hand force</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">rotate</a>	<a href="#">Design</a>	67	<a href="#">Embodiment(Geometric Level)</a>	2	3	<a href="#">rotate around pivot</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">connect</a>	<a href="#">Design</a>	65	<a href="#">Embodiment(Geometric Level)</a>	2	4	<a href="#">connect to pivot</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">support</a>	<a href="#">Design</a>	65	<a href="#">Embodiment(Part Level)</a>	2	5	<a href="#">support force</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">connect</a>	<a href="#">Design</a>	68	<a href="#">Embodiment(Geometric Level)</a>	2	6	<a href="#">connect to linkage</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">transmit</a>	<a href="#">Design</a>	68	<a href="#">Embodiment(Part Level)</a>	2	7	<a href="#">transmit force</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">transmit</a>	<a href="#">Design</a>	68	<a href="#">Embodiment(Part Level)</a>	2	8	<a href="#">transmit solid material</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">guide</a>	<a href="#">Design</a>	69	<a href="#">Elementary</a>	2	9	<a href="#">guide motion</a>	<a href="#">Yes</a>	<input type="checkbox"/>
<a href="#">move</a>	<a href="#">Design</a>	69	<a href="#">Elementary</a>	2	10	<a href="#">move to plunge</a>	<a href="#">Yes</a>	<input type="checkbox"/>

Products  
Function Servers  
Functions List

Done Local intranet

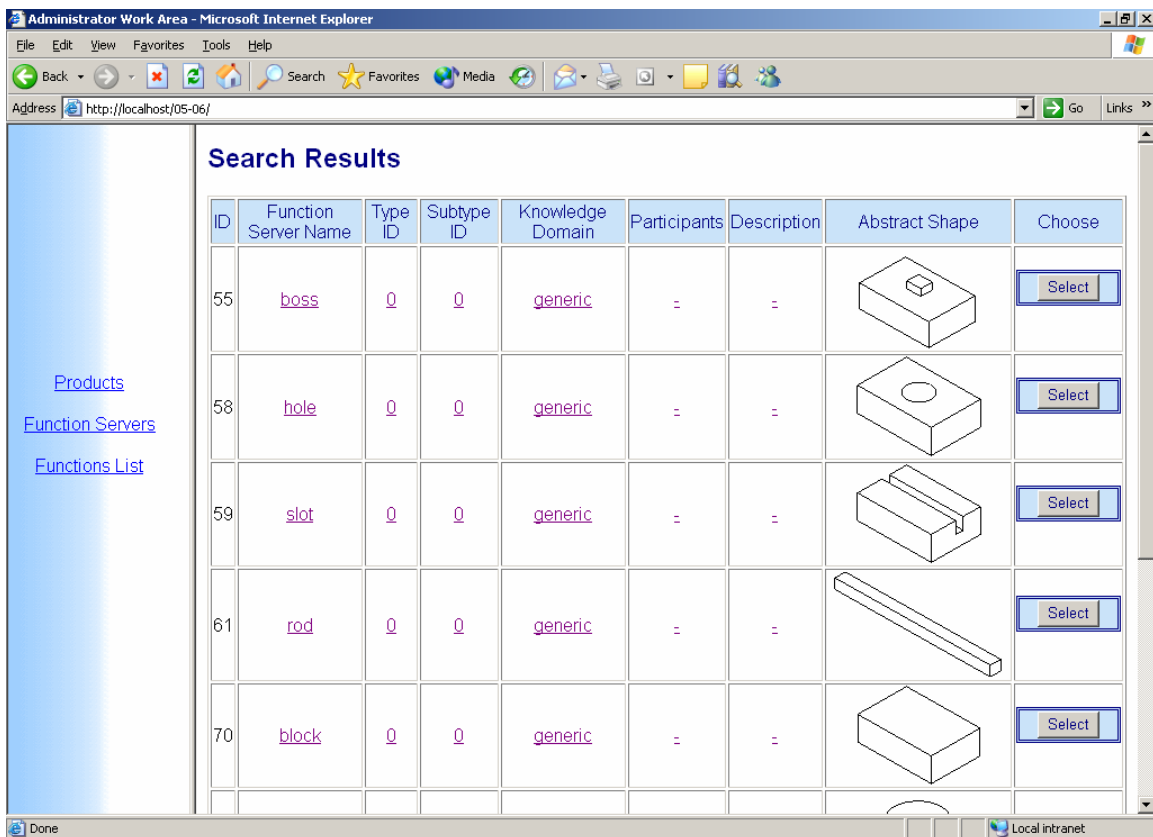
Figure 6.28 Function decomposition structure of the toggle clamp



**Figure 6.29 Add/Modify function model components (operands, relations, and constraints) of the toggle clamp**

To conduct a search, the designer must first define the functions, individual and combined, to be used in the search. To do this, the designer should go over the function decomposition and try to figure out the interactions between material and energy. In addition, the designer should try to distinguish and specify part-level functions and geometric-level functions. For example, the lower level sub-functions, input hand force, transmit hand force, transmit force, and transmit material by rotation, can be used as one combined function for conducting a search. These functions are chosen by checking the checkbox to the right of each of them. The Search

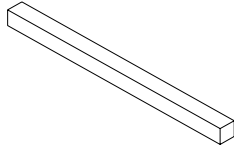
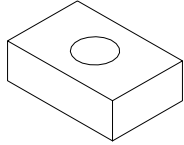
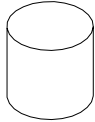
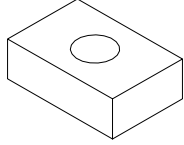
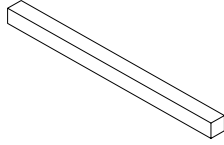
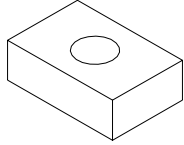

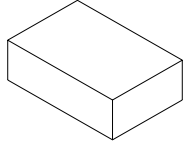
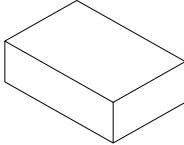
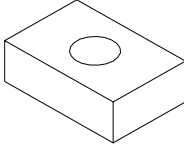
button is then pressed. Function servers are searched for and the following solutions, as shown in Figure 6.30, are found: boss, hole, slot, rod, block, cylinder, and sphere. Of these, the rod is chosen as the solution by pressing the Select button. This solution is then automatically saved, with its related functions, in the product function server list (i.e., database). Table 6.17 presents the functions used for the search and the chosen solution for each group in this example.



**Figure 6.30 Search process results**



**Table 6.17 Sub-functions used to search with, and solutions**

<b>Group #</b>	<b>Function(s)</b>	<b>Solution (function server)</b>	<b>Abstract shape</b>
1	Input hand force Transmit force Transmit material	Rod	
2	Connect to pivot	Hole	
3	Support force	Cylinder	
4	Connect to linkage	Hole	
5	Transmit force Transmit material	Rod	
6	Allow passage Support Guide	Hole	
7	Transmit force Transmit material	Cylinder	
8	Hold the object Transmit material	Block	
9	Support components	Block	
10	Fasten with fixed surface	Hole	

## STEP V: Function server modeling

In the function server model, function servers are instantiated as design objects with their corresponding attributes. The function server model is defined as:

**FS:** {**FD**, **SH**, **M**, **MI**, **IN**, **WC**}

Where,

**FD:** function description.

**SH:** shape description.

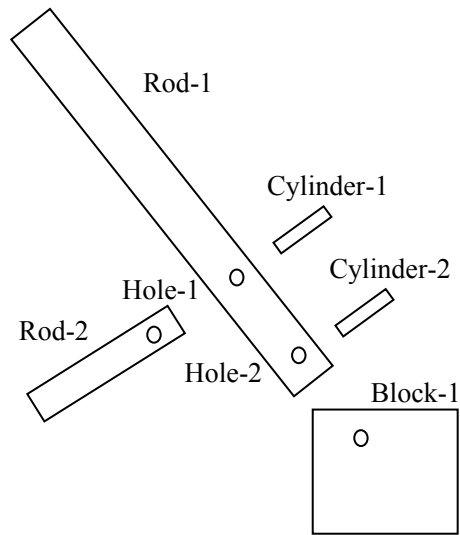
**M:** material model.

**MI:** manufacturing information.

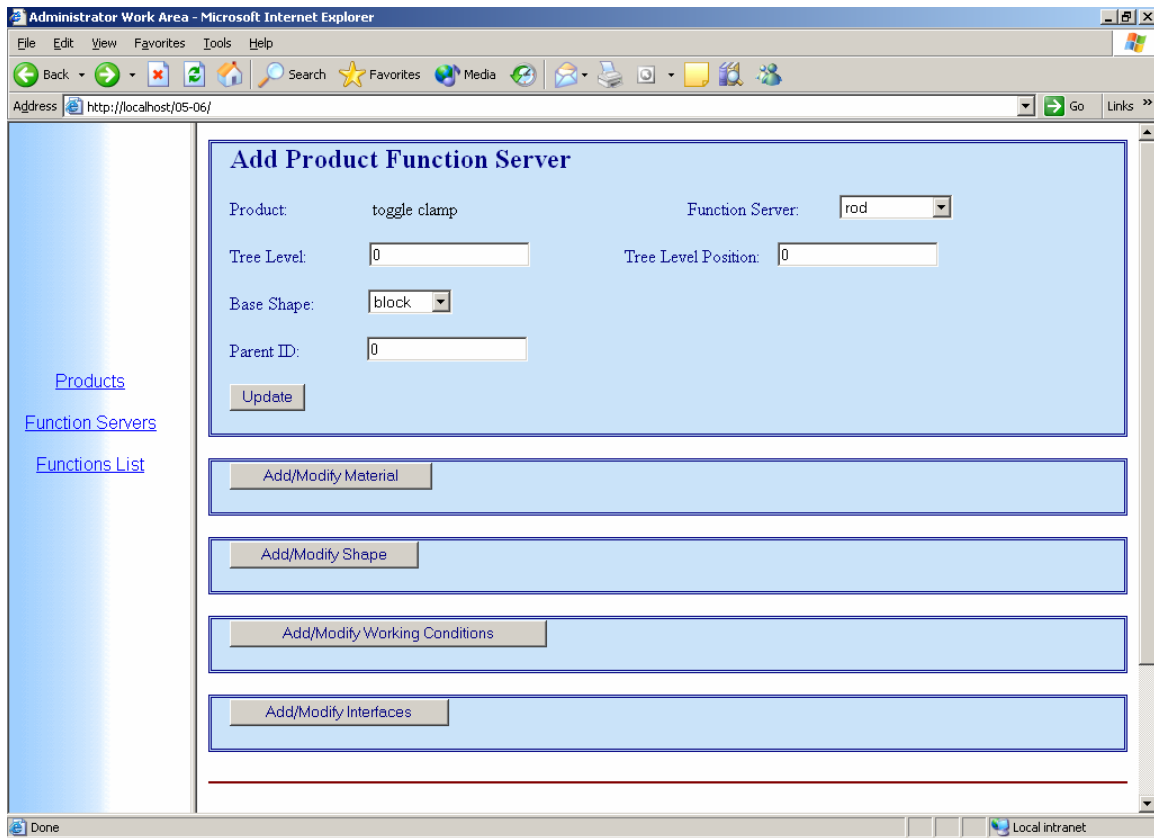
**IN:** interface description.

**WC:** working conditions.

A detailed description of the function server model for solution 1 (rod-1) is presented in Table 6.18. Rod-1 interfaces with six other function servers: hole-1, hole-2, cylinder-1, cylinder-2, rod-2, and block-1. Figure 6.31 illustrates this interaction. Information concerning the function servers is also saved in the product function server list (i.e., database). The screen capture of Figure 6.32 illustrates this process.

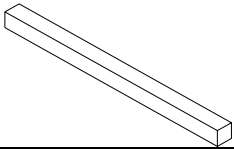


**Figure 6.31 Interface of function server rod-1 with other function servers**



**Figure 6.32 User page to add/modify function server model**

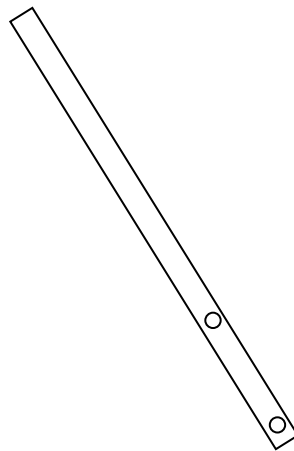
**Table 6.18 Description of Function server model for solution 1 (Rode-1)**

<b>Function Server Name</b>	Rod-1
<b>Function</b>	<u>Function(s) served:</u> <i>Input hand force , Transmit force, Transmit material</i>
<b>Shape</b>	<u>Base shape:</u> <i>block</i> <u>Abstract shape:</u> 
<b>Material</b>	<u>Material type:</u> <i>steel</i> <u>Known material properties:</u> <i>strength</i>
<b>Manufacturing</b>	<u>Manufacturing process:</u> <i>forming (extruding)</i> <u>Surface Properties:</u> <i>Unknown</i> <u>Alternate material:</u> <i>Unknown</i>
<b>Working Condition</b>	<u>Energy type:</u> <i>mechanical energy</i> <u>Load nature:</u> <i>steady</i> <u>Physical phenomena:</u> <i>stress, lever effect</i> <u>Failure modes:</u> <i>stress rupture, yielding</i>
<b>Interface</b>	<p><u>Interface 1:</u> Interface with <i>hole-1</i> Interface type: <i>permanent (part_of)</i> Interface location: <i>point B</i></p> <p><u>Interface 2:</u> Interface with <i>hole-2</i> Interface type: <i>permanent (part_of)</i> Interface location: <i>point O</i></p> <p><u>Interface 3:</u> Interface with <i>cylinder-1</i> Interface type: <i>permanent (flexible)</i> Spatial relationship: <i>align ( centerlines of hole-1 and cylinder-1)</i> DoF: {rod-1: <i>rot-z</i>, cylinder-1: <i>cir-z</i>}</p> <p><u>Interface 4:</u> Interface with <i>cylinder-2</i> Interface type: <i>permanent (flexible)</i> Spatial relationship: <i>align ( centerlines of hole-2 and cylinder-2)</i> DoF: {rod-1: <i>rot-z</i>, cylinder-2: <i>fix</i>}</p> <p><u>Interface 5:</u> Interface with <i>rod-2</i> Interface type: <i>permanent (flexible)</i> Spatial relationship: <i>against (contact surfaces)</i> DoF: {rod-1: <i>rot-z</i>, rod-2: <i>plane-xy</i>}</p> <p><u>Interface 6:</u> Interface with <i>block-1</i> Interface type: <i>permanent (flexible)</i> Spatial relationship: <i>against (contact surfaces)</i> DoF: {rod-1: <i>rot-z</i>, block-1: <i>fix</i>}</p>

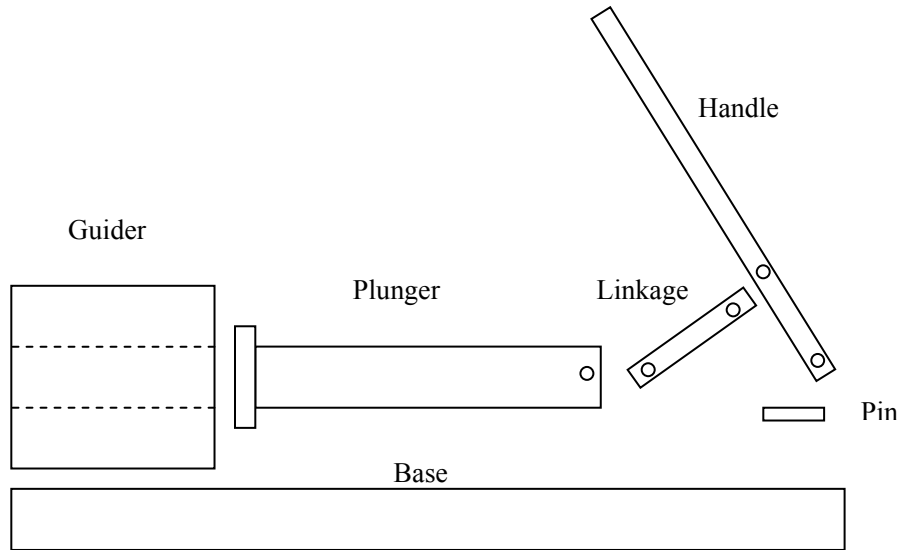
## STEP VI: Construct the conceptual form of toggle clamp

Going back to table 6.18, the solutions rod, hole (with function: connect to pivot), and hole (with function: connect to linkage) can be combined into one part, as shown in Figure 6.33. This part is given the name, “handle”. The cylinder solution (with function: support force) can be a pin. The other rod solution (with functions: transmit force and transmit material) can be a component used to change the direction of the magnified force from vertical to horizontal. This component is given the name “linkage”.

The cylinder solution (with functions: transmit force and transmit material) and the block solution (with functions: hold the object and transmit reaction force) can be collapsed into a single component. This component is given the name “plunger”. The hole solution (with functions: allow passage, support, and guide) can be one component with a cylindrical base shape. This component is given the name “guider”. Finally, the block solution (with function: support components) can be the base that supports the reaction force and the other components. The conceptual form of the toggle clamp components is shown in Figure 6.34.

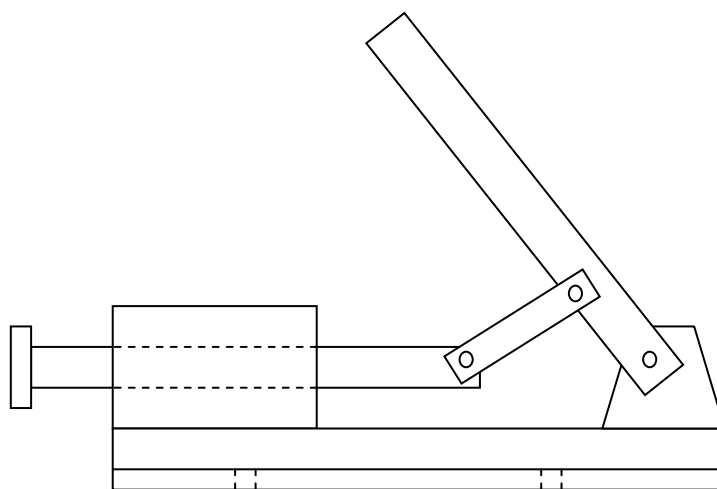


**Figure 6.33 Conceptual form of toggle clamp's handle**

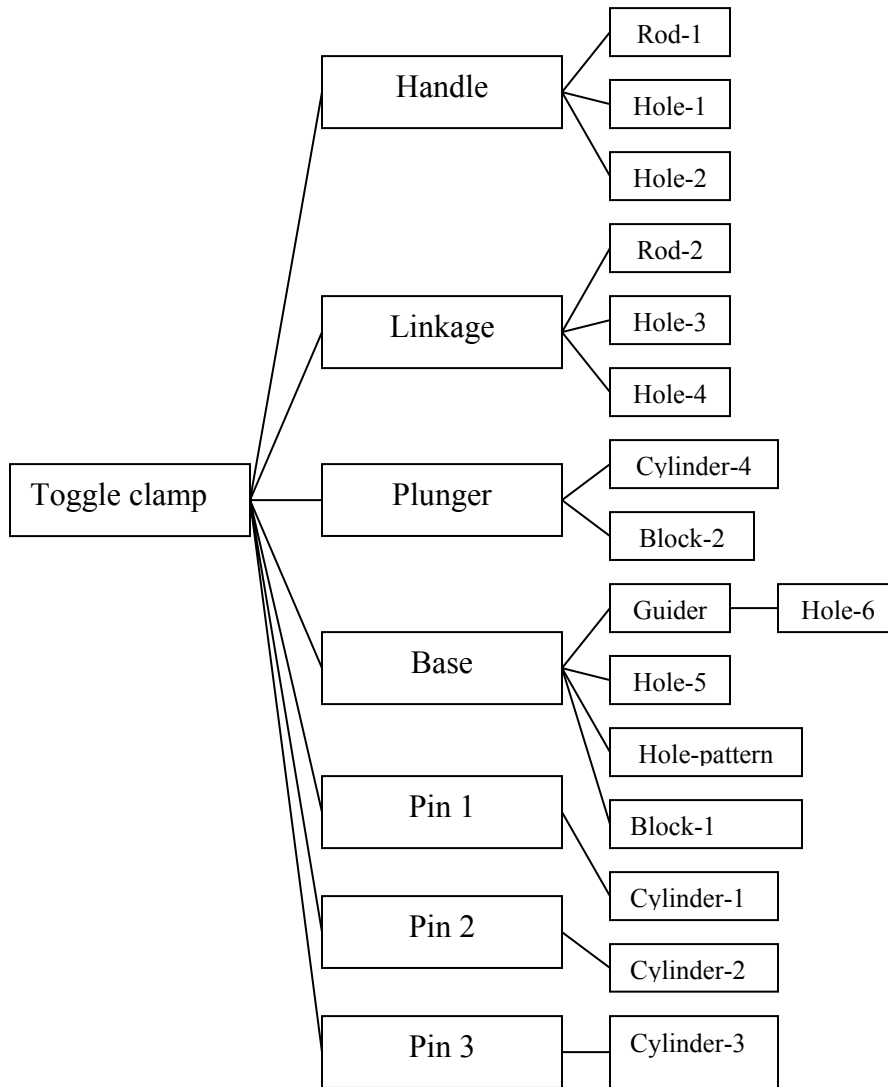


**Figure 6.34 Conceptual forms of toggle clamp components**

The base should have a hole to connect with the handle and a pattern of holes to fasten to the fixed surface. In addition, the guider can be combined into the base to form one component. The final conceptual form of the toggle clamp is shown in Figure 6.35. A complete function server tree is shown in Figure 6.36.



**Figure 6.35 The conceptual form of the toggle clamp assembly**



**Figure 6.36 Function server tree for toggle clamp**

### 6.5.3 Experimental Test

An experiment had been designed to test and evaluate the efficiency and effectiveness of the translation tool. In this experiment the evolution of toggle clamp conceptual design was traced and recorded for four separate designers. Two designers had performed the conceptual design of toggle clamp by using traditional design methods. The other two designers had used the translation tool developed in this research to come up with the required conceptual design. In

both cases the designers were given only the objective function with some requirements and constraints without knowing the name of the product. The period times for finishing the conceptual design steps were recorded over several separate design sessions on subsequent days for each individual designer. These period records are shown in table 6.19.

**Table 6.19 The recorded design period times**

Step Name	Period time (min)	
	Traditional Design Methods	Translation Tool
Preliminary meeting	30	30
Planning & defining the problem	67.5	65
Constructing function structure	42.5	20
Concept & physical embodiment	58	12.5
Conceptual form sketching	75	57.5
Total Time period	273 (4 hr 33min)	185 (3 hr 5min)

As we can see in table 6.19, the traditional conceptual design group encompassed a total design period of 4 hours and 33 minutes, while the group who used the translation tool encompassed a total design period of 3 hours and 5 minutes. This reduction in designing time is achieved because the translation tool helps speed up the process of finding physical embodiments for the specified function requirements. Also, it is easier and faster to sketch the conceptual shape of the product when the primitive function servers are available to the designer. The function driven database also helps expand the designer's thinking by offering options that he or she might not have thought of. The difference in period times between the two cases will extend more as the complexity and the number of the components in the product increase. Hence we can achieve a significant save in time.



Another important issue to compare is the design documentation. The group who used the translation tool had utilized the available models to completely document and organize the product's functional and physical information. This includes what was developed and the reasons why it was developed. In this case the conceptual product information was efficiently and clearly managed and represented. As a result, when a feature, component, or assembly is modified the designer can then be aware of which functions and which parts of the specification may be affected. On the other hand, the design documentation of the traditional design group lacked this organized documentation of information, which in turn reduced the opportunity to reuse the current design data.

## **7.0 CONCLUSIONS AND FUTURE WORK**

### **7.1 CONCLUSIONS**

This research provides a framework to support translating function specifications to conceptual forms during the conceptual design phase. It provides a tool for transforming an abstract representation of a product into a more concrete representation. This tool helps to map product needs, given in the form of a problem statement and preferences, via functional specification analysis into a functional data model describing all the functional aspects of the product's design. This research project has developed a functional data model to provide a complete description of the product by means of functionality relations and constraints imposed on the physical resources used in a given task. The physical resources required to accomplish a function have been described in the form of functionality operands (solid material and energy). In addition, a new function classification scheme based on operand interaction has been introduced. This classification helps the designer to create the function decomposition and to identify the function operations in the functionality model.

The translation tool developed in this research assists in translating the function specifications into physical embodiments by accessing and extracting solutions from a function driven database. The translation tool targets the functionality and relations possessed by a physical element (i.e., feature, component, or sub-assembly) as the primary solution search source. These solutions and all related information, including functionality, are represented and captured in a data structure known as a function server model. The function server model was

developed to provide a conceptual physical description of the product as regards material, shape, manufacturing information, working conditions, and interface with other objects.

The models and concepts developed in this work have been implemented and tested in a computer system. The object-oriented Unified Modeling Language (UML) was used to construct the object models. The demonstration was accomplished by building a web-based function-to-conceptual form translation tool using MySQL, PHP, and Apache server technology. Specifically, the following computer tools were developed to realize this research project and to support function-oriented design:

- The functional data model, function server model, function driven database, and conceptual product model have been implemented as a relational dataset system using MySQL.
- A dynamic web-based graphic user interface was developed using PHP to provide an interactive environment for the modeling of conceptual product information and for the searching of possible solutions.
- Data structures for the developed models were created using the object modeling approach. This data modeling approach provides an easy and flexible means for managing functionality and function server information.
- Propagation and exchange of conceptual product information to downstream design activates was realized using the XML data representation schema.

The translation tool developed in this dissertation speeds up the process of moving from objective function to the realization of conceptual form. It supports and facilitates conceptual design for both experienced and novice designers. With this program, the designer can try out a

greater number of iterations in less time. The function driven database helps expand the designer's thinking by offering options that he or she might not have thought of. The conceptual product model documents the search results and organizes the product's functional and physical information, which, in turn, makes it easy for designers from different backgrounds to access and use the design information. The fact that the system is Internet based reduces the communication expense, allows for remote collaborative work, and makes the system platform-independent.

The proposed function-to-conceptual form translation tool provides a framework that allows a designer to carry out conceptual design with the aid of a computer. It also serves as a bridge between the conceptual design phase and the detailed design phase of a product.

## **7.2 FUTURE WORK**

The opportunities that could be pursued further for extending this work are listed below:

1. The set of operands covered in this work includes solid materials and mechanical energy. Future work would extend this set of operands to include all material and energy operands.
2. The results of this work have been restricted to the design of mechanical devices. Future work could extend the results to include the design of other engineering products.
3. The function driven database developed in this work includes knowledge base about only functional features. This database could be extended to encompass other classes of primitive function servers (i.e. functional standard parts and functional modules).
4. A possible extension of this work is to integrate a decision-making tool with the system to help the designer in the selection of competing function servers. The future work in this part should investigate the possible classes of criteria and metrics that could be used

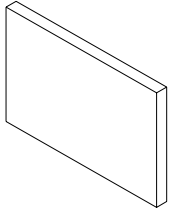
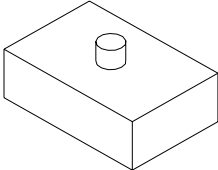
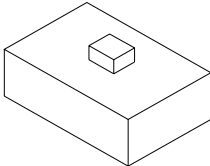
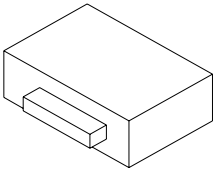
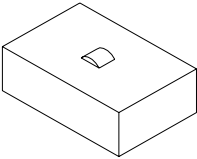
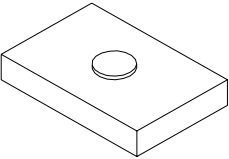
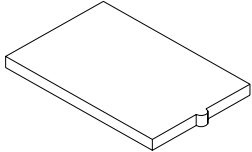
to rank and compare the solutions. This could be also extended to include overall conceptual forms.

5. In the implementation, the constraints have not been used during the search for possible solutions. An extension to include constraints in the search process could be implemented in future versions of this translation tool.
6. A future work could be conducted to extend the shape representation in function server model to include dimension and tolerance representation
7. For practical use of the function server knowledge base (i.e. the list of functions for each function server that are introduced in this work), extensive data needs to be collected from industry, and should cover different manufacturing domains to improve this knowledge base.
8. Converting the natural languages of customers to structured function specifications is an open issue that could be considered in future work.
9. Exploring the integration of the translation tool into commercial CAD systems to give the designer a more flexible design environment and reduce the gap with detailed design stage could be an opportunity to improve this work further.

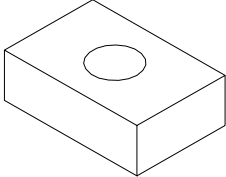
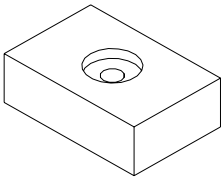
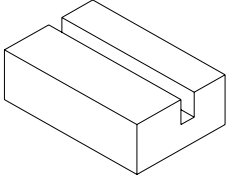
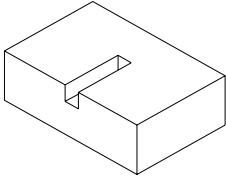
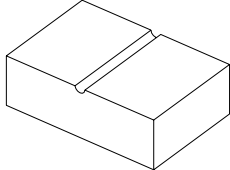
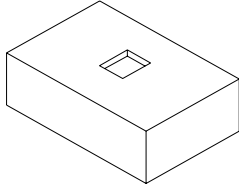
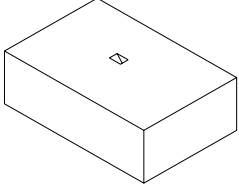
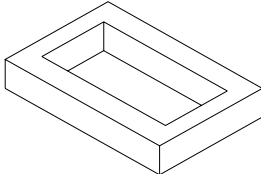
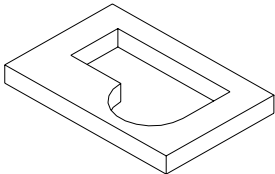
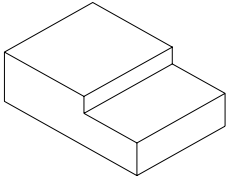
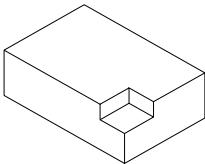
## APPENDIX A

### ABSTRACT SHAPES OF FUNCTIONAL FEATURES

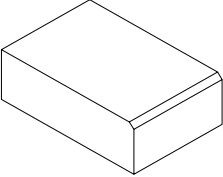
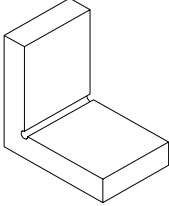

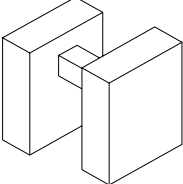
**Table A.1 Abstract shapes of functional features**

Functional Feature Class	Functional Feature Subclass	Abstract Shapes
Walls		
Additives	Boss	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">             Rotational         </div> <div style="text-align: center;">             Prismatic         </div> </div>
	Rib/gusset/web	
	Tab/flange/protrusion	<div style="display: flex; justify-content: space-around; align-items: center;">    </div>

**Table A.1 (continued)**

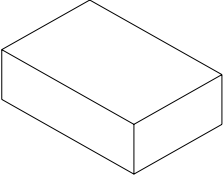
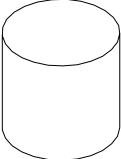
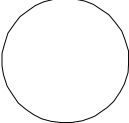
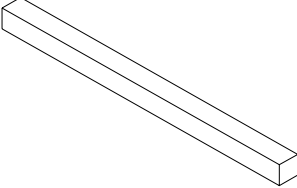
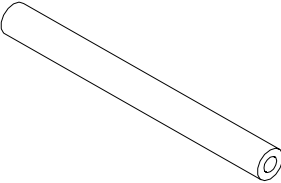
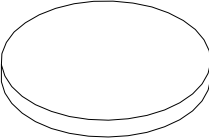
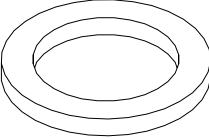
Functional Feature Class	Functional Feature Subclass	Abstract Shapes	
Subtractives	Hole	 <p>Regular hole ( Blind or through ) ( Threaded or not threaded )</p>	 <p>Countersink ( Blind or through ) ( Threaded or not threaded )</p>
	Slot	 <p>Through slot</p>	 <p>Blind slot (keyway)</p>
	Groove		
	Pocket/indent /depression		
	Window / cut-out		
	Step/corner		

**Table A.1 (continued)**

Functional Feature Class	Functional Feature Subclass	Abstract Shapes
Beside walls	Chamfer/fillet	
	Groove /undercut	
	Rib/web	
	Bridge	



**Table A.1 (continued)**

Functional Feature Class	Functional Feature Subclass	Abstract Shapes
Solid elements	Block	
	Cylinder	
	Sphere	
	Rod	
	Tube	
	Disk	
	Ring	

## APPENDIX B

### LIST OF FUNCTIONS USED IN FUNCTION DRIVEN DATABASE

**Table B.1 List of functions and synonyms**

Function	Synonyms
access	Pass, enter, input.
align	Straighten, adjust, line up, follow.
allow	Permit, let, approve, grant.
assemble	Fabricate, combine, bring together, gather, unite.
assist	Help, aid, work for, work with.
balance	Stabilize, steady, equal, adjust, level.
block	Obstruct, bar, stop.
connect	Attach, join, couple, link, fasten, mate, engage, mesh.
control	Regulate, limit, constrain, adjust, modify, restrict.
eject	Dispose, remove, discharge, export, bump.
enclose	Cover, contain, shield, protect, surround, include, guard, hide, wrap, insulate.
fit	Agree, conform, match, meet, go together.
friction	Rub, resist.
guide	Direct, straighten, steer, control, regulate, lead.
hold	Stop, lock, remain, stay, secure.
mount	Secure, lock, fasten, hold, attach, fix.
position	Orient, locate, place, put, set, settle, stand, occupy.
prevent	Avoid, hinder, prohibit, restrict, inhibit.
protect	Guard, cover, care for, keep, insulate, keep safe.
reduce	Decrease, step down.
reflect	Bend back, mirror, reverse, throw back, return.
rigidify	Increase rigidity, provide rigidity.
seal	Close, fasten, secure, shut, stop.
space	Separate, provide gap, offset.
strengthen	Add, confirm, harden, mount, increase strength.
support	Load bear, base, carry, hold, sustain, offset.
transform	Convert, change, alter
transmit	Convey, conduct, transport, move, lift, transfer, translate, rotate, conduct.
view	Look, see, watch, check out, inspect.

## **APPENDIX C**

### **FUNCTIONS OF FUNCTIONAL FEATURES**

**Table C.1 Functional features and their possible functions in generic domain**

<b>Functional Feature Class</b>	<b>Functional Feature Subclass</b>	<b>Functions</b>
Walls		Block, reflect, cover, enclose, protect, support, strengthen, hold, position, friction, connect.
Additives	Boss	Position, guide, support, transmit, connect, eject, mount, assist, strengthen, offset.
	Rib/gusset /web	Strengthen, support, guide, hold, position, connect, assist, rigidify.
	Tab/flange /protrusion	Hold, position, align, support, connect, stop, assemble, block, rigidify, fit, balance.
Subtractives	Hole	Align, connect, position, mount, access, limit, assemble, support, transmit, guide, assist, allow rotation, allow passage, enclose.
	Slot	Guide, support, align, connect, stop, prevent, hold, access, transmit, assist, allow passage.
	Groove	Assist, position, limit, connect, guide, mount, space, reduce.
	Pocket/indent /depression	Position, fit, assemble, connect, guide, reduce.
	Window / cut-out	Access, align, view, position, guide, join, reduce, transmit, allow.
	Step/corner	Position, support, stop, fit, guide, reduce, allow.
Beside walls	Chamfer/fillet	Reduce, guide, assemble, assist, strengthen, align, protect, prevent.
	Groove /undercut	Strengthen, reduce, guide, limit, position, support, access.
	Rib/web	Strengthen, support, guide, hold, position, connect, access.
	Bridge	Connect, strengthen, rigidify, support, reduce, allow.
Solid elements	Block	Support, friction, block, hold, position, transmit.
	Cylinder	Support, align, connect, assemble, mount, fit, transmit, allow rotation.
	Sphere	Support, transmit, fit, reduce.
	Rod	Connect, strengthen, rigidify, support, transmit.
	Tube	Transmit, guide, connect, limit, enclose.
	Disk	Strengthen, support, align, space, reduce, friction.
	Ring	Strengthen, support, align, space, reduce, seal.

## BIBLIOGRAPHY

1. Wang, L., Shen, W., Xie, H., Neelamkavil, J., and Parasani, A., “Collaborative conceptual design – state of the art and future trends”, Computer-Aided Design, Vol. 34, (2002), pp. 981-996.
2. Pahl, G., and Beitz, W., Engineering Design – A Systematic Approach, Second Edition, Springer-Verlag, London, 1996.
3. Hsu, W., and Woon, I., “Current research in the conceptual design of mechanical products”, Computer-Aided Design, Vol. 30, (1998), pp. 377-389.
4. Deng, Y., Tor, S., and Britton, G., “A computerized design environment for functional modeling of mechanical products”, Proceedings of the fifth ACM symposium on solid modeling, Ann Arbor, Michigan, (1999), pp. 1-12.
5. Deng, Y., Britton, G., Tor, S., “A design perspective of mechanical function and its object-oriented representation scheme ”, Engineering with Computers, Vol. 14, (1998), pp. 309-320.
6. Cutherell, D., Product Architecture, The PDMA Handbook of New Product Development, M. D. Rosenan, Ed., John Wiley and Sons, 1996.
7. Lotter, B., Manufacturing Assembly Handbook, Butterworths, Boston, (1986).
8. Gardan, Y., Lanuel, Y., Pallez, D., Vexo, F., “A methodology for a function-to-shape translation tool in foundry ”, Computers in Industry, Vol. 44, (2001), pp. 117-130.
9. Gorti, S., Sriram, R., D., “From symbol to form: a framework for conceptual design”, Computer-Aided Design, Vol. 28, No. 11, (1996), pp. 853-870.
10. Schulte, M., Weber, C., “The relationship between function and shape”, Proceedings of the International conference on Engineering Design (ICED’93), The Hague, The Netherlands, (1993), pp. 9-20.
11. Cross, N. P., Engineering Design Methods: Strategies for Product Design, Third Edition, John Wiley & Sons, LTD, Chichester, England, (2000).
12. Suh, N. P., The Principles of Design, Oxford University Press, New York, (1990).

13. Shah, J., Summers, J., Hernandez, N., Zhao, Z., Lacroix, Z., “Comparative study of representation structures for modeling function and behavior of mechanical devices”, Proceedings of Computers and Information in Engineering Conference (DETC’01), Pittsburgh, Pennsylvania, (2001), pp. 1-14.
14. Ullman, D., The Mechanical Design Process, McGraw-Hill, New York, (1992).
15. Ulrich, K., Eppinger, S., Product Design and Development, McGraw-Hill, New York, (1995).
16. Dixon, J., Poli, C., Engineering Design and Design for Manufacturing: a structured approach, Filed Store Publishers, Conway. Massachusetts, (1995).
17. Umeda, Y., Tomiyama, T., “Functional reasoning in design ”, IEEE Expert Intelligent Systems, Vol. 12, Issue 2, (1997), pp. 42-48.
18. Cole, E., “Functional analysis: a system conceptual design tool ”, IEEE Transactions on Aerospace and Electronic systems, Vol. 34, No. 2, (1998), pp. 354-365.
19. Tomiyama, T., Umeda, Y., Yoshikawa, H., “A CAD for functional design ”, Annals of the CIRP, Vol. 42, No. 1, (1993), pp. 143-164.
20. Stone, R., Wood, K., “Development of a functional basis for design”, Journal of Mechanical Design, Vol. 122, (2000), pp. 359-370.
21. Keuneke, A., “Device representation: the significance of functional knowledge”, IEEE Expert Intelligent Systems, Vol. 6, (1991), pp. 22-25.
22. Hundal, M., “A systematic method for developing function structures, solutions and concept variants”, Mechanism and Machine Theory, Vol. 25, No. 3, (1990), pp. 243-256.
23. Johnson, A. L., “Designing with functions ”, Design Studies, Vol. 12, No. 1, (1991), pp. 51-57.
24. Thoma, J., Simulation by Bondgraphs- Introduction to a graphical Method, First Edition, Springer-Verlag, Berlin, (1990).
25. Schmekel, H., “Functional models and design solutions”, Annals of the CIRP, Vol. 38, (1989), pp. 129-132.
26. Szykman, S., Racz, J., Sriram, R., “The representation of function in computer-based design”, Proceedings of the 1999 ASME Design Engineering Technical Conferences, Las Vegas, Nevada, (1999), pp. 233-246.

27. Roy, U., Pramanik, N., Sudarsan, R., Sriram, R., Lyons, K., "Function-to-form mapping: model, representation and applications in design synthesis", Computer-Aided Design, Vol. 33, No. 10, (2001), pp. 699-719.
28. Mukherjee, A., Liu, C. R., "Conceptual design, manufacturability evaluation and preliminary process planning using function-form relationships in stamped metal parts", Robotics & Computer-Integrated Manufacturing, Vol. 13, No. 3, (1997), pp. 253-270.
29. Muogboh, O. S., Supporting Functionality-Based Design in Computer-Aided Design Systems, Ph.D. Dissertation, School of Engineering, University of Pittsburgh, (2003).
30. Xu, Z., Frazer, J., Tang, M., "Novel design methodology supporting product life-cycle design", Computers in Industry, Vol. 49, (2002), pp. 253-265.
31. Schulte, M., Weber, C., Stark, R., "Functional features for design in mechanical engineering", Computers in Industry, Vol. 23, (1993), pp. 15-24.
32. Feng, C., Huang, C., Kusiak, A., Li, P., "Representation of function and features in detailed design", Computer-Aided Design, Vol. 28, No. 12, (1996), pp. 961-971.
33. Shapiro, V., Voelcker, H., "On the role of geometry in mechanical design", Research in Engineering Design, Vol. 1, (1989), pp. 69-73.
34. Rosenman, M. A., "The generation of form using an evolutionary approach", Artificial Intelligence, J. S. Gero and F. Sudweeks (Eds.), Kluwer Academic Publisher, Dordrecht, The Netherlands, (1996), pp. 643-662.
35. Iyengar, G., Lee, C., Kota, S., "Towards an objective evaluation of alternate designs", Journal of Mechanical Design, Vol. 116, (1994), pp. 487-492.
36. Gardan, Y., Minich, C., Pallez, D., "On shape to specifications adequacy", Proceedings of the IEEE International Conference on Information Visualization, Londres, (1999), pp. 315-320.
37. Huang, G. Q., Mak, K. L., "Web-based collaborative conceptual design", Journal of Engineering Design, Vol. 10, No. 2, (1999), pp. 183-194.
38. Hsu, W., Liu, B., "Conceptual design: issues and challenges", Computer-Aided Design, Vol. 32, (2000), pp. 849-850.
39. Wallace, D. R., Jakiela, M. J., "Automated product concept design: unifying aesthetics and engineering", IEEE Computer Graphics & applications, Vol. 13, No. 4, (1993), pp. 66-75.
40. Anthony, L., Regli, W., John, J., Lombeyda, S., "An approach to capturing structure, behavior, and function of artifacts in computer-aided design", Journal of Computing and Information Science in engineering, Vol. 1, (2001), pp. 186-192.

41. Al-Hakim, L., Kusiak, A., Mathew, J., "A graph-theoretic approach to conceptual design with functional perspectives", Computer-Aided Design, Vol. 32, (2000), pp. 867-875.
42. Brunetti, G., Golob, B., "A feature-based approach towards an integrated product model including conceptual design information", Computer-Aided Design, Vol. 32, (2000), pp. 877-887.
43. Qin, S. F., Wright, D. K., Jordanov, I. N., "From on-line sketch to 2D and 3D geometry: a system based on fuzzy knowledge", Computer-Aided Design, Vol. 32, No. 14, (2000), pp. 851-866.
44. Deng, Y., Britton, G., Tor, S., "Constraint-based functional design verification for conceptual design", Computer-Aided Design, Vol. 32, (2000), pp. 889-899.
45. Unger, M., Ray, S., "Feature-based process planning in the AMRF", Proceedings of the ASME Computers in Engineering Conference, (1988), pp. 245-626.
46. Shah, J., Rogers, M., "Functional requirements and conceptual design of the feature-based modeling system", Computer-Aided Engineering Journal, Vol. 5, (1988), pp9-15.
47. Shah, J., "Assessment of features technology", Computer-Aided Design, Vol. 23, No. 5, (1991), pp. 331-343.
48. Shah, J., "Conceptual development of form features and feature modelers", Research in engineering Design, Vol. 2, (1991), pp. 93-108.
49. McGinnis, B., Ullman, D., "The evolution of commitments in the design of a component", Journal of Mechanical Design, Vol. 114, (1992), pp. 1-7.
50. Salomons, O., Van Houten, F., Kals, H., "Review of research in feature-based design", Journal of Manufacturing Systems, Vol. 12, No. 2, (1993), pp. 113-132.
51. Liu, H.C., and Nnaji, B.O., "Design with Spatial Relationships," Journal of Manufacturing Systems, Vol 10, No. 6, 1991, pp 449-463.
52. Rembold, U., Nnaji, B., Storr, A., Computer Integrated Manufacturing and Engineering, Addison-Wesley, (1993), pp. 345.
53. Rosen, D., "Feature-based design: four hypotheses for future CAD systems", Research in Engineering Design, Vol. 5, No. 3, (1991), pp. 125-138.
54. Zamanian, M., Fenves, S., Thewalt, C., Finger, S., "A feature-based approach to structural design", Engineering with computers, Vol. 7, (1991), pp. 1-9.



55. Baxter, J., Juster, N., dePennington, A., "A functional data model for assemblies used to verify product design specifications", Proceedings of the Institution for Mechanical Engineers, Part B- Journal of engineering Manufacture, Vol. 208, (1994), pp. 235-244.
56. Krause, F., Kimura, F., Kjellberg, T., Lu, S. C., "Product modeling", Annals of the CIRP, Vol. 42, No. 2, (1993), pp. 695-706.
57. Gu, P., Chan, K., "Product modeling using STEP", Computer-Aided Design, Vol. 27, (1995), pp. 163-179.
58. Bradley, H., Maropoulos, P., "A relation-based product model for computer-supported early design assessment", Journal of Materials Processing Technology, Vol. 76, (1998), pp. 88-95.
59. Arai, E., Iwata, K., "Product modeling system in conceptual design of mechanical products", Robotics & Computer-Integrated Manufacturing, Vol. 9, No. 4/5, (1992), pp. 327-334.
60. Tay, F. E. Gu, J., "Product modeling for conceptual design support", Computers in Industry, Vol. 48, (2002), pp. 143-155.
61. Ambler, A. P. Popplestone, R. J., "Inferring the Positions of Bodies from Specified Spatial Relationships", Artificial Intelligence, Vol. 6,(1997), pp. 157-174.
62. Gardan, Y., Minich, C., Pallez, D., Perrin, E., "Towards a specification-to-shape translation tool", Proceedings of the third International Symposium on Tools and Methods of Competitive Engineering- TMCE'2000, Delft, The Netherlands, (2000), pp. 373-382.
63. Szyman, S., Sriram, R. D., Bochenek, C., Racz, J., Senfaute, J., "Design repositories: engineering designer's new knowledge base", IEEE Intelligent Systems & Their Applications, Vol. 15, No. 3, (2000), pp. 48-54.
64. Suh, N. P., "Axiomatic design theory for systems", Research in Engineering Design, Vol. 10, (1998), pp. 189-209.
65. Szykman, S., Sriram, R. D., Regli, W., "The role of knowledge in next-generation product development systems", Journal of Computing and Information Science in Engineering, Vol. 1, (2001), pp. 3-11.
66. Pallez, D., Dartigues, C., Ghodous, P., Martinez, M., "Data architecture for collaborative conceptual design", 8<sup>th</sup> IEEE International Conference on Emerging Technologies and Factory Automation, France, (2001), pp. 597-602.
67. Finkelstein, A., Nuseibeh, B., Finkelstein, L., Huang, J., "Technology transfer: software engineering and engineering design", Computer & Control Engineering Journal, Vol. 3, Issue 6, (1992), pp. 259-265.

68. Kumara, S., Kamarthi, S., “Function-to-structure transformation in conceptual design: an associative memory-based paradigm”, Journal of Intelligent Manufacturing, Vol. 2, (1991), pp. 281-292.
69. Hirtz, J., Stone, R., McAdams, D., Szykman, S., Wood, K., “A functional basis for engineering design: reconciling and evolving previous efforts”, Research in Engineering Design, Vol. 13, (2002), pp. 65-82.
70. Gu, P., Chan, K., “Product modeling using STEP”, Computer-Aided Design, Vol. 27, (1995), pp. 163-179.
71. Fowler, T. C., Value analysis in Design, Van Nostrand Reinhold, New York, (1990).
72. Yan, H., Creative Design of Mechanical Devices, Springer-Verlag Ltd., Singapore, (1998), pp. 17-31.
73. Hibbeler, R. C., Engineering Mechanics: Statics and Dynamics, Prentice-Hall, New Jersey, (1998).
74. Wang, Y., Nnaji, B. O., “Functionality-Based Modular Design for Mechanical Product Customization Over the Internet”, Journal of Design and Manufacturing Automation, Vol. 1, (2001), pp. 107-121.
75. Nnaji, B. O., Liu, H. C., and Rembold, U., “A Product Modeller for Discrete Components”, International Journal of Production Research, Vol. 31, No. 9, (1993), pp. 2017-2044.
76. Hix, C. F., and Alley, R. P., Physical Laws and Effects, John Wiley & Sons Inc., New York, (1958).
77. Collins, J. A., Failure of Materials in Mechanical Design: Analysis, Prediction, Prevention, Wiley Interscience, New York, (1993).
78. Walker, I., “Requirements of an object-oriented design method”, Software Engineering Journal, Vol. 7, (1992), pp. 102-113.
79. [Http://www.mysql.com](http://www.mysql.com)
80. [Http://www.php.net](http://www.php.net)
81. Fitch, P., and Cooper, J. S., “Life-Cycle Modeling for Adaptive and Variant Design. Part 1: Methodology ”, Research in Engineering Design, Vol. 15, (2005), pp. 216-228.
82. Fitch, P., and Cooper, J. S., “Life-Cycle Modeling for Adaptive and Variant Design. Part 2: Case Study ”, Research in Engineering Design, Vol. 15, (2005), pp. 229-241.

83. Wood, S. L., An Architecture for a Function Driven Mechanical Design Solution Library, Ph. D. Dissertation, Oregon State University, (1994).
84. Sudarsan, R., Fenves, S. J., Sriram, R. D., and Wang, F., “A Product Information Modeling Framework for Product Lifecycle Management”, Computer-Aided Design, In Press, Available on line 21 March 2005, (2005), pp. 1-13.
85. Farag, M. M., Materials Selection for Engineering Design, Prentice Hall, New Jersey, (1997), pp. 229-231.
86. Bo, Y., and Salustri, F. A., “Function Modeling Based on Interactions of Mass, Energy, and Information”, American Association for Artificial Intelligence, (A. N. Kumar, and I. Russell Eds.), Proc. Flairs 99, (1999), pp. 384-388.
87. Bronsvort, W. F., Noort, A., “Multiple-View Feature Modeling for Integral Product Development”, Computer-Aided Design, Vol. 36, Issue 10, (2004), pp. 929-946.
88. Szykman, S., Senfaute, J., and Sriram, R. D., “The Use of XML for Describing Functions and Taxonomies in Computer-Based Design”, Proceedings of the 1999 ASME Design Engineering Technical Conferences, Las Vegas, Nevada, (1999), pp. 945-955.
89. Kurakawa, K., “A Scenario-Driven Conceptual Design Information Model and Its Formation”, Research in Engineering Design, Vol. 15, (2004), pp. 122-137.